

1. EDM Home	6
1.1 Overview	6
1.1.1 Database - EDM	8
1.2 Installation Guide	8
1.2.1 EDM Installation and Setup	9
1.2.1.1 EDM Install	9
1.2.1.2 Loading Remote Location Data Directly	12
1.2.1.3 Setting up Multiple Companies	12
1.2.1.4 Apache Tomcat Web Server	13
1.2.1.5 Using Microsoft IIS as the Primary Web Server	13
1.2.1.6 Allowing Full Access to Your System	14
1.2.1.7 Using Secure Socket Layer (SSL) to Protect Data	15
1.2.1.8 Increasing Central Site Memory	20
1.2.1.9 Increasing Remote Site Memory	20
1.2.1.10 Uninstalling the Central System	20
1.2.1.11 Uninstalling a Remote System	20
1.2.1.12 Hiding the Applet Warning Icon	20
1.2.2 EDM Upgrade Installation	21
1.2.3 EDM Silent Installation Instructions	23
1.2.4 EDM Interactive Installation Instructions	33
1.2.5 Configuring the Xenial Connector	45
1.3 Release Notes	46
1.3.1 EDM 7.10 - Release Version - 2015-11-20	46
1.3.2 EDM 7.11.33 - Release Version - 2015-02-03	48
1.3.3 EDM 7.12.223 - Release Version - 2016-09-07	50
1.3.4 EDM 7.13 - Release Version - 2017-06-21	53
1.3.5 Release Notes Archive	54
1.3.5.1 EDM 6.0 Release Notes	55
1.3.5.1.1 EDM 6.9 Release Notes	55
1.3.5.1.2 EDM 6.8 Release Notes	57
1.3.5.1.3 EDM 6.7 Release Notes	57
1.3.5.1.4 EDM 6.6 Release Notes	58
1.3.5.1.5 EDM 6.5 Release Notes	60
1.3.5.2 EDM 7.0 Release Notes	61
1.3.5.2.1 EDM 7.9 Release Notes	61
1.3.5.2.2 EDM 7.8.100 Release Notes	61
1.3.5.2.3 EDM 7.8.30 Release Notes	62
1.3.5.2.4 EDM 7.8.27 Release Notes	62
1.3.5.2.5 EDM 7.8.24 Release Notes	62
1.3.5.2.6 EDM 7.8 Release Notes	63
1.3.5.2.7 EDM 7.7 Release Notes	63
1.3.5.2.8 EDM 7.6 Release Notes	64
1.3.5.2.9 EDM 7.5 Release Notes	67
1.3.5.2.10 EDM 7.4 Release Notes	67
1.3.5.2.11 EDM 7.3.86 Release Notes	67
1.3.5.2.12 EDM 7.3.36 Release Notes	68
1.3.5.2.13 EDM 7.2 Release Notes	68
1.3.5.2.14 EDM 7.1.12 Release Notes	72
1.3.5.2.15 EDM 7.1.11 Release Notes	73
1.3.5.2.16 EDM 7.1.10 Release Notes	76
1.3.5.2.17 EDM 7.1.9 Release Notes	76
1.3.5.2.18 EDM 7.1.8 Release Notes	77
1.3.5.2.19 EDM 7.1.7 Release Notes	77
1.3.5.2.20 EDM 7.1.6 Release Notes	78
1.3.5.2.21 EDM 7.1.5 Release Notes	78
1.3.5.2.22 EDM 7.1.4 Release Notes	78
1.3.5.2.23 EDM 7.1.3 Release Notes	82
1.3.5.2.24 EDM 7.1.2 Release Notes	85
1.3.5.2.25 EDM 7.1.1 Release Notes	89
1.3.5.2.26 EDM 7.1.0.108 Release Notes	90
1.3.5.2.27 EDM 7.1.0.70 Release Notes	91
1.3.5.2.28 EDM 7.0.8 Release Notes	92
1.3.5.2.29 EDM 7.0.7 Release Notes	93
1.3.5.2.30 EDM 7.0.6 Release Notes	96
1.3.5.2.31 EDM 7.0.5 Release Notes	98
1.3.5.2.32 EDM 7.0.4 Release Notes	98
1.3.5.2.33 EDM 7.0.3 Release Notes	100
1.3.5.2.34 EDM 7.0.2 Release Notes	101
1.3.5.2.35 EDM 7.0.1 Release Notes	102
1.4 User Guide	104
1.4.1 Getting Started	105
1.4.1.1 Starting a Central System	106

1.4.1.2 Starting a Remote System	106
1.4.1.3 Using the Menu	107
1.4.1.4 Modifying the Menu	108
1.4.1.5 Using an LDAP with EDM	109
1.4.2 Managing Sites	111
1.4.2.1 Creating a Site	111
1.4.2.2 Location Data	113
1.4.2.2.1 Copying Location Data	113
1.4.2.2.2 Sharing Location Data	114
1.4.2.2.3 Unsharing Location Data	116
1.4.2.2.4 Deleting Location Data	116
1.4.2.3 Creating a Database for a New Remote Location	116
1.4.2.4 Sending a Database to a New Remote Location	117
1.4.2.4.1 Sending Data to a New Remote Location in a Transaction	117
1.4.2.5 Keeping Central and Remote Data Synchronized	117
1.4.3 Data and Transactions	117
1.4.3.1 Maintaining Remote Data	118
1.4.3.2 Override Flags	118
1.4.3.3 Transactions	119
1.4.3.3.1 Data Changes and Transaction Files	119
1.4.3.3.2 Generating Transactions	119
1.4.3.3.3 Transmitting Transactions	119
1.4.3.3.4 Receiving Transactions	119
1.4.3.3.5 Processing Transactions	120
1.4.3.3.6 Recording Conflicts	120
1.4.3.3.7 Supporting Multiple Application Versions	120
1.4.3.3.8 Audit Trails	120
1.4.3.3.9 Transaction File Numbers	120
1.4.3.3.10 Reset Transactions	124
1.4.3.4 Import-Export Features	125
1.4.3.5 Central Data Import	125
1.4.3.6 International Language Support	127
1.4.3.7 Data Recovery	127
1.4.3.7.1 Recovering Central System Data	128
1.4.3.7.2 Recovering Central Application Data	128
1.4.3.7.3 Recovering Remote System Data	129
1.4.3.7.4 Recovering Remote Application Data	129
1.4.3.7.5 Recovering Transaction Files	130
1.4.3.8 Database Purge Process	130
1.4.3.9 Validating Data	130
1.4.4 Editing Data	131
1.4.4.1 Edit Data in Tables	131
1.4.4.2 Edit Data in Queries	132
1.4.4.3 Editing Data on Forms	132
1.4.4.4 Moving Between Records	134
1.4.4.5 Changing the Column Width on Spreadsheets	135
1.4.4.6 Adding Records	135
1.4.4.7 Editing Records	136
1.4.4.8 Deleting Records	137
1.4.4.9 Saving Records	137
1.4.4.10 Undoing Changes	137
1.4.4.11 Finding Records	137
1.4.4.12 Editing Central Data	138
1.4.4.13 Editing Data with Other Applications	138
1.4.4.14 Testing Data Changes Before Transmission	139
1.4.5 Server Text File Editor	139
1.4.6 Queries	142
1.4.6.1 Creating Queries	143
1.4.6.2 Query Columns	145
1.4.6.2.1 Adding a Column from a Data Source	146
1.4.6.2.2 Renaming Columns	147
1.4.6.2.3 Creating Calculated Columns	148
1.4.6.2.4 Adding Multiple Fields as a Group	148
1.4.6.2.5 Changing Field Values	148
1.4.6.2.6 Rearranging Columns	148
1.4.6.2.7 Changing Column Widths	149
1.4.6.2.8 Inserting Columns	149
1.4.6.2.9 Deleting Columns	149
1.4.6.2.10 Viewing Table Names	150
1.4.6.2.11 Hiding Columns	150
1.4.6.2.12 Specifying Column Conditions	150
1.4.6.3 Sorting Results	154

1.4.6.4 Viewing and Editing Query Results	155
1.4.6.5 Multi-Table Queries	155
1.4.6.5.1 Joining Tables	156
1.4.6.5.2 Types of Joins Between Data Sources in a Query	156
1.4.6.5.3 Deleting a Join	157
1.4.6.5.4 Saving Joins	157
1.4.6.6 Editing the SQL Statement for a Query	158
1.4.6.7 Query Parameters	158
1.4.6.8 Calculating Totals	159
1.4.6.8.1 Calculating Totals for All Records	160
1.4.6.8.2 Calculating Totals for Groups of Records	161
1.4.6.8.3 Specifying Conditions for Groups	162
1.4.6.8.4 Specifying Conditions for Totals	162
1.4.6.8.5 Specifying Conditions Before Calculating Totals	163
1.4.6.9 Action Queries	163
1.4.6.9.1 Make Table Query	164
1.4.6.9.2 Insert Query	164
1.4.6.9.3 Update Query	166
1.4.6.9.4 Deleting Rows	167
1.4.6.10 Changing a Query Design	168
1.4.6.11 Copying Queries	168
1.4.6.12 Getting Query Parameters from a Form	168
1.4.7 Forms	169
1.4.7.1 Creating a Form	171
1.4.7.2 Designer Windows	171
1.4.7.2.1 Form Design Window	171
1.4.7.2.2 Toolbar Tools	172
1.4.7.2.3 Attribute Window	189
1.4.7.3 Controls	190
1.4.7.3.1 Basic Controls	191
1.4.7.3.2 Setting Control Attributes	195
1.4.7.3.3 Selecting Controls	203
1.4.7.3.4 Moving Controls	204
1.4.7.3.5 Sizing Controls	205
1.4.7.3.6 Copying Controls	205
1.4.7.3.7 Deleting Controls	205
1.4.7.3.8 Setting the Tab Order for Controls	205
1.4.7.3.9 Advanced Controls	206
1.4.7.4 Form Sections	216
1.4.7.5 Form Attributes	218
1.4.7.6 Events	220
1.4.7.7 Opening Forms	221
1.4.7.8 Navigating Forms	223
1.4.7.9 Record Selectors	223
1.4.8 Form Viewer	224
1.4.8.1 Using the Form Viewer	225
1.4.8.1.1 Versioning - Form Viewer	225
1.4.8.1.2 Editing Data - Form Viewer	227
1.4.9 Expressions	230
1.4.9.1 Constants	230
1.4.9.2 Referring to Object Values	231
1.4.9.2.1 Referring to Query Columns	231
1.4.9.2.2 Referring to Form Values	231
1.4.9.2.3 Object References	231
1.4.9.3 Expression Operators	232
1.4.9.4 Expression Examples	232
1.4.9.5 Functions	233
1.4.10 Client Functions	233
1.4.11 Server Functions	371
1.4.12 Format Strings	440
1.4.12.1 Format Strings for Text	440
1.4.12.2 Format Strings for Numbers	440
1.4.12.3 Format Strings for Dates and Times	443
1.4.13 File Transfer	445
1.4.13.1 Web Transfer	445
1.4.13.2 Amazon S3 Transfer	446
1.4.13.3 FTP Transfer	448
1.4.13.3.1 Setting Up an FTP Server	449
1.4.13.3.2 Central FTP Server	449
1.4.13.3.3 Internet FTP Server	450
1.4.13.3.4 Remote FTP Server	451
1.4.13.4 Batch Process for File Transfer at Central Locations	452

1.4.13.5	Batch Process for File Transfer at Remote Locations	452
1.4.13.6	Changing the Process Transactions Menu Options to also Transfer Files	453
1.4.13.7	Adding File Transfer to the Menu	453
1.4.13.8	Managing Files	453
1.4.14	Polling Data	455
1.4.14.1	Detect Changes with Snapshots	455
1.4.14.2	Refresh Entire Table	456
1.4.14.3	Refresh Partial Table	456
1.4.14.4	Polling Aloha POS (TM) Data	457
1.4.14.4.1	Remote Setup	457
1.4.14.4.2	Central Setup	458
1.4.14.4.3	Managing Polling	458
1.4.15	Security	459
1.4.15.1	Internet Security	460
1.4.15.2	Application Security	460
1.4.15.3	Permission Types	460
1.4.15.4	Permissions	461
1.4.15.5	Roles	461
1.4.15.6	Assigning Permissions to Roles	462
1.4.15.7	Editing Security Information	462
1.4.15.8	File Security	463
1.4.15.9	Preventing Embedded Executable Scripts	463
1.4.16	Supported Databases	463
1.4.16.1	dBASE IV Using the DataDirect ODBC Driver	463
1.4.16.2	dBASE IV Using the Microsoft ODBC Driver	463
1.4.16.3	Microsoft Access 97	463
1.4.16.4	Microsoft Access 2000	463
1.4.16.5	Microsoft SQL Server 2000 Using JDBC Driver	464
1.4.16.6	Microsoft SQL Server 2005 Using JDBC Driver	464
1.4.16.7	Microsoft SQL Server Using ODBC Driver	465
1.4.16.8	MSDE Using JDBC Driver	465
1.4.16.9	Oracle	466
1.4.16.10	PostgreSQL	467
1.4.17	Configuration Files	468
1.4.17.1	Config.xml	468
1.4.17.1.1	Setting Property Values for LDAP	475
1.4.17.1.2	Getting Property Values from the Windows Registry	475
1.4.17.1.3	Adding an Additional Company	475
1.4.17.2	Snapshots	476
1.4.17.2.1	Adding a Snapshot	476
1.4.17.2.2	Changing a Snapshot	477
1.4.17.3	Main Menu	477
1.4.17.3.1	Editing the Main Menu	477
1.4.17.4	Other Configuration Files	478
1.4.18	Upgrading EDM or Your Application	480
1.4.18.1	Upgrading Your Application	480
1.4.18.2	Upgrading EDM from JAR to an EXE File Structure	483
1.4.18.3	Upgrading from EDM 3.1 or Later	484
1.4.18.4	Upgrading from EDM 3.0.x	485
1.4.18.5	Upgrading from EDM 2.9.x or Earlier	485
1.4.19	Frequently Asked Questions	486
1.4.20	Glossary	489
1.4.21	Extending EDM Functionality	494
1.4.22	Translating EDM System Text	500
1.4.23	Tax Rules Editor for IRIS	503
1.4.24	IRIS Config Group Editor	507
1.5	Master Data Management	518
1.5.1	How Does Master Data Management Work	519
1.5.1.1	Step 1 - Shared Table Groups	520
1.5.1.2	Step 2 - Configuration Types	523
1.5.1.3	Step 3 - Configurations	527
1.5.1.4	Step 4 - Version	529
1.5.1.5	Step 5 - Configuration Versions	531
1.5.1.6	Step 6 - Configuration Subscriptions	532
1.5.1.7	Step 7 - Version Subscriptions	536
1.5.1.8	Step 8 - Identify Sites to Deploy	542
1.5.2	Selecting the Seed Data	543
1.5.3	Updating Application Data	543
1.5.4	Updating the different data tables	544
1.5.5	Package Editor	545
1.5.6	MDM for EDM Users	545
1.5.7	Subscribing Multiple Configurations to a Site	546

1.5.8 MDM Data API	548
1.6 Basic User Guide	550
1.6.1 Editing Data Using EDM	550
1.6.2 Editing Data for Multiple Stores	550
1.6.3 Versioning	551
1.7 Java Message Service (JMS) Queues for Transaction Transfer	551
1.7.1 ActiveMQ Setup	552
1.8 Automated Setup for New Customers	553
1.9 Automated Setup of New Remote Sites	557
1.10 RESTful Web Service API	559
1.10.1 Creating a Signature for the RESTful API	576
1.11 RESTful Web Service for Authentication	577
1.12 EDM File Transactions	587
1.12.1 Send File	588
1.12.2 Request File	590
1.12.3 Managed Files	593
1.12.4 File Sharing	594
1.12.5 Synchronization	600
1.13 The EDM Build System	600
1.14 EDM Client Package Structure Changes	601
1.15 How-to articles	603
1.16 EDM to Xenial Integration	603

EDM Home

The Enterprise Data Manager (EDM) provides the capability to manage and synchronize distributed location configuration data between the EDM Central system and the remote locations.

The ability to create, update, activate and deactivate information such as inventory items, change prices, define offers, POS configuration and tax tables among all the other items that define the rules of operation for each location is a significant challenge for the multi-unit operator.

EDM was specifically designed to simplify and automate these data management tasks and dramatically reduces the effort and expense involved in the administration of the configuration information.

Contents

Overview

Introduction

Any organization that operates multiple locations has faced the problem of maintaining data for remote locations. Changes to menu item pricing, adding menu items or coupons, implementing a special program, maintaining employee data, changing inventory or recipe information, updating sales tax tables and many other examples present unique challenges for the multi-unit operator. The task of managing this information can be difficult and tiresome.

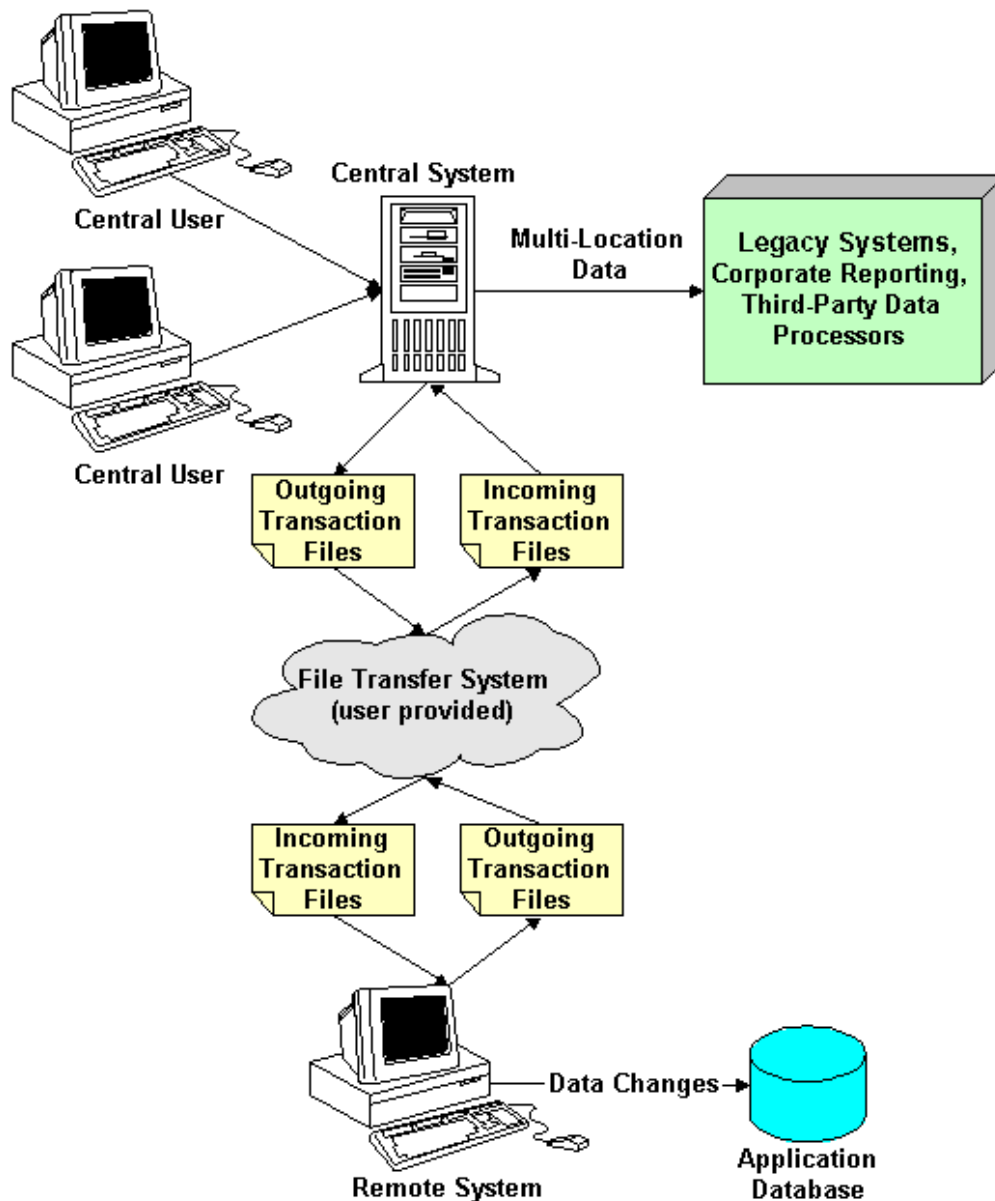
XPIENT Manager supports database formats that are accessible via Open Database Connectivity (ODBC) drivers and Java Database Connectivity (JDBC) drivers. This design enables Enterprise Data Manager to serve as a true centralized database management application. End-users are able to maintain all databases using a single application—significantly reducing the time required to maintain this data while improving change management control and accuracy.

What is Enterprise Data Manager?

Enterprise Data Manager (EDM) is a system that is designed to allow a company to maintain selected portions of the databases for multiple remote locations. For example, the corporate headquarters for a restaurant chain may want to maintain the menu items and prices for all its restaurants while leaving maintenance of employees to the restaurant managers. Changes to the databases are recorded along with an effective date so that users can enter changes to be applied at some later date. For example, users at the central location might set the price of coffee to go up to seventy-nine cents on January 1. This change will be saved but will not actually be applied to the database until the effective date of January 1.

Changes can be made to the data at any location and these changes will be automatically sent to all other locations that maintain a copy of that data thereby synchronizing all locations at all times. When the same data is changed at two different locations at the same time, a conflict is reported and the change that caused the conflict is undone.

Users can define which tables and fields are maintained at the central location, the remote location, or both. When changes are made at either the central or remote location, a transaction log is created. The transaction log is sent to the destination location via network connection, FTP connection or a user supplied mechanism. When the transaction log arrives at the destination, a utility running at that location will merge it with any other currently pending updates. Another utility will scan all pending updates and apply any whose effective date has been reached. The basis on which you activate this utility should be determined by the frequency with which data is transferred between locations.



To handle data from multiple remote locations at a central location, automatically adds a field to every table in the central database. The extra field is the location field and it will contain the location number from which each record came.

Powerful import and export capabilities are also built into the system to provide the ability to generate customer polling files or send/receive information between a location and a vendor or processing service. For example, restaurants may send a summary of the day's sales to the corporate office at the end of each day and a file containing their payroll information to the payroll processing company once per week.

Benefits

The benefits of Enterprise Data Manager are that it:

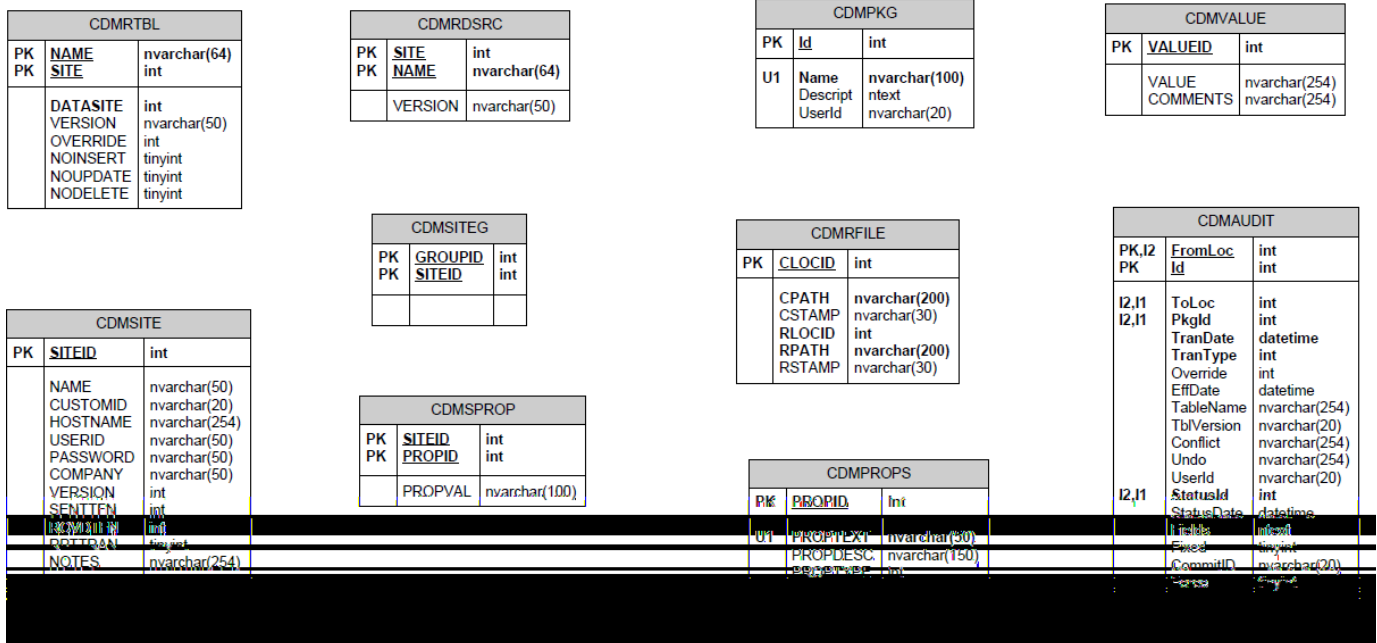
- Gives corporate users complete control over data used at remote sites
- Maintains consistent data across all locations or groups of locations
- Supports reporting with data consolidated from all or groups of locations
- Can be used to build flexible replacements for custom polling systems
- Supports corporate Decision Support Systems (DSS)
- Minimize data entry requirements to maintain data for hundreds or thousands of remote locations
- Ensures data changes are timely and accurate
- Can be used to launch promotions without setup problems at individual locations

See Also

Database - EDM

This page contains the Entity-Relationship Diagram for the EDM Core data model. Please note that these are the tables that the EDM applications requires to operate, and not the tables of the system under management (for example, IRIS).

Entity Relationship Diagram (ERD)



Installation Guide

Software Requirements

The following is a list of required software to run the system:

- Microsoft Windows 2000 Server or XP Professional
- Supported Databases

Hardware Requirements

Note

All systems must support DirectX and 256-color video at a minimum or the system cannot run.

The following table shows the required hardware to run the system at a central or remote location.

	Central Location Recommended	Remote Location Recommended
CPU	Pentium 4 HT Server Class PC	750Mhz or faster Pentium class
RAM	4GB DDR	512MB

Hard Drive Free Space	50MB for software + 10MB per location	50MB
Hard Drive Interface	SATA IDE Drives Multiple HD configured in a RAID 0 (for performance) or RAID 5 for fault tolerance and performance	SCSI U2W or ATA/66
Video Card	1024x768, 256 colors, and DirectX	1024x768, 256 colors, and DirectX

Installation Instructions

EDM Installation and Setup

The following provides instructions for installing EDM at both a central site, and a remote site. At central sites, the system is typically installed on a web server so users can access it from a web browser. If the web server is visible over the internet, users can access it from anywhere they have an Internet. At remote locations, the system is usually installed in standalone mode without a web server. Standalone mode allows users to access the system without installing a web server but it can only be used in single-user installations. The standalone mode does not support multiple users because it does not synchronize changes to cached system data among multiple users.

IMPORTANT NOTE

If you are upgrading EDM from Apache 6/7 using an SSL connection, you must make the following change to the server.xml file located in the \EDMServer\conf folder. The headers in the Connector protocol section that include the certificate information must read as follows:

```
keystoreType="",keystoreFile="", and keystorePass=""
```

The EDM service will not start until after you make this change.

Additional Topics

EDM Install

Overview

This page describes the EDM installation process. Installation instructions are provided for both a central system and a remote system.

Notes

- The EDM installation files are located on the XPIENT FTP site. Contact your Professional Services representative if you do not know your login credentials.
- The SA username and password is used with the install.
 - Once the database connects to SQL, you can encrypt the SA password in the [Config.xml](#) file using the *encryptedPassword* setting.
- To implement Integrated Security, the user must have Windows credentials with the appropriate permissions.

Installing a Central System

The system has been tested with the Apache Tomcat Servlet Container version 6.x, 7.x and 8.x. Note that the application being maintained by the system, such as the POS system, does not have to be installed on the server. If you already have the Apache Tomcat Web Server installed, you will need to set its folder in the setup batch file before continuing.

Running the Installation

Run the executable file that begins with **EDM-Server-Setup** and follow the prompts to install for your system. See [EDM Interactive Installation Instructions](#). By default, the system will be installed in the folder **C:\EDMServer**.

Authentication

Authentication for EDM is done from a separate database. All users are migrated from the company database into a new authentication database named "auth" by default.

When you install EDM you are prompted to specify the location of the authentication database. This database contains the mapping of the users to the companies that they have access to, and is maintained by the User editor. If you select the **Use system data source info** option from this new prompt, an Authentication data source is not created. EDM will use the system datasource to create one itself.

Click [here](#) to read more about the API used to authenticate with EDM.

Testing the Installation

1. Test your installation by opening your browser to: `http://localhost:8080/EDM` (if you are not on the same machine as the server, replace **localhost** with the server name or IP address). Login using userid **admin** and password **admin**. At installation, the system has the following predefined user id/password combinations:
 - a. **remote/remote** - Typical login for a user at a remote site who only has permission to process transactions.
 - b. **central/central** - Typical login for a central user who has permission to use all the forms and functions at headquarters.
 - c. **admin/admin** - Administrative login with all permissions including setting up new users and roles and modifying the forms and queries in the application.
2. The Tomcat Web Server is installed as a service, so, to start, stop, or restart Apache Tomcat, use the Services option under Administrative Tools in the Control Panel.
3. To use the Tomcat Manager to reload the application after updates or configuration changes, add a user to the `conf\tomcat-users.xml` file in the Tomcat folder with the following format: `<user username="me" password="mypassword" roles="tomcat,manager,admin"/>` (change the username and password as necessary), restart Tomcat, browse to `http://localhost:8080/manager/html` (replace localhost with server name if necessary), enter your userid and password, and click **Reload** on the line that has the `/EDM` path.

Support processing transactions on a separate server

Users may set up a 'transaction server' in addition to the main web server. The transaction server handles inserting rows into the audit table, committing packages, and processing incoming transactions. Using a separate server for transaction processing relieves the main web server of those tasks and improves performance on the main web server. The main web server checks the transaction server's status before forwarding each command to verify that it is available and is running the same version of the software as the main web server. If the transaction server is not available then transaction processing will be performed on the main web server. Since committing and transaction processing may occur on either server, it is important to set the `localSendDir` and `localReceiveDir` elements in `config.xml` to refer to the same server and directory on both the transaction server and the main web server.

To configure the web server to forward transaction processing tasks to a separate transaction server take the following steps:

1. Set a separate server running Tomcat and copy the `webapps\EDM` folder from the main server to the transaction server.
2. Set the `localSendDir` and `localReceiveDir` elements in `config.xml` to refer to the same server and directory on both the transaction server and the main web server.
3. Install a valid license file on the transaction server and restart the Tomcat service on the transaction server and check the `log.txt` file to ensure a successful start.
4. On the main web server, add a line to `config.xml` for each company that is to forward transaction processing commands to the transaction server like this:
`transactionServerURL = "http://transerver:8080/EDM"`
5. On the main web server, remove the schedule tasks for processing transactions and purging the audit trail since these will now run on the transaction server.
6. Restart the Tomcat service on the main web server.
7. Configure all remote sites to communicate directly with the transaction server when transferring transaction files.

If command forwarding is working properly, you will see a line in the `log.txt` file on the main web server similar to the following the first time you commit or process transactions:

```
INFO Connected to transaction server http://transerver:8080/EDM/xmlcommand.do.
```

If the connection to the transaction server fails, you will see a warning message in the log like this

```
WARNING Failed to connect to transaction server http://transerver:8080/EDM/xmlcommand.do.
```

and the transaction processing will be performed on the main web server.

If transaction server is running a different software version than the main web server, you will see a warning message in the log like this

```
WARNING Invalid software version on transaction server: Version 6.2 at  
http://transerver:8080/EDM/xmlcommand.do, must match the software version on this server:  
Version 6.3.
```

and the transaction processing will be performed on the main web server.

If main web server is configured to forward messages to itself, you will see a warning message in the log like this

```
WARNING This server is configured to forward to itself, forwarding is ignored.
```

and the transaction processing will be performed on the main web server.

Although it is possible to configure the transaction server to forward transaction processing commands to yet another server, this will reduce performance and leads to the possibility of creating an endless circular forwarding loop.

Installing Remote System


The system is usually installed in standalone mode at remote sites because there is no need to support more than one user at a time. For a convenient lab system, the remote system may be installed on the same computer as the central system.

To install the standalone remote system, take the following steps:

1. Run the executable file that begins with **EDM-Remote-Setup** and follow the prompts to install for your system. By default, the system will be installed in the folder **C:\EDMWeb**.
2. Both 32 bit and 64 bit versions of EDM are available. The 64 bit version has "64" in the filename.
3. You can start the system for the remote location by double-clicking **C:\EDMWeb\go.bat**.

Creating Central Tables

Because the central system does not have direct access to the application (such as the POS system), the system must request the table list and table structures from a remote site to create the central database tables. Once the system is installed at the first remote site, take the following steps to create the application tables in the central database:

1. At the central site, add a site record for the site from which you plan to get the application table list by double-clicking **Edit Locations And Groups** on the **Locations** branch of the main menu. Click the add record  on the tool bar and enter the site number and name, then close the site form.
2. Double-click **Request Table List** on the **Transactions** branch of the main menu, select the new site and click **OK**, click **OK** again to select the default package, click **Yes** to commit the transaction.
3. Configure the remote site to be able to communicate with the central web server to transfer transaction files. Reference [Web Transfer](#) for detailed instructions.
4. Transfer the transaction file (1.xml) from the **Outgoingsitenumber** folder to the **Incoming\0** folder at the remote site by double-clicking **transfer_web.bat** in the installation folder (usually C:\EDMWeb).
5. At the central site, click **Process All Transactions** on the **Transactions** branch of the main menu to process the incoming table list and create the central application tables.


Supporting Additional Data Sources

If you already have a system set up with central tables and you want to support additional data sources at the remote sites, take the following steps:

1. Backup the EDMWeb database and C:\EDMWeb folder at the site.
2. Add a new <dataSource section to the config.xml at the store that has a unique name and appropriate settings for server name, database, userid, and password.
3. Run go.bat so EDM loads the new dataSource into the local schema.xml file.
4. Backup the EDMWebHQ database and jakarta...\webapps\EDM\web-inf\classes folder at central.
5. Use Transactions - Request Table List from central to get the new table structures set up at central. This step only needs to be done once.
6. Use Transactions - Request Table Refresh from central to load data for the new table. This step needs to be repeated for each site.

Requesting Data from Remote Sites

Once the central database tables have been created using the instructions in the previous section, you can load site data for each remote site by taking the following steps:

1. At the central site, add a site record for each site from which you plan to load data by double-clicking **Edit Locations And Groups** on the **Locations** branch of the main menu. Click the add record  on the tool bar and enter the site number and name. When finished, close the site form.
2. Click **Request Table Refresh** on the **Transactions** branch of the main menu, select **POS v.v.v Configuration Tables** (where v.v.v is the version of POS you are using) and click **Ok**. Select the site and click **Ok**. Click **Ok** again to select the default package. Click **Yes** to commit the transaction.
3. Transfer the transaction files from the **Outgoingsitenumber** folder to the **Incoming\0** folder at the remote site by double-clicking **transfer_web.bat** in the installation folder (usually C:\EDMWeb).
4. At the central site, click **Process All Transactions** on the **Transactions** branch of the main menu to process the incoming table refresh which loads the site data into the central database.

Loading Remote Location Data Directly

Optionally, users can load location data into the central database by physically bringing the location data to the central location and loading it directly. This is normally only used in lab settings as it is problematic to bring entire application databases from a large number of remote sites.

Remote Sites

For remote sites running IRIS, take the following steps:

1. Backup the SQL Server IRIS database at the remote location.
2. Restore the database on a SQL Server system that is visible to the central system.

Central Site

At the central site, take the following steps to add a site record for each site from which you plan to load data:

1. Select the **Sites** module.
2. Select **Edit Locations And Groups** from the **Setup** menu.
3. Click the plus sign on the tool bar to add a new record.
4. Type the site number and name in the provided fields.
5. Select **Save Record** from the **Records** menu.
6. Click **Load Location Data** on the **Location** branch of the menu.
7. Select the new location.

Setting up Multiple Companies

The system allow you to have more than one company on the central server. For example, if you have a group of hamburger restaurants and a group of coffee shops that are maintained separately, you could have two separate companies set up. Each company has its own database of sites and data so that there can be a site 1 in both companies. To set up an additional company take the following steps:

1. Edit your `Config.xml` file (by default it is in the folder `C:\EDMServer\webapps\EDM\WEB-INF\classes`).
2. Copy the existing company element to the clipboard as shown below (from the `<company` line to the `</company>` line inclusive).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE config>

<config
  version = "3">
  <company
    .
    .
    .
  </company>
</config>
```

3. Paste the copied company in after the `</company>` line and before the `</config>` line as shown below so that you have two **company** sections inside the **config** section.

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE config>

<config
  version = "3">
  <company
    .
    .
    .
  </company>
  <company
    .
    .
    .
  </company>
</config>

```

4. Edit the properties in the second company section as follows (use an appropriate name instead of COFFEE):

centralSiteName = "COFFEE HQ"
thisSiteName = "COFFEE HQ"
schemaFile = "schema_iris_COFFEE.xml"
(In the systemDataSource section) database = "EDMWebCOFFEEHQ"

5. Restart the **Apache Tomcat** service.

Apache Tomcat Web Server

If you plan to use the Apache Tomcat Web Server that was installed with the system as the primary web server instead of an Apache or Microsoft or some other web server, you will want to change the default HTTP port for Tomcat from 8080 (it's default value) to 80 (the standard HTTP port).

1. Use Windows Notepad to edit *TomcatFolder\conf\server.xml*.
2. Find the line that begins with **<Connector port="8080"**
3. Change **8080** to **80**, so that the line now begins with **<Connector port="80"**
4. From the **Windows - Start** menu, go to **Control Panel>Administrative Tools>Services**.
5. Click the **Apache Tomcat** service, and then click **Restart** or **Start** Service.
6. Now you can use the central system by opening your browser to: <http://localhost/EDM> (if you are not on the same machine as the server, replace **localhost** with the server name or IP address).

Using Microsoft IIS as the Primary Web Server

If you are already using Microsoft Internet Information Services as your web server, you can integrate Tomcat with IIS so that users can access the system through IIS. Tomcat is still required because IIS does not include support for Java Server Pages (JSP) or Java Servlets.

To configure IIS to allow Tomcat to handle JSP's and Servlets, take the following steps.

Getting Started

1. From the **Windows - Start** menu, go to **Control Panel>Administrative Tools**.
2. Double-click **Internet Information Services**.

Configuring the Default Web Site

1. Right-click **Default Web Site**, and then select **New>Virtual Directory** from the menu that appears.
2. Click **Next**.
3. Type **jakarta** as the alias, and then click **Next**
4. Click **Browse** and select *TomcatFolder\bin\win32\386*. Click **Next**.
5. Uncheck the **Read** option. Check the **Execute** option.
6. Click **Next**, and then click **Finish**.

7. Right-click **Default Web Site**, and then select **Properties** from the menu that appears.
8. Click the **ISAPI Filters** tab, and then click **Add**.
9. Type **jakarta** as the filter name.
10. Click **Browse**, and then select **TomcatFolder\bin\win32\i386\isapi_redirect.dll**.
11. Click **OK**. Click **OK** again to close the **Properties** window.

Configuring the Web Sites

Follow the steps in this section if you are running IIS 6 (or higher).

1. Right-click **Web Sites**, and then select **Properties** from the menu that appears.
2. Click the **Service** tab. Check the **IIS 5.0 isolation mode** option, and then click **OK**.
3. Click **Web Service Extensions**. Select **Add a new Web service extension**.
4. Type **jakarta** as the extension name.
5. Check the checkbox for **Set extension status to Allowed**.
6. Click the **Add** button. Click **Browse** and select **TomcatFolder\bin\win32\i386\isapi_redirect.dll**.
7. Click **OK**. Click **OK** again to close the extension window.

Restarting IIS

1. Right-click the **Local Computer**, and then select **All Tasks – Restart IIS...** from the menu that appears.
2. Close the IIS management window.

Testing the Installation

Test your installation by opening your browser to: <http://localhost/EDM> (if you are not on the same machine as the server, replace **localhost** with the server name). Login using **admin** as the user ID and password.

Allowing Full Access to Your System

When it is not running in standalone mode, the system runs as an applet. Therefore, it is restricted from many operations, including:

- Reading or writing your hard drives
- Accessing the system clipboard
- Connecting to other computers via the Internet

If you want to allow the system to access your computer (e.g. to copy data to and from the clipboard or update a local database), then you must give it explicit permission to do so.

To give the system permission to access your system, you must first create a security policy file on the server containing permissions, and then update each user's Java Runtime Environment to use the new security policy file.

Creating a Security Policy File on the Server

Using a text editor, type the following text into a new file (replace *myserver* with the name of your server).

```
grant codeBase "http://myserver/EDM/-" {  
    permission java.security.AllPermission;  
};
```

Save the file as *system.policy* in the web application folder on your web server (the default folder is *C:\EDMServer\webapps\EDM*).

Updating the Java Runtime Environment for Each User

For each user that wants to use the new security policy file, take the following steps:

1. Use a text editor (such as WordPad) to edit the *java_folder\lib\security\java.security* file. By default, the java folder is *C:\Program Files\Java\jre7* on each user's PC.
2. Find the line that reads: *policy.url.2=file:\${user.home}/.java.policy*
3. Add a line after it that reads: (replacing *myserver* with the name of your server).
4. Close and save the file.

Updating the Java Policy File

To enable the clipboard in the Java Policy file, take the following steps:

1. Using a text editor (such as WordPad), open the "*java_folder*"\lib/security/java.policy file. By default, the "*java_folder*" is in the C:\Program Files\Java\jre7 directory.
2. Find the line that reads: // "standard" properties that can be read by anyone
3. Add the following line after it: *permission java.awt.AWTPermission "accessClipboard";*
4. Close and save the file.
5. Copy this file to the user home directory as *.java.policy*. By default, user folders are in the C:\Users directory. The new file will start with a "."
6. Clear the Java cache and reload the browser.

The EDM default folder for versions prior to 6.7 may be *C:/apache-tomcat-6.0.18/webapps/EDM* or similar.

Using Secure Socket Layer (SSL) to Protect Data

To protect your data from hackers, you can optionally configure your system to use SSL to encrypt all communications between browsers and the web server. Because this configuration varies based on your web server, you should refer to your web server documentation for details. If you are using Tomcat only as a JSP container, while using Apache or IIS as the primary web server, you should configure the primary web server for SSL communication.

Configuring a Tomcat Web Server

1. From a command line, use the following command to generate a keystore. Remember the keystore password.

```
%JAVA_HOME%\bin\keytool -genkey -alias tomcat -keyalg RSA -keystore /my/keystore/path/.keystore
```

When prompted for the *Common Name*, type the server name (e.g. www.mycompany.com) that you use when you browse to the web page. If you are using the *transfer_web* function, type this same server name in the *Transfer Host* field in *Edit Locations and Groups* (available from the *Sites* module).

2. Uncomment and update the SSL Connector configuration in *CATALINA_HOME/conf/server.xml*.

```
<!-- Define a SSL Coyote HTTP/1.1 Connector on port 8443 -->
<Connector port="8443"
    minSpareThreads="25" maxSpareThreads="75"
    enableLookups="false" disableUploadTimeout="true"
    acceptCount="100" debug="0" scheme="https" secure="true"
    clientAuth="false" sslProtocol="TLS"

    keystorePass="myKeystorePassword"
    keystoreFile="/my/keystore/path/.keystore"
/>
```

3. Uncomment the following security constraint in *CATALINA_HOME\webapps\EDM\WEB-INF\web.xml*.

```

<!-- Ensure SSL used for all pages -->
<security-constraint>
  <web-resource-collection>
    <web-resource-name>main</web-resource-name>
    <url-pattern>/main</url-pattern>
    <url-pattern>/main/*</url-pattern>
    <url-pattern>*.do</url-pattern>
    <url-pattern>/login.jsp</url-pattern>
    <http-method>GET</http-method>
    <http-method>POST</http-method>
  </web-resource-collection>
  <user-data-constraint>
    <transport-guarantee>CONFIDENTIAL</transport-guarantee>
  </user-data-constraint>
</security-constraint>

```

4. Restart the Apache Tomcat service.
5. Access the system as usual from a browser and accept the certificates when prompted. Notice that the protocol on the address line is *https*: instead of *http*:

Sample Script

The following batch file can be used as a reference for all the steps involved in setting up SSL for both EDM running under Tomcat and for ActiveMQ used to transfer EDM transactions.

```

set JAVA_HOME=C:\Program Files (x86)\Java\jre7
set KEYSTORE=C:/.keystore
set CERTIFICATE=C:/server.crt
set P12KEYSTORE=C:/.p12keystore
set PEM=C:/server.pem
set OPENSSSL=C:\Me\EDM\tnative\win64\openssl.exe

:=====
:
: This batch file sets up a certificate allowing EDM to use SSL to connect
to the
: Tomcat web server using HTTPS://myserver:8443/EDM (server and port may
differ).
:
: Edit the keystore parameters to point to your actual Java folders.
:
: When the keytool prompts you to enter the information for the
certificate,
: BE SURE COMMON NAME IS EXACT SERVER NAME (like chris1 or
www.xpient.com)!!!
: The other values don't matter and can be left unknown.
:=====
:
: Delete tomcat certificate from all Java keystores
"%JAVA_HOME%\bin\keytool" -delete -alias tomcat -keypass changeit
-storepass changeit -keystore "%JAVA_HOME%\lib\security\cacerts"
"%JAVA_HOME%\bin\keytool" -delete -alias tomcat -keypass changeit
-storepass changeit -keystore "C:\Program Files

```



```

(x86)\Java\jre6\lib\security\cacerts"
"%JAVA_HOME%\bin\keytool" -delete -alias tomcat -keypass changeit
-storepass changeit -keystore "C:\Program Files
(x86)\Java\jdk1.6.0_45\jre\lib\security\cacerts"
"%JAVA_HOME%\bin\keytool" -delete -alias tomcat -keypass changeit
-storepass changeit -keystore "C:\Program Files
(x86)\Java\jdk1.6.0_07\jre\lib\security\cacerts"
"%JAVA_HOME%\bin\keytool" -delete -alias tomcat -keypass changeit
-storepass changeit -keystore "C:\Program
Files\Java\jdk1.6.0_31\jre\lib\security\cacerts"

: Delete the existing keystore and certificate if they exist
if exist "%KEYSTORE%" del "%KEYSTORE%"
if exist "%CERTIFICATE%" del "%CERTIFICATE%"

: Generate new certificate - BE SURE COMMON NAME (CN) IS EXACT SERVER NAME
(like chris1 or www.xpient.com) when asked for First and Last Name!!!
"%JAVA_HOME%\bin\keytool" -genkey -alias tomcat -keypass changeit
-storepass changeit -keyalg RSA -keystore "%KEYSTORE%"

: Export certificate
"%JAVA_HOME%\bin\keytool" -export -alias tomcat -keypass changeit
-storepass changeit -file "%CERTIFICATE%" -keystore "%KEYSTORE%"

: Import certificate into other Java keystores (not needed for sites with
one JRE)
"%JAVA_HOME%\bin\keytool" -import -alias tomcat -keypass changeit
-storepass changeit -file "%CERTIFICATE%" -keystore
"%JAVA_HOME%\lib\security\cacerts"
"%JAVA_HOME%\bin\keytool" -import -alias tomcat -keypass changeit
-storepass changeit -file "%CERTIFICATE%" -keystore "C:\Program Files
(x86)\Java\jre6\lib\security\cacerts"
"%JAVA_HOME%\bin\keytool" -import -alias tomcat -keypass changeit
-storepass changeit -file "%CERTIFICATE%" -keystore "C:\Program Files
(x86)\Java\jdk1.6.0_45\jre\lib\security\cacerts"
"%JAVA_HOME%\bin\keytool" -import -alias tomcat -keypass changeit
-storepass changeit -file "%CERTIFICATE%" -keystore "C:\Program Files
(x86)\Java\jdk1.6.0_07\jre\lib\security\cacerts"
"%JAVA_HOME%\bin\keytool" -import -alias tomcat -keypass changeit
-storepass changeit -file "%CERTIFICATE%" -keystore "C:\Program
Files\Java\jdk1.6.0_31\jre\lib\security\cacerts"

: Convert keystore into PKCS#12 keystore:
if exist "%P12KEYSTORE%" del "%P12KEYSTORE%"
"%JAVA_HOME%\bin\keytool" -importkeystore -srckeystore "%KEYSTORE%"
-srcstorepass changeit -destkeystore "%P12KEYSTORE%" -deststorepass
changeit -srcstoretype jks -deststoretype pkcs12

: Convert PKCS#12 keystore into PEM file:
if exist "%PEM%" del "%PEM%"
"%OPENSSSL%" pkcs12 -in "%P12KEYSTORE%" -out "%PEM%"

```

```

:=====
=====
: To configure Tomcat for SSL, set up a Connector in conf/server.xml as
shown below:
:
: If you are NOT using the APR protocol set up the connector like this:
:
: <!-- Define a SSL Connector on port 8443 -->
: <Connector port="8443"
: protocol="org.apache.coyote.http11.Http11Protocol"
: maxThreads="200"
: scheme="https" secure="true" SSLEnabled="true"
: keystoreFile="C:/.keystore" keystorePass="changeit"
: clientAuth="false" sslProtocol="TLS"/>
:
:
: If you are using the APR protocol set up the connector like this:
:
: <!-- Define a SSL APR Connector on port 8443 -->
: <Connector port="8443"
: maxThreads="200"
: scheme="https"
: secure="true"
: SSLEnabled="true"
: SSLCertificateFile="C:/server.crt"
: SSLCertificateKeyFile="C:/server.pem"
: clientAuth="false"
: SSLProtocol="TLSv1"/>
:
: For details see: http://tomcat.apache.org/tomcat-6.0-doc/ssl-howto.html
:
:=====
=====

:=====
=====
: To configure ActiveMQ to use SSL, set the SSL_OPTS environment variable
before
: starting ActiveMQ like this:
:
: set SSL_OPTS=-Djavax.net.ssl.keyStore=C:/.keystore
-Djavax.net.ssl.keyStorePassword=changeit
:
: Also, change the transport connector in the conf/activemq.xml file to be
like
: the one shown below and replace 'neimeisterc' with the real server name.
:
: <transportConnector name="ssl"
uri="ssl://neimeisterc:443?maximumConnections=1000&wireformat.maxFrame
Size=104857600"/>
:
: For details see: http://activemq.apache.org/ssl-transport-reference.html
: and: http://activemq.apache.org/how-do-i-use-ssl.html

```

```
:
:=====
=====

:=====
=====
: To configure EDM to use ActiveMQ with SSL, set an 'onStartup' expression
in
: config.xml like the following and replace 'neimeisterc' with the real
server name.
:
:  onStartup = "addProcessReceivedTransactionsListener() +
addJMSCommitListener( '*', 'com.xpiend.EDM.customerid.outgoing',
'ssl://neimeisterc:443', 'system', 'manager' ) + addJMSReceiveListener(
'com.xpiend.EDM.customerid.incoming', 'ssl://neimeisterc:443', 'system',
'manager' )"
:=====
=====
```

pause

Increasing Central Site Memory

The following describes how to change the amount of memory allocated to Java and EDM on a central system that is running with the Apache Tomcat web server.

1. Go to the server where Apache Tomcat is running.
2. Select *Start>All Programs>Apache Tomcat>Configure Tomcat* to open the Tomcat configuration screen. If this menu selection is not available, take the following steps.
 - Open a command prompt.
 - Find the path to *tomcat8w.exe* in your *EDMServer* directory (the executable name will vary depending upon the version of Tomcat you are using).
 - Call *tomcat8w.exe* with the flag *//ES/EDMServer* (for example, *C:\EDMServer\bin\tomcat6w.exe //ES/EDMServer*)
 - Press *Enter* to run the command and open the Tomcat configuration screen.
3. From the Tomcat configuration screen, click the *Java* tab.
4. Set the *Initial memory pool size* to the desired number of megabytes. The default is 64.
5. Set the *Maximum memory pool size* to the desired number of megabytes. The default is 64.
6. Click *OK*.

Increasing Remote Site Memory

This page describes how to increase the amount of memory that is available to the system at a remote site. This is accomplished by editing the parameters in the *go.bat* file that are used when starting Java from a command-line.

1. Right-click *go.bat* in the installation folder (*C:\EDMWeb* by default), and select *Edit* from the menu that appears.
2. Insert the following parameters immediately before *-Duser* (change the number of megabytes as desired): *-Xms64m -Xmx256m*
3. Close and save the *go.bat* file.

Sample go.bat File

The following *go.bat* runs the system in the *C:\EDMWeb* folder with 64mb of initial memory and 256mb of maximum memory use.

```
@ "%JAVA_HOME%\bin\java.exe" -Xms64m -Xmx256m "-Duser.dir=C:\EDMWeb" -jar
C:\EDMWeb\lib\system.jar "-user=remote" "-pwd=remote" %1 %2 %3 %4 %5
```

Uninstalling the Central System

The following describes how to uninstall the central system.

1. Use *SQL Server Enterprise Manager* to drop the *EDMWebHQ* database.
2. Go to *Start>Control Panel>Add/Remove Programs* and uninstall *Apache Tomcat*. When prompted to delete all files, answer *Yes*.
3. If you are not using Java for other purposes on the system, uninstall the *Java Runtime Environment*.

Uninstalling a Remote System

The following describes how to uninstall a remote system.

1. Use *SQL Server Enterprise Manager* to drop the *EDMWeb* database.
2. Delete the *C:\EDMWeb* folder and all subfolders.
3. If you are not using Java for other purposes on the system, go to *Start>Control Panel>Add/Remove Programs* and uninstall the *Java Runtime Environment*.

Hiding the Applet Warning Icon

When users access the system through a browser that is using a Java 6 plugin, they may see a yellow warning icon pop up next to the EDM windows. To stop this behavior for the EDM applet (and all other applets), users can modify the *java.policy* file which is found in the security folder. The path is something like *%JRE_HOME%\lib\security\java.policy*.

Add the following line to the *java.policy* file, and then restart your browser:

```
permission java.awt.AWTPermission "showWindowWithoutWarningBanner";
```

EDM Upgrade Installation

For data security, you should back up your system database and the files in the installation directory and its subdirectories if you are updating an existing installation.

IMPORTANT NOTE

If you are upgrading EDM from Apache 6/7 using an SSL connection, you must make the following change to the server.xml file located in the \EDMServer\conf folder. The headers in the Connector protocol section that include the certificate information must read as follows:

```
keystoreType="",keystoreFile="", and keystorePass=""
```

The EDM service will not start until after you make this change.

- [Upgrading from EDM 6.0 or Later](#)
- [Upgrading from EDM 3.1 or Later](#)
- [Upgrading from EDM 3.0.x](#)
- [Upgrading from EDM 2.9 or earlier](#)

Upgrading from EDM 6.0 or Later

Upgrading a Central System

To upgrade a central EDM 6.0 or later installation, take the following steps:

Get the installation files

1. Create an install directory, such as C:\EDM_INSTALL.
2. Download the EDM CD Image from <https://ftp.xpient.com>.
3. Extract the archive file to the install directory.

Back up your existing system

1. Stop the Apache Tomcat service
2. Create a backup directory, such as C:\EDM_BACKUP.
3. Backup your EDM folder on the central server (usually C:\apache-tomcat-6.0.18\webapps\EDM).
4. Backup all the central EDM databases referenced in your central config.xml file (usually called EDMWebHQ).

Install Updated EDM Files

1. Copy system.jar from the EDM_INSTALL\data\lib folder to C:\apache-tomcat-6.0.18\webapps\EDM\web-inf\lib
2. Copy mail.jar from the EDM_INSTALL\data\lib folder to C:\apache-tomcat-6.0.18\webapps\EDM\web-inf\lib
3. Copy EDMclient.jar from the EDM_INSTALL\data\lib folder to C:\apache-tomcat-6.0.18\webapps\EDM
4. Copy jcalendar.jar from the EDM_INSTALL\data\lib folder to C:\apache-tomcat-6.0.18\webapps\EDM
5. Copy loginPage.jsp from the EDM_INSTALL\data\config\common folder to C:\apache-tomcat-6.0.18\webapps\EDM
6. Copy mail.properties from the EDM_INSTALL\data\config\common folder to C:\apache-tomcat-6.0.18\webapps\EDM\web-inf\classes
7. Restart the Apache Tomcat service from the Start menu by using Control Panel - Administrative Tools - Services.
8. Test the installation by opening your browser to: <http://localhost:8080/EDM> (if you are not on the same machine as the server, replace localhost with the server name).

Upgrading a Remote System

To update a remote site running EDM 6.0 (or later) take the following steps:

1. Determine your installation directory (dir). By default, the installation folder is C:\EDMWeb.

2. Backup dir\lib\system.jar to a different folder (don't just rename it or it will still be used).
3. Copy the new system.jar to dir\lib.

Upgrading from EDM 3.1 or Later

Upgrading a Central System

To upgrade a central EDM 3.1 or later installation, take the following steps:

Get the installation files

1. Create an install directory, such as C:\EDM_INSTALL.
2. Download the EDM 6.0 CD Image from <https://ftp.xpient.com>.
3. Extract the archive file to the install directory.

Back up your existing system

1. Stop the Apache Tomcat service
2. Create a backup directory, such as C:\EDM_BACKUP.
3. Backup your EDM folder on the central server (usually C:\jakarta-tomcat-5.5.7\webapps\EDM).
4. Backup all the central EDM databases referenced in your central config.xml file (usually called EDMWebHQ).
5. If you have modified the Tomcat users file to provide access to the Tomcat Manager or for other reasons, copy the jakarta-tomcat-5.5.7\conf\Tomcat-users.xml file to the backup directory.

Uninstall Tomcat 5.5.7

1. Uninstall Apache via the Uninstall.exe in jakarta-tomcat-5.5.7 or Add/Remove Programs in the Control Panel.
2. When prompted to delete the contents of jakarta-tomcat-5.5.7, answer No.

Install EDM

1. From the install directory, run the batch file that most closely matches your application version. For example, if you are running IRIS 3.5.7 or later with LaborPro, run setup_central_iris_3.5.7_laborpro_2.6.2.bat.
2. Stop the Apache Tomcat service.
3. Copy the files in the EDM_BACKUP\WEB-INF\Classes folder EXCEPT APP_?.XML and MENU_?.XML to C:\apache-tomcat-6.0.18\webapps\EDM\WEB-INF\classes
4. Copy and overwrite the following folders and their contents (if applicable) from the EDM_BACKUP directory to the C:\apache-tomcat-6.0.18\webapps\EDM folder:
 - EDM_BACKUP\IRIS
 - EDM_BACKUP\Incoming
 - EDM_BACKUP\Outgoing
5. Copy the following files (if applicable) from the EDM_BACKUP directory to the specified directories:
 - EDM_BACKUP\System.policy to C:\apache-tomcat-6.0.18\webapps\EDM\
 - EDM_BACKUP\Tomcat-Users.xml to C:\apache-tomcat-6.0.18\conf\Tomcat-Users.xml
6. Edit the C:\apache-tomcat-6.0.18\webapps\EDM\WEB-INF\classes\config.xml and replace C:\jakarta-tomcat-5.5.7 with C:\apache-tomcat-6.0.18
7. Restart the Apache Tomcat service from the Start menu by using Control Panel - Administrative Tools - Services.
8. Test the installation by opening your browser to: <http://localhost:8080/EDM> (if you are not on the same machine as the server, replace localhost with the server name).
9. Upon successful load, you will be prompted with an "Invalid License" error. You will need to provide the error information to your XPIENT representative to have a new license generated.

Upgrading a Remote System

To update a remote EDM 3.1 or later installation, take the following steps:

1. Determine your installation directory (dir). By default, the installation folder is C:\EDMWeb.
2. Backup dir\lib\system.jar to a different folder (don't just rename it or it will still be used).
3. Copy the new system.jar to dir\lib.
4. Install Java 1.6.0_07 by extracting the jre1.6.0_07.zip file included in the EDM CD Image into the installation directory.
5. Modify go.bat to change the path to Java.exe to C:\EDMWeb\jre1.6.0_07.

Upgrading from EDM 3.0.x

If EDM 3.0.x is already installed, take the following steps to upgrade to EDM 3.1 or later:

1. Copy the updated system.jar over the existing one in the installation folder.
2. Copy the updated EDMclient.jar over the existing one in the installation folder.
3. Copy the updated app_appname.xml over the existing one in the installation folder.
4. Copy the appropriate conversion file (convert_appname_version.xml to the installation folder containing config.xml).

Upgrading from EDM 2.9 or earlier

If EDM 2.9.x or earlier is already installed, contact your vendor for installation instructions.

EDM Silent Installation Instructions

Overview

EDM is capable of being installed quickly and silently using the "1 Click Installation". An Install_Customization.ini file can be included in the same directory as the EDM install to allow customization. Alternatively, the EDM installation packaged inside an executable RAR file (alongside the Install_Customization.ini) can be provided to allow the installation to be customized and executed silently.

- [Customization Instructions](#)
- [About Security Authentication](#)
- [Server Customization](#)
 - [Required Settings](#)
 - [POS Type Versions](#)
 - [Optional Values](#)
 - [Install_Customization.ini Setup Example](#)
- [Remote Customization](#)
 - [Required Settings](#)
 - [POS Type Versions](#)
 - [Optional Values](#)
 - [Install_Customization.ini Setup Example](#)
- [Installation](#)
 - [Server Installation](#)
 - [Remote Installation](#)
- [Common Issue Troubleshooting](#)

Customization Instructions

- The customization is done by editing the setting values in the Install_Customization.ini. If you intend to simply place the file in the same directory as the install, then it can be edited with any text editor.
- If you have the executable RAR, you will first have to open the archive. If you do not have a program to open RAR files, then WinRAR can be used.
- You want to open the archive (not extract) and edit the file inside the archive. Select the Install_Customization.ini inside the archive and open it using any text editor.
- Once you have finished editing the file, either save it and place it in the folder with the install or add it to the archive and overwrite the old file.

About Security Authentication

Authentication is done from a separate database. All users are migrated from the company database into a new authentication database named "auth" by default.

This database contains the mapping of the users to the companies that they have access to, and is maintained by the upgraded User editor. This means that only 1 user record exists, and that a user with access to multiple companies no longer needs to change their password in each separate company. When the authentication database is initially created, if there is a user record in 2 or more companies with different passwords, only the first user record is created. If you have multiple companies in your EDM server, check the log file after initial startup to see if there were any users with this scenario.

RESTful API - There is an API that can be used to authenticate with EDM. You can read more about it [here](#).

Server Customization

Required Settings

This section specifies the required setting values in the Install_Customization.ini file for a Server installation.

Setting Name	Description	Available Options	Examples
installDir	The installation directory	The directory must be EDMServer but this can be on the root of any drive.	<ul style="list-style-type: none">C:\EDMServerD:\EDMServer
companyName	The company name property	Any text string will work	HQ
companyId	The company ID property	Any text string will work	IRISHQ
installActiveMQ	Whether or not to install ActiveMQ Message Queuing	Y or N indicating Yes or No	N
thisSiteName	The site's name	Any text string that indicates this is the Server	HQ
thisSiteNumber	The site's number	Server installations should be site 0	0
webServerConnectorPort	The web server connection port.	Any server port (use default 8080 if unsure)	80
isCentralSite	Install central instance of the server deployment.	Y (must be Y)	Y
posType	POS Type being supported	Please see <i>POS Type Versions</i> below for a list of POS Types.	IRIS
posVersion	The version of the specified posType	Please see <i>POS Type Versions</i> below for a list of supported versions	4.0
systemDataSourceName	The EDM system data source name	Any text string will work	EDMWeb
systemDataSourceServer	The EDM system data source server	Any text string will work, you can specify an instance	localhostXSIRIS

systemDataSourceDatabase	The EDM system data source database name	Any text string will work	EDMWeb
systemDataSourceUser	The EDM system data source database EDM user	Any text string will work; leave this setting value blank when using integrated security	EDMUser
systemDataSourceEncryptedPassword	The EDM system data source database EDM user's encrypted password	This value will most likely be blank. EDM encrypts the supplied password on startup.	
systemDataSourceConnectionString	The EDM system data source database connection string for integrated security	Any text string will work, can be left blank	Server=myServerAddress;Database=myDataBase;Trusted_Connection=True;
systemDataSourceConnectionType	The EDM system data source database connection type	Any text string will work	SQLSERVER_JDBC
authenticationDataSourceName	The EDM authentication data source name	Any text string will work	authenticationDataSource
authenticationDataSourceServer	The EDM authentication data source server	Any text string will work	localhost
authenticationDataSourceVersion	The version of the EDM authentication data source	Any text string will work	1
authenticationDataSourceDatabase	The EDM authentication data source database name	Any text string will work	Auth
authenticationDataSourceUser	The EDM authentication data source database EDM user	Any text string will work; leave this setting value blank when using integrated security	sa
authenticationDataSourcePassword	The EDM authentication data source database EDM user's password	Any text string will work; leave this setting value blank when using integrated security	thesapassword

authenticationDataSourceEncryptedPassword	The EDM authentication data source database EDM user's encrypted password	This value will most likely be blank. EDM encrypts the supplied password on startup.	
authenticationDataSourceConnectionString	The EDM authentication data source database connection string for integrated security	Any text string will work, can be left blank	
authenticationDataSourceConnectionType	The EDM authentication data source database connection type	Any text string will work	SQLSERVER_JDBC

POS Type Versions

The following table provides the valid values for the *posType* and *posVersion* settings in the Install_Customization.ini file.

posType	posVersion
Aloha	1 6
Elstar	3.1.1 4.0
IRIS	3.7.4_PR10 3.7.4_PR10_laborpro_2.6.2 3.7.9 4.0
Micros	4.8
Nucleus	1
Panasonic	7500mwsqs4
POSitouch	5.3.4
Smart	6.21
XPRESS	4.5.0

Optional Values

This section specifies the optional setting values in the Install_Customization.ini file for a Server installation.

Setting Name	Description	Available Options	Examples
onStartup	The command that is triggered by EDM when it starts	Any text string will work	"addProcessReceivedTransactionsListener() + addJMSSCommitListener('*', 'com.xpient.EDM.customerid.outgoing', 'tcp://localhost:61616', 'system', 'manager')"
browserPageTitle	The title that will be displayed in the browser	Any text string will work	EDM Central Site\EDM Remote Site
systemDataSourceVersion	The system data source database version, defaults to 1. Can be used to track changes made to the database.	Any number will work	1

Install_Customization.ini Setup Example

The following is an example of the Install_Customization.ini configuration for a Server Installation.

```
[ INFO ]
webServerConnectorPort=8080
isCentralSite=Y
installActiveMQ=N
installDir=C:\EDMServer
companyName=HQ
companyId=HQ
browserPageTitle=
thisSiteName=HQ
thisSiteNumber=0
posType=iris
posVersion=3.7.4_PR10_laborpro_2.6.2
onStartup=

systemDataSourceName=EDMWebHQ
systemDataSourceServer=localhost
systemDataSourceVersion=1
systemDataSourceDatabase=EDMWebHQ
systemDataSourceUser=sa
systemDataSourcePassword=thesapassword
systemDataSourceEncryptedPassword=
systemDataSourceConnectionString=
systemDataSourceConnectionType=SQLSERVER_JDBC

authenticationDataSourceName=authenticationDataSource
authenticationDataSourceServer=localhost
authenticationDataSourceVersion=1
authenticationDataSourceDatabase=Auth
authenticationDataSourceUser=sa
authenticationDataSourcePassword=thesapassword
authenticationDataSourceEncryptedPassword=
authenticationDataSourceConnectionString=
authenticationDataSourceConnectionType=SQLSERVER_JDBC
```

Remote Customization

Required Settings

This section specifies the required setting values in the Install_Customization.ini file for a Remote installation.

Setting Name	Description	Available Options	Examples
--------------	-------------	-------------------	----------

installDir	The installation directory	The directory must be EDMServer but this can be on the root of any drive.	<ul style="list-style-type: none"> • C:\EDMWeb • D:\EDMWeb
companyName	The company name property	Any text string will work	EDM
companyId	The company ID property	Any text string will work	EDMWeb
installActiveMQ	Whether or not to install ActiveMQ Message Queuing	Y or N indicating Yes or No	N
posType	POS Type being supported	Please see <i>POS Type Versions</i> below for a list of POS Types.	IRIS
posVersion	The version of the specified posType	Please see <i>POS Type Versions</i> below for a list of supported versions	4.0
systemDataSourceName	The EDM system data source name	Any text string will work	EDMWeb
systemDataSourceServer	The EDM system data source server	Any text string will work, you can specify an instance	localhost\XSIRIS
systemDataSourceDatabase	The EDM system data source database name	Any text string will work	EDMWeb
systemDataSourceUser	The EDM system data source database EDM user	Any text string will work; Can be left blank if using integrated security	EDMUser
systemDataSourcePassword	The EDM system data source database EDM user password	Any text string will work; Can be left blank if using integrated security	Pa\$\$w0rd!
systemDataSourceEncryptedPassword	The EDM system data source database EDM user's encrypted password	This value should be left blank. EDM encrypts the supplied password on startup.	Leave blank

systemDataSourceConnectionString	The EDM system data source database connection string for integrated security.	Any text string will work, can be left blank	Server=myServerAddress;Database=myDataBase;Trusted_Connection=Tru
systemDataSourceConnectionType	The EDM system data source database connection type.	Any text string will work	SQLSERVER_JDBC
dataSource1Name	The data source #1 name	Any text string will work	IRIS
dataSource1Version	The data source #1 database version	Any text string will work	4.0
dataSource1Server	The data source #1 server	Any text string will work, you can specify an instance	localhost\XSIRIS
dataSource1Database	The data source #1 name	Any text string will work	EDMWeb
dataSource1User	The data source #1 database EDM user	Any text string will work; Can be left blank if using integrated security	EDMUser
dataSource1Password	The data source #1 database EDM user password	Any text string will work; Can be left blank if using integrated security	Pa\$\$w0rd!
dataSource1EncryptedPassword	The data source #1 database EDM user's encrypted password	This value will most likely be blank. EDM encrypts the supplied password on startup.	leave blank
dataSource1DatabaseConnectionString	The data source #1 database connection string for integrated security.	Any text string will work, can be left blank	Server=myServerAddress;Database=myDataBase;Trusted_Connection=Tru
dataSource1ConnectionType	The data source #1 database connection type.	Any text string will work	SQLSERVER_JDBC
authenticationDataSourceName	The EDM authentication data source name	Any text string will work	authenticationDataSource
authenticationDataSourceServer	The EDM authentication data source server	Any text string will work	localhost
authenticationDataSourceVersion	The version of the EDM authentication data source	Any text string will work	1

authenticationDataSourceDatabase	The EDM authentication data source database name	Any text string will work	Auth
authenticationDataSourceUser	The EDM authentication data source database EDM user	Any text string will work; leave this setting value blank when using integrated security	sa
authenticationDataSourcePassword	The EDM authentication data source database EDM user's password	Any text string will work; leave this setting value blank when using integrated security	thesapassword
authenticationDataSourceEncryptedPassword	The EDM authentication data source database EDM user's encrypted password	This value will most likely be blank. EDM encrypts the supplied password on startup.	
authenticationDataSourceConnectionString	The EDM authentication data source database connection string for integrated security	Any text string will work, can be left blank	
authenticationDataSourceConnectionType	The EDM authentication data source database connection type	Any text string will work	SQLSERVER_JDBC

POS Type Versions

The following table provides the valid values for the *posType* and *posVersion* settings in the Install_Customization.ini file.

posType	posVersion
Aloha	1.6
Elstar	3.1.1 4.0
IRIS	3.7.4_PR10 3.7.4_PR10_laborpro_2.6.2 3.7.9 4.0
Micros	4.8
Nucleus	1
Panasonic	7500mwsqs4
POSitouch	5.3.4
Smart	6.21
XPRESS	4.5.0

Optional Values

This section specifies the optional setting values in the Install_Customization.ini file for a Remote installation.

Setting Name	Description	Available Options	Examples
onStartup	The command that is triggered by EDM when it starts	Any text string will work	"addProcessReceivedTransactionsListener() + addJMSCommitListener('*', 'com.xpient.EDM.customerid.outgoing', 'tcp://localhost:61616', 'system', 'manager')"
browserPageTitle	The title that will be displayed in the browser	Any text string will work	EDM Central Site\EDM Remote Site
systemDataSourceVersion	The system data source database version, defaults to 1. Can be used to track changes made to the database.	Any number will work	1
dataSource#Name	You can add up to 4 total data sources. Indicate the subsequent data sources using the same values names as data source but number them accordingly.	See example	dataSource2Name: The data source #2 name. dataSource3Name: The data source #3 name. dataSource4Name: The data source #4 name.

Install_Customization.ini Setup Example

The following is an example of the Install_Customization.ini configuration for a Remote Installation.

```
[INFO]
installActiveMQ=N
installDir=C:\EDMWeb
companyName=HQ
companyId=HQ
browserPageTitle=
posType=iris
posVersion=3.7.4_PR10_laborpro_2.6.2
onStartup=

systemDataSourceName=EdmWeb
systemDataSourceServer=localhost
systemDataSourceDatabase=EdmWeb
systemDataSourceUser=sa
systemDataSourcePassword=thesapassword
systemDataSourceEncryptedPassword=
systemDataSourceConnectionString=
systemDataSourceConnectionType=SQLSERVER_JDBC

dataSourceName=IRIS
dataSourceServer=localhost
dataSourceVersion=1
dataSourceDatabase=IRIS
dataSourceUser=sa
dataSourcePassword=thesapassword
dataSourceEncryptedPassword=
dataSourceDatabaseConnectionString=
dataSourceConnectionType=SQLSERVER_JDBC

authenticationDataSourceName=authenticationDataSource
authenticationDataSourceServer=localhost
authenticationDataSourceVersion=1
authenticationDataSourceDatabase=Auth
authenticationDataSourceUser=sa
authenticationDataSourcePassword=thesapassword
authenticationDataSourceEncryptedPassword=
authenticationDataSourceConnectionString=
authenticationDataSourceConnectionType=SQLSERVER_JDBC
```

Installation

Server Installation

To install the Server system, take the following steps:

1. Launch the install from either the installation executable in the same folder as the install_customization.ini(step a) or from the executable RAR archive(step b)
 - a. To launch the silent install when the install_customization.ini and the installation executable are in the same folder use the command prompt. Navigate to the installation executable and launch the file using the /verysilent flag. It should look like:
"EDM-Server-Setup.exe /verysilent"
 - b. After adding the updated install_customization.ini to the archive double click the archive to run the installation
2. Allow time for the installation to complete, approximately 3-5 minutes, then start the Apache Tomcat service. You can find this in the services under administrator tools.

3. Test your installation by opening your browser to: <http://localhost:8080/EDM> (if you are not on the same machine as the server, replace **localhost** with the server name or IP address, use the same port as specified in the installation). Login using userid **admin** and password **admin**. At installation, the system has the following predefined user id/password combinations:
 - a. **remote/remote** - Typical login for a user at a remote site who only has permission to process transactions.
 - b. **central/central** - Typical login for a central user who has permission to use all the forms and functions at headquarters.
 - c. **admin/admin** - Administrative login with all permissions including setting up new users and roles and modifying the forms and queries in the application. If the application loads and the logins work then EDM has installed successfully
4. If the application does not load refer to the log file at \EDMServer\Webapps\EDM\log.txt

Remote Installation

To install the Remote system, take the following steps:

1. Launch the install from either the installation executable in the same folder as the install_customization.ini(step a) or from the executable RAR archive(step b)
 - a. To launch the silent install when the install_customization.ini and the installation executable are in the same folder use the command prompt. Navigate to the installation executable and launch the file using the /verysilent flag. It should look like:
"EDM-Remote-Setup.exe /verysilent"
 - b. After adding the updated install_customization.ini to the archive double click the archive to run the installation
2. Allow time for the installation to complete, approximately 3-5 minutes, then run go.bat. This file will be located in the EDMWeb folder. If the application window launches then EDM has successfully installed.
3. If the application does not launch refer to the log at \EDMWeb\log.txt

Common Issue Troubleshooting

IMPORTANT NOTE

If you are upgrading EDM from Apache 6/7 using an SSL connection, you must make the following change to the server.xml file located in the \EDMServer\conf folder. The headers in the Connector protocol section that include the certificate information must read as follows:

```
keystoreType="",keystoreFile="", and keystorePass=""
```

The EDM service will not start until after you make this change.

- Make sure the user specified for the installs has permission to create databases
- Confirm the password entered for the users was correct. This can be corrected in the [config.xml](#) file.
 - Remote Config.xml is located at: \EDMWeb\Config.xml
 - Server Config.xml is located at: \EDMServer\webapps\EDM\WEB-INF\classes\Config.xml
- Ensure the correct database is specified in the connect string for integrated security
- If you see Java errors when loading the EDM Central webpage, ensure the Java version is at least 1.8 on the machine from which you are browsing to the website.

EDM Interactive Installation Instructions

This page provides instructions for installing EDM using the interactive installation wizard.

Overview

The following describes the prompt-based interactive installation process for EDM and defines the values expected for each prompt. The Server and Remote installations are both described.

- [Server Installation](#)
 - [Step 1 - Execute the Installation File](#)
 - [Step 2 - Service or Stand Alone](#)
 - [Step 3 - EDM Version](#)
 - [Step 4 - EDM Information](#)
 - [Step 5 - System Data Source Information](#)
 - [Step 6 - Authentication data source information](#)
 - [Step 7 - wpauthenticationDataSource](#)
 - [Step 8 - Ready to Install](#)
 - [Step 9 - Starting the Apache Tomcat Service](#)
 - [Step 10 - Testing the Server](#)
- [Remote Installation](#)
 - [Step 1 - Execute the Installation File](#)
 - [Step 2 - EDM Version](#)
 - [Step 3 - EDM Information](#)

- Step 4 - System Data Source Information
- Step 5 - Data Source 1 Information
- Step 6 - Authentication data source information
- Step 7 - ActiveMQ
- Step 8 - Ready to Install
- Step 9 - Testing the Installation
- Common Issue Troubleshooting

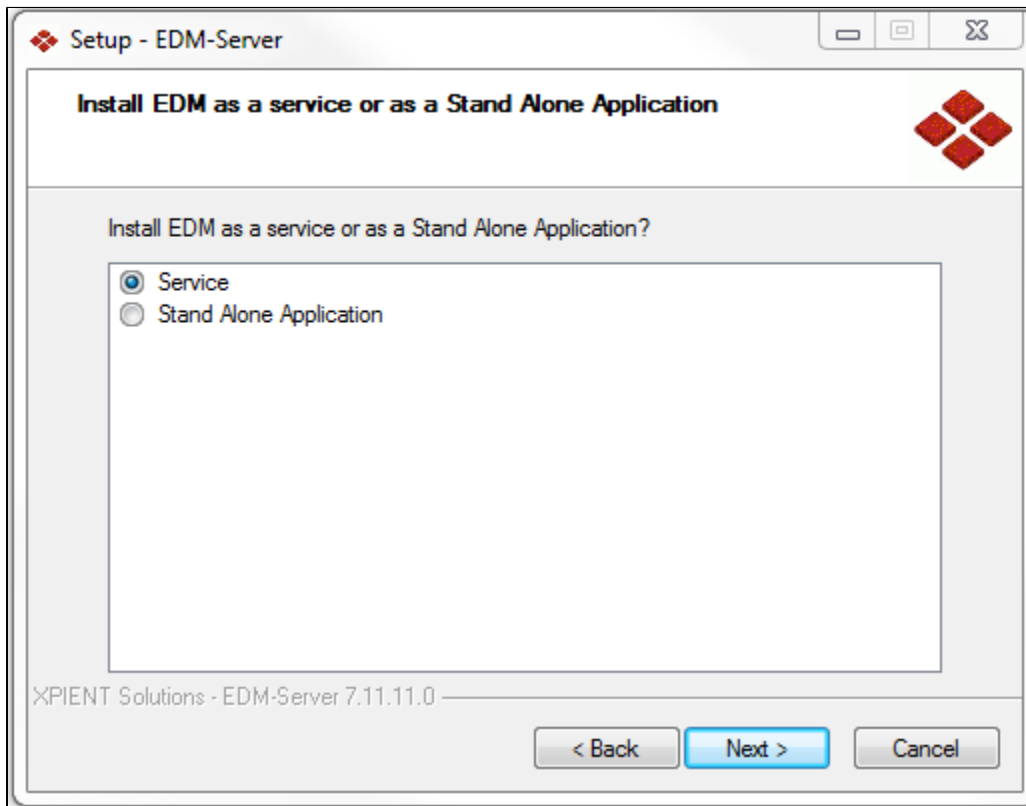
Server Installation

Step 1 - Execute the Installation File

If *Install_Customization.ini* exists in the same directory as *EDM-Server-Setup.exe*, then rename *Install_Customization.ini*.

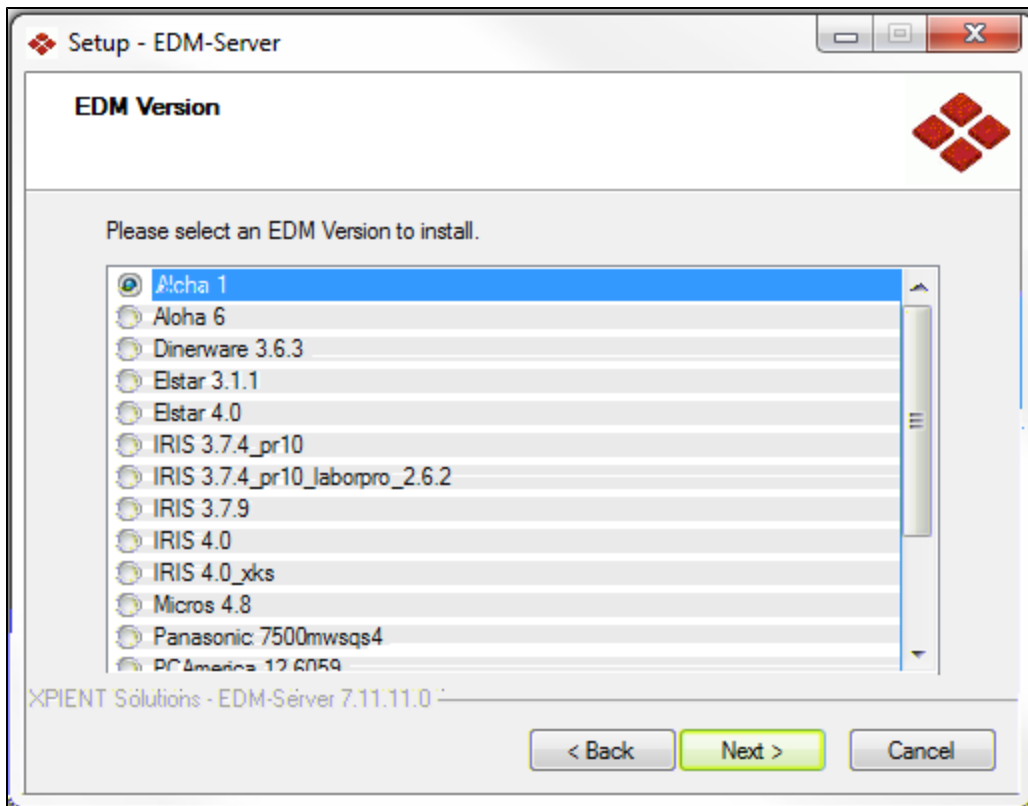
1. Run *EDM-Server-Setup.exe* file. This opens the first prompt, which indicates the version of EDM you are installing.
2. Click *Next* to proceed.
3. The next prompt is the License Agreement. Read the agreement then select the *I agree* button.
4. Click *Next* to proceed.

Step 2 - Service or Stand Alone



1. Select *Service* from this prompt to install EDM as a service.
2. Click *Next* to proceed.

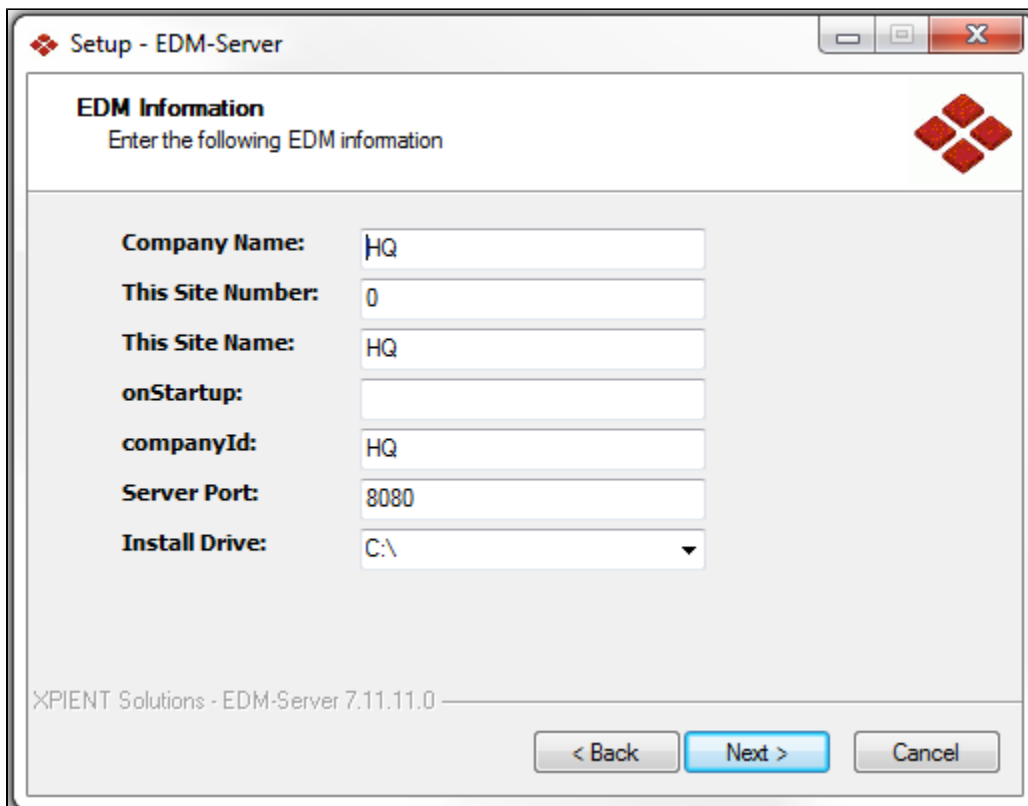
Step 3 - EDM Version



The next prompt is where you select the POS product and version that EDM will be managing.

1. Select the applicable POS product and version.
2. Click *Next* to proceed.

Step 4 - EDM Information



The next prompt is where you enter the information about this specific EDM server setup. The following table provides instructions for each field. Some of the fields are automatically populated with the default value, but these values can be changed.

Field name	Description
Company Name	Type the name of your company.
This Site Number	Type a number to identify the site. Servers are number 0, by default.
This Site Name	Type a name to identify the Server site.
Install Central	This value must be Y.
onStartup	Type the command you want EDM to execute upon startup. This can be left blank, if not needed.
CompanyId	Type an ID to identify your company that is shorter than the name.
Server Port	Type the port that EDM uses for communication. <div style="border: 1px solid yellow; padding: 5px; text-align: center;"> Make note of this information. You will need it later. </div>
Install Drive	Type the path of the EDM installation folder. The default is C:\EDMServer. You can specify a different drive.

After you have completed all the fields, click *Next* to proceed.

Step 5 - System Data Source Information

Setup - EDM-Server

System data source information
Please enter the information needed to connect to your EDM system data source database.

Name: EDMWebHQ

Database: EDMWebHQ

Server: localhost

User Name: sa

Password:

Version: 1

Type: SQLSERVER_JDBC

Connection:

XPIENT Solutions - EDM-Server 7.11.11.0

< Back Next > Cancel

The next prompt is where you specify details about the EDM database. The EDM installation will create this database when the server is started after the installation is complete, so do not identify an existing database. The following table provides instructions for each field. Some of the fields are automatically populated with the default value, but these values can be changed.

Field name	Description
------------	-------------

Name	Type the name of the system data source for EDM.
Database	Type the name of the database EDM will create to store its data.
Server	Type the location of the server where the database will exist. An instance can be specified (exp: localhost\xsiris).
User Name	Type the user name EDM will use to create and access the database. Leave this field blank if integrated security is used.
Password	Type the password associated with the user name above, if there is one. Leave this field blank if integrated security is used.
Version	This value is used to keep track of the version of the EDM database. This value is for the user to track their changes to the database. This number is not used by the system
Type	Type the connection type used to connect to the database. Use the default value if you do not know.
Connection	Type the connect string for integrated security. Leave this field blank if integrated security is NOT used.

After you have completed all the fields, click *Next* to proceed.

Step 6 - Authentication data source information

Setup - EDM-Server

Authentication data source Information
Please enter the information needed to connect to your EDM authentication database.

Name: authenticationDataSource

Database: Auth

Server: localhost

User Name: sa

Password:

Version: 1

Type: SQLSERVER_JDBC

Connection:

Use system datasource info? :

XPIENT Solutions - EDM-Server 7.11.11.0

< Back Next > Cancel

Please enter the information needed to connect to your EDM authentication database. This database contains the mapping of the users to the companies that they have access to, and is maintained by the User editor.

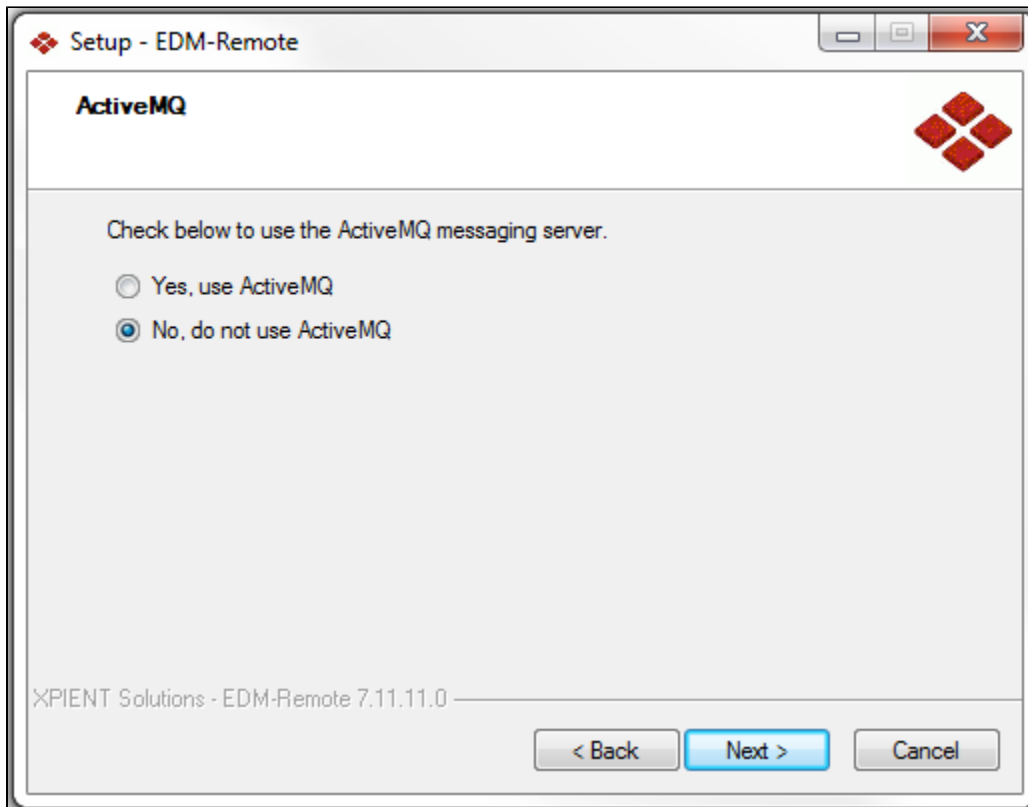
Some of the fields are automatically populated with the default value, but these values can be changed.

Field name	Description
Name	Type the name of the authentication data source for EDM.
Database	Type the name of the authentication database EDM will create to store its data.
Server	Type the location of the server where the authentication database will exist. An instance can be specified (exp: localhost\xsiris).
User Name	Type the user name EDM will use to create and access the authentication database. Leave this field blank if integrated security is used.

Password	Type the password associated with the user name above, if there is one. Leave this field blank if integrated security is used.
Version	This value is used to keep track of the version of the EDM authentication database. This value is for the user to track their changes to the database. This number is not used by the system
Type	Type the connection type used to connect to the authentication database. Use the default value if you do not know.
Connection	Type the connect string for integrated security. Leave this field blank if integrated security is NOT used.
Use system data source info?	If you select this option, an authentication data source is not created. EDM will use the system data source to create one itself.

After you have completed all the fields, click *Next* to proceed.

Step 7 - wpauthenticationDataSource



1. Indicate whether EDM will use the ActiveMQ messaging server. The default is *No*.
2. Click *Next* to proceed.

Step 8 - Ready to Install

1. Review the installation information you provided. Click *Back* to return to a previous prompt and make changes, if needed.
2. Click *Install* to launch the installation.

Step 9 - Starting the Apache Tomcat Service

1. When the installation is complete, click *Finish*.
2. From the Windows Start menu, open the Control Panel.
3. Navigate to *Administrative Tools>Services*.
4. Find *Apache Tomcat Service* in the list of services.
5. Right-click *Apache Tomcat Service*, and then select *Start* from the menu that appears.

Step 10 - Testing the Server

The following describes how to test the EDM installation.

1. Open your browser to: <http://localhost:8080/EDM>
2. If you are not on the same machine as the server, replace *localhost* in the URL with the server name or IP address. Use the port number you specified in [Step 4](#) above.
3. Logon to EDM using one of the following predefined User ID/Password combinations.

User ID / Password	Description
remote/remote	Typical login for a user at a remote site who only has permission to process transactions.
central/central	Typical login for a central user who has permission to use all the forms and functions at headquarters.
admin/admin	Administrative login with all permissions including setting up new users and roles and modifying the forms and queries in the application. If the application loads then EDM has installed successfully. This may take a moment the first time.

If the application does not load, refer to the log file at `\\EDMServer\Webapps\EDM\log.txt`. Additional [troubleshooting](#) steps can be found at the bottom of this page.

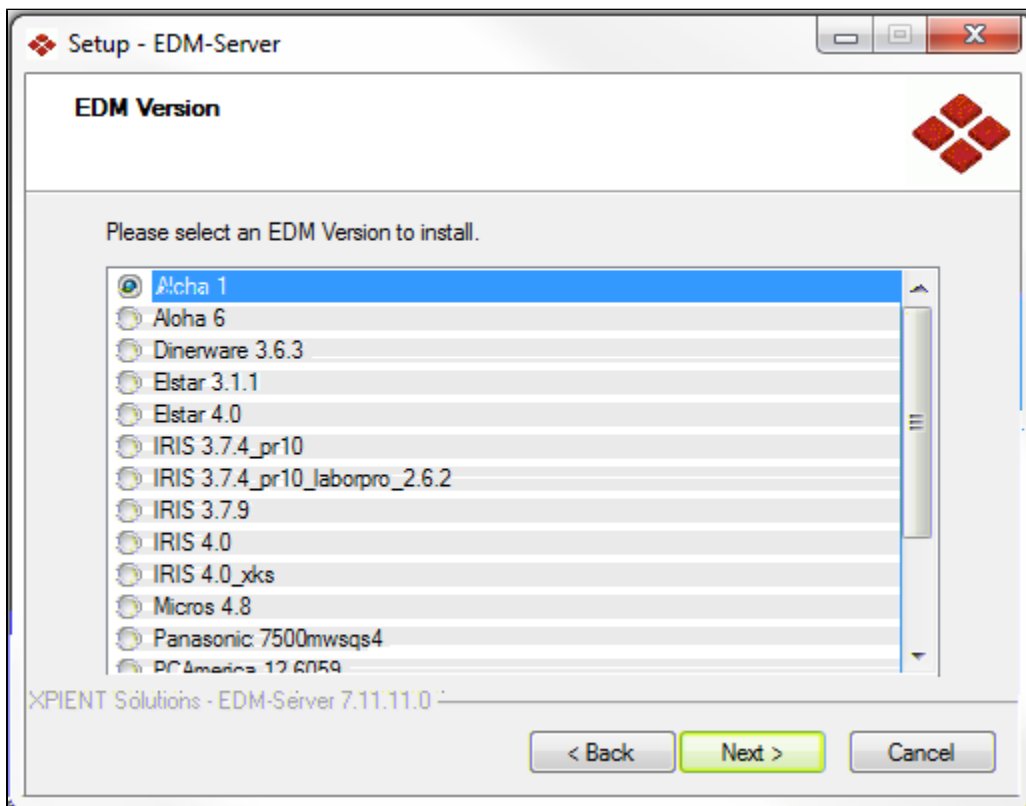
Remote Installation

Step 1 - Execute the Installation File

If `Install_Customization.ini` exists in the same directory as `EDM-Remote-Setup.exe`, then rename `Install_Customization.ini`.

1. Run `EDM-Remote-Setup.exe` file. This opens the first prompt, which indicates the version of EDM you are installing.
2. Click *Next* to proceed.
3. The next prompt is the License Agreement. Read the agreement then select the *I agree* button.
4. Click *Next* to proceed.

Step 2 - EDM Version



The next prompt is where you select the POS product and version that EDM will be managing.

1. Select the applicable POS product and version.
2. Click *Next* to proceed.

Step 3 - EDM Information

The next prompt is where you enter the information about this specific EDM site. The following table provides instructions for each field. Some of the fields are automatically populated with the default value, but these values can be changed.

Field name	Description
Company Name	Type the name of your company.
onStartup	Type the command you want EDM to execute upon startup. This can be left blank, if not needed.
CompanyId	Type an ID to identify your company that is shorter than the name.
Transfer Expression	Leave this field blank.
Install Drive	Type the path of the EDM installation folder. The default is <i>C:\EDMWeb</i> . You can specify a different drive.

After you have completed all the fields, click *Next* to proceed.

Step 4 - System Data Source Information

The next prompt is where you specify details about the EDM database. The EDM installation will create this database when EDM is started after the installation is complete, so do not identify an existing database. The following table provides instructions for each field. Some of the fields are automatically populated with the default value, but these values can be changed.

Field name	Description
Name	Type the name of the system data source for EDM.
Database	Type the name of the database EDM will create to store its data.
Server	Type the location of the server where the database will exist. An instance can be specified (exp: localhost\xsiris).
User Name	Type the user name EDM will use to create and access the database. Leave this field blank if integrated security is used.
Password	Type the password associated with the user name above, if there is one. Leave this field blank if integrated security is used.
Type	Type the connection type used to connect to the database. Use the default value if you do not know.
Connection	Type the connect string for integrated security. Leave this field blank if integrated security is NOT used.

After you have completed all the fields, click *Next* to proceed.

Step 5 - Data Source 1 Information

The screenshot shows a Windows-style dialog box titled "Setup - EDM-Remote". Inside, there is a section titled "Data source 1 Information" with a sub-instruction: "Please enter the information needed to connect to your EDM database." Below this are several input fields: "Name:", "Database:", "Server:" (with a dropdown arrow), "User Name:", "Password:", "Version:", "Type:", and "Connection:". At the bottom left of the form area is a checkbox labeled "Add another?". At the bottom right are three buttons: "< Back", "Next >" (highlighted in blue), and "Cancel". The footer of the window reads "XPIENT Solutions - EDM-Remote 7.11.11.0".

The next prompt is where you specify details about the EDM database. The EDM installation will connect to this database when EDM is started after the installation is complete, so do not identify an existing database. The following table provides instructions for each field. Some of the fields are automatically populated with the default value, but these values can be changed.

Field name	Description
Name	Type the name of the system data source for EDM.
Database	Type the name of the database EDM will create to store its data.
Server	Type the location of the server where the database will exist. An instance can be specified (exp: localhost\xsirsis).
User Name	Type the user name EDM will use to create and access the database. Leave this field blank if integrated security is used.
Password	Type the password associated with the user name above, if there is one. Leave this field blank if integrated security is used.
Version	Type the version of the database EDM is maintaining.
Type	Type the connection type used to connect to the database. Use the default value if you do not know.
Connection	Type the connect string for integrated security. Leave this field blank if integrated security is NOT used.
Add another	Select this option if EDM will maintain more than one data structure. You will then be able to enter another data source. EDM can maintain up to four data sources.

After you have completed all the fields, click *Next* to proceed.

Step 6 - Authentication data source information

Setup - EDM-Remote

Authentication data source Information
Please enter the information needed to connect to your EDM authentication database.

Name: authenticationDataSource

Database: Auth

Server: localhost

User Name: sa

Password:

Version: 1

Type: SQLSERVER_JDBC

Connection:

Use system datasource info? :

XPIENT Solutions - EDM-Remote 7.11.11.0

< Back Next > Cancel

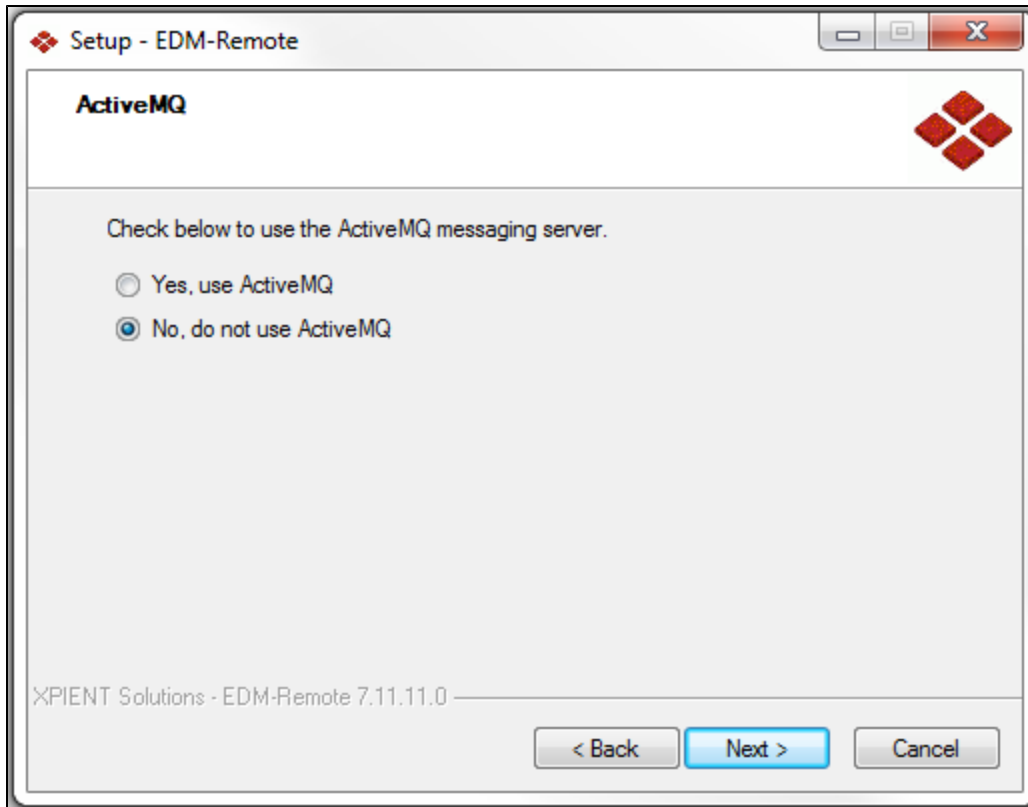
Please enter the information needed to connect to your EDM authentication database. This database contains the mapping of the users to the companies that they have access to, and is maintained by the User editor.

Some of the fields are automatically populated with the default value, but these values can be changed.

Field name	Description
Name	Type the name of the authentication data source for EDM.
Database	Type the name of the authentication database EDM will create to store its data.
Server	Type the location of the server where the authentication database will exist. An instance can be specified (exp: localhost\xsirir).
User Name	Type the user name EDM will use to create and access the authentication database. Leave this field blank if integrated security is used.
Password	Type the password associated with the user name above, if there is one. Leave this field blank if integrated security is used.
Version	This value is used to keep track of the version of the EDM authentication database. This value is for the user to track their changes to the database. This number is not used by the system
Type	Type the connection type used to connect to the authentication database. Use the default value if you do not know.
Connection	Type the connect string for integrated security. Leave this field blank if integrated security is NOT used.
Use system data source info?	If you select this option, an authentication data source is not created. EDM will use the system data source to create one itself.

After you have completed all the fields, click *Next* to proceed.

Step 7 - ActiveMQ



1. Indicate whether EDM will use the ActiveMQ messaging server. The default is *No*.
2. Click *Next* to proceed.

Step 8 - Ready to Install

1. Review the installation information you provided. Click *Back* to return to a previous prompt and make changes, if needed.
2. Click *Install* to launch the installation.

Step 9 - Testing the Installation

The following describes how to test the EDM installation.

1. After the installation is complete, click *Finish*.
2. Navigate to the `\EDMWeb` folder and run `go.bat`.

`Go.bat` creates the EDM system database and connects to the data source(s) you configured. If EDM is launched, then EDM was installed successfully. If the application does not launch, refer to the log at `\EDMWeb\log.txt`. Additional troubleshooting steps can be found below.

Common Issue Troubleshooting

IMPORTANT NOTE

If you are upgrading EDM from Apache 6/7 using an SSL connection, you must make the following change to the `server.xml` file located in the `\EDMServer\conf` folder. The headers in the Connector protocol section that include the certificate information must read as follows:

```
keystoreType="",keystoreFile="", and keystorePass=""
```

The EDM service will not start until after you make this change.

- Make sure the user specified for the EDM installation has permission to create databases.
- Confirm the password entered for users is correct. This can be corrected in the `config.xml` file.
 - `Config.xml` is located at: `\EDMWeb\Config.xml`
 - `Server Config.xml` is located at: `\EDMServer\webapps\EDM\WEB-INF\classes\Config.xml`
- If you are using the connection type used to connect to the database, ensure the correct database is specified in the connect string for integrated security.

- If you see Java errors when loading the EDM Central webpage, ensure the Java version is at least 1.8 on the machine from which you are browsing to the website.

Configuring the Xenial Connector

This page provides instructions on how to configure EDM Remote to use the Xenial Connector. Click [here](#) to go to the Installation Guide home page.

How to Configure the Xenial Connector

Follow these steps to configure EDM Remote to use the Xenial Connector.

1. Upgrade EDM to 7.12.152 (or later).
2. Create a Xenial connection configuration file named **configurationXENIAL.xml**.
3. Add a configuration parameter named **configXENIALFile** that points to the location of **configurationXENIAL.xml** (starting from the root of the classes directory). For example, **configXENIALFile = "configurationXENIAL.xml"** (if the file is located at *WEB-INF/classes/configurationXENIAL.xml*).
4. Add the tables to transfer to Xenial to a **XenialTableList** node in the snapshots file.

Configuration File Parameters

The configuration file should provide the following parameters.

Parameter	Description
name	Unique name of feature
endpoint	"local" unless AWS is utilized for the connector
hostName	Name of Amazon host where the transaction is sent (if applicable)
url	URL of the endpoint where the transaction is sent (if applicable)
pathName	Path on which to deposit files
useTranRefresh	Use transaction-based table refresh (this should always be set to true)

Sample Xenial configuration file

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE S3Configuration>
<XENIALConfiguration version="1">
  <feature
    name = "XENIALmapper"
    endpoint = "local"
    hostname = "HostURL"
    urlName="/dev/storeupdate"
    pathName="C:\PATH_TO\XCC\messages\out"
    useTranRefresh = "true"
  />
</XENIALConfiguration>
```

Sample XenialTableList for the snapshots file

```
<snapshot name="XenialTableList">
  <table name="DB_Table_Name" />
  <table name="DB_Table_Name2" />
  <table name="DB_Table_Name3" />
</snapshot>
```

Release Notes

This page provides access to the EDM release notes.

EDM 7.10 - Release Version - 2015-11-20

This page details the release of the XPIENT product, EDM. This update was delivered on **2015-11-20**.

Click [here](#) to go back to the Release Notes home page.

Contents

This release delivers an upgrade installation of EDM. This release upgrades **EDM v7.9** to **EDM v7.10**.

Java 1.8 Required

This version of EDM must be run with Java version 1.8 or higher. Java 1.8 will be installed with EDM by default when you use the installation program.

Installing Java 1.8 is necessary due to Oracle discontinuing security fixes to older Java libraries, which have been retired. If you just upgrade the JAR files at the store without using the installation program, you must also upgrade Java to the latest version. If you use a different version of Java than what is shipped with EDM, you may also need to change your End of Day scripts to use the correct version of Java. This can be accomplished by using EDM's new [File Sharing](#) and [Server Text File Editor](#).

This release was tested with Java version 1.8 update 60.

Documentation

This release supports all EDM functions detailed in the latest version of our documentation:

- [Installation Guide](#)
- [User Guide](#)

Release Notes

The issues listed in this section are addressed in this release.

Business Value

This EDM release offers features and functionality that provide business value across the following areas:

- Enhanced EDM user management
- Enhanced Security authentication
- New forms for updating XENON

Resolved Defects

This table describes the defects that were resolved in this release.

Issue number	Issue description	Issue details
EDM-2692	Item Master - Pricing tab generates error	An error was generated when an item's parentID was selected under Child Pricing. In response to this issue, the forms have been updated. The Child Pricing records that show on the Pricing form are read-only. When adding a Child Record, the synopsis box to the right of Child Pricing opens a sub-form where the records are added. The sub-form also displays the current child records and the properties of the child item.
EDM-2695	Item Combos form confirmation prompt generates error	The confirmation prompt check box on the Item Combos form displayed an error when it was selected.
EDM-2696	Modifier Group Sets form generates error	After opening the Modifier Group Sets form, an error was generated after a package and locations were selected.
EDM-2699	Item Master by SKU - General tab radio buttons not functioning	When the Default or Tare Menu radio buttons were selected from the Item Master by SKU - General tab, the None button remained selected. When the None button was unselected, the other buttons were also unselected. The selected options were not saved once the form was closed.

Enhancements

This table describes the enhancements that were implemented in this release.

Issue number	Issue description	Issue details
EDM-2701	User editor	The User editor has been upgraded and now supports editing all users in any company, as long as the current user has access to the company and the user. If the current user has access to only 1 company, the editor hides UI controls that are only pertinent to multiple companies. The Job Code forms enable XENON user to manage Job Codes from a central form, so that job codes can be added, edited and distributed appropriately throughout the enterprise.
EDM-2710 EDM-2728 EDM-2731 EDM-2793	Security authentication	Authentication is now done from a separate database. All users are migrated from the company database into a new authentication database named "auth" by default. This database contains the mapping of the users to the companies that they have access to, and is maintained by the upgraded User editor. This means that only 1 user record exists, and that a user with access to multiple companies no longer needs to change their password in each separate company. <div style="border: 1px solid yellow; padding: 5px; margin: 10px 0;"> <p>When the authentication database is initially created, if there is a user record in 2 or more companies with different passwords, only the first user record is created. If you have multiple companies in your EDM server, check the log file after initial startup to see if there were any users with this scenario.</p> </div> <p>RESTful API - There is a new API that can be used to authenticate with EDM. You can read more about it here.</p>
EDM-2782	EDM forms for XNU	Numerous EDM forms have been created for the XPIENT product XENON. Please contact your Professional Services Project Manager for more details.
EDM-2802	Discount form - Days of Week	Check boxes are now utilized to set which day of the week a discount is valid on the Discount form.
EDM-2833	Menu Editor - Button Sequencing	The IRIS menu editor menu buttons only update the sequence numbers when absolutely necessary, which eliminates the number of updates that are sent to remote sites and minimizes the threat of packages released at different times updating the wrong menu buttons.
EDM-2885	BulkInserter upgraded	The Bulk Inserter logic that EDM uses to insert a lot of records quickly into a SQL Server database has been upgraded to be faster by using the "Insert into tbl (col1, col2, col3) values (va1, val2, val3), (val4, val5, val6), (val7, val8, val9)" format. This allows SQL Server to create a plan for the inserts and minimizes the number of batches and transactions that are created.

N/A	Importing Transactions	Importing transactions from a file now creates the associated package and will also populate the default sites from what was specified in the import file. This means that you can bring up the package and all of the sites that had imported transactions will be selected by default when the site selection screen is shown.
-----	------------------------	--

Installation

To get the installation file for EDM (and the required license files), please contact your Professional Services Project Manager. Once you have the installer, please follow the steps detailed in the [Installation Guide](#).

EDM 7.11.33 - Release Version - 2015-02-03

This page details the release of the XPIENT product, EDM. This update was delivered on **2015-02-03**.

Click [here](#) to go back to the Release Notes home page.

Contents

This release delivers an upgrade installation of EDM. This release upgrades **EDM v7.10** to **EDM v7.11.33**.

This release was tested with Java version 1.8 update 65.

Documentation

This release supports all XPEDITE functions detailed in the latest version of our documentation:

- [Installation Guide](#)
- [User Guide](#)

Release Notes

The issues listed in this section are addressed in this release.

Business Value

This EDM release offers features and functionality that provide business value across the following areas:

- AWS package transfer speed has been enhanced by implementing logic that transfers more than one file at a time
- Installation now prompts for Authentication data source specifications
- Resolved defects found in EDM v7.10

Resolved Defects

This table describes the defects that were resolved in this release.

Issue number	Issue description	Issue details
EDM-2629	New package creation - Search field error	If the <i>Search</i> field contains any characters when the user attempts to create a new package, an error message is generated and all packages are hidden from view. This defect has been resolved.

EDM-2753	Database locked up during Audit table query	<p>With this release, EDM now warns the user before generating a Transaction Report (including Price and Tax changes in IRIS) if the specified parameters for the report will return a very large number of transactions. This allows the user to cancel the operation and reset the parameters to values that will return a more workable number of transactions.</p> <p>You can specify the minimum number of transactions that generates this warning using the following setting in the config.xml file. The default value is 1000.</p> <ul style="list-style-type: none"> • <i>transactionReportItemWarningCount</i>
EDM-2880	Users could assign item records to multiple Major Item Categories	<p>With this release, users can only assign item records to a single Major Item Category when using the Item Master forms for IRIS. EDM validates that the Major and Minor Item categories assigned to an item record are the same on both the parent and the child forms.</p>
EDM-2994 EDM-2996	Importing Transactions failed if the Audit table was locked	<p>When the <code>importTransactionsFromFile()</code> function was used to import data into EDM central, the data was updated in the central data tables (e.g. IRIS_dbo_tbl_ItemPricing), but the Audit table was locked and the transaction was not inserted in the audit trail. When this occurred, the remote store never received the update. A table refresh would have to be sent to the remote store.</p> <p>In response to this issue, logic has been implemented that notifies the Bulk Inserter utility of the status of the Audit table. Additionally, the performance speed of the Bulk Inserter utility has been enhanced.</p>
EDM-3017	Issue with cloning a menu item record with components	<p>When cloning a menu item that has records in <code>tbl_ItemComponents</code> (e.g. Conversational Ordering menu items), the item details were cloned successfully at the Central site, but the client was not updated properly. Prior to this release, it was necessary to send a separate refresh of <code>tbl_ItemComponents</code> after a client site processed a package with new menu items.</p>
EDM-3024	Redirected EDM users results in error	<p>An Apache Tomcat - Invalid State Error was generated when EDM users were redirected when attempting to access EDM. Accessing EDM directly did not generate this error.</p>
EDM-3034	Access was denied to a newly added company	<p>After adding a new company, access to the company was not granted to EDM users.</p> <p>The following changes have been implemented in response to this issue. If a policy file doesn't already exist when EDM creates a central database, EDM creates the "default" users/roles/permissions that would be in the policy.xml file.</p> <div style="border: 1px solid yellow; padding: 10px; margin-top: 10px;"> <p>If the Auth database contains Admin/Central/Remote users with passwords that are different from the default passwords, you will need to manually add the users and user/company mappings. EDM will not grant the users access to the new company if their passwords do not match.</p> </div>

Enhancements

This table describes the enhancements that were implemented in this release.

Issue number	Issue description	Issue details
EDM-2752	AWS package transfer speed	<p>AWS package transfer speed has been enhanced by implementing logic that transfers more than one file at a time. Transaction files are now uploaded to Amazon S3 in parallel so that smaller file transfers can continue while larger file transfers are in progress. This requires the installation of Java 1.8 runtime on the server machine. The new Java 8 "Streams" API is used to send the transaction files in parallel to the Amazon S3 server.</p>
EDM-2883	Installation files for IRIS 4.0 and the XPEDITE Kitchen System	<p>This release includes the installation files for IRIS 4.0 and the XPEDITE Kitchen System. New EDM forms have been developed for the purpose of configuring the XPEDITE Kitchen System.</p> <p>For information about the XPEDITE Kitchen System, please click here.</p>

EDM-3009	EDM installation prompt for authentication database settings	<p>Authentication for EDM is done from a separate database. All users are migrated from the company database into a new authentication database named "auth" by default.</p> <p>Beginning with this release, when you install EDM you are prompted to specify the location of the authentication database. This database contains the mapping of the users to the companies that they have access to, and is maintained by the User editor.</p> <div data-bbox="428 306 891 667" style="border: 1px solid gray; padding: 5px; margin: 10px 0;"> </div> <p>If you select the Use system datasource info option from this new prompt, an Authentication datasource is not created. EDM will use the system datasource to create one itself.</p> <div data-bbox="423 770 1482 852" style="border: 1px solid gray; padding: 5px; margin: 10px 0; text-align: center;"> <p>Click here to read more about the API used to authenticate with EDM.</p> </div>
EDM-3028	RFC-3339 support for date/time format in EDM Client	<p>EDM has been updated to parse incoming date/time strings that are received in RFC-3339 format and convert them to the server's timezone before processing.</p>

Installation

To get the installation file for EDM (and the required license files), please contact your Professional Services Project Manager. Once you have the installer, please follow the steps detailed in the [Installation Guide](#).

IMPORTANT NOTE

If you are upgrading EDM from Apache 6/7 using an SSL connection, you must make the following change to the server.xml file located in the \EDMServer\conf folder. The headers in the Connector protocol section that include the certificate information must read as follows:

```
keystoreType="",keystoreFile="", and keystorePass=""
```

The EDM service will not start until after you make this change.

EDM 7.12.223 - Release Version - 2016-09-07

This page details the release of the XPIENT product, EDM. This update was delivered on **2016-09-07**.

Click [here](#) to go back to the Release Notes home page.

Contents

This release delivers an upgrade installation of EDM. This release upgrades **EDM v7.11.33** to **EDM v7.12.223**

This release was tested with Java version **1.8** update **92**.

Documentation

This release supports all XPEDITE functions detailed in the latest version of our documentation:

- [Installation Guide](#)
- [User Guide](#)

Release Notes

The issues listed in this section are addressed in this release.

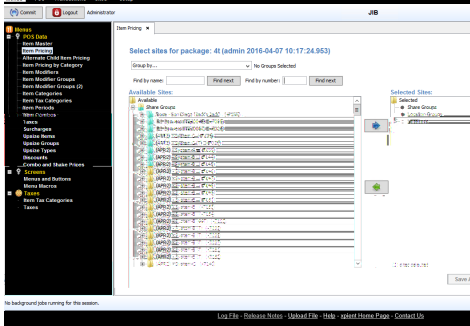
Business Value

This EDM release offers features and functionality that provide business value across the following areas:

- Improved security
- Ability to cut and paste store list during package creation improving user efficiency
- Improved bulk inserter enhances product performance
- Support for Zingo Backoffice
- REST API to connect with the application
- Resolved defects found in EDM v7.11.33

Resolved Defects

This table describes the defects that were resolved in this release.

Issue Number	Issue Description	Issue Details
EDM-2526	Menu Button 'Type' - All Types were not listed under Menu Item Properties	Not all Menu Button Types were available for selection when editing menu item properties using the screen properties form.
EDM-3027	Setting System user affected multiple HTTP request calls	If the current user did not have database access to perform a function, the system changed the user to a "System" user with full database access. Now the system allows the user to perform the function without changing their database access rights.
EDM-3044	Error when editing entries on newly-created quick forms	Oracle removed the xBase driver from Java 8, which caused EDM to fail when connecting to databases that require this driver. EDM has been changed to use the hxtt driver, which restores support for DBASE and FoxPro databases. Microsoft Access is no longer supported in EDM.
EDM-3066 EDM-3072	Inability to search for price groups "-7150"	<p>The user was unable to search for the price groups, as shown in this screenshot:</p>  <p><i>Click the image to view the full size version</i></p> <p>The Site Selection Find by Number field will now accept "-" (only) in the first position, allowing users to enter negative numbers.</p>
EDM-3068	Cannot access EDM forms through Zingo Backoffice UI	The user was unable to access EDM forms from the Zingo Backoffice UI.

EDM-3074	Incorrect X/Y coordinates error	This error was generated when viewing button properties.
EDM-3075	"Assumed decimals" were ignored in grid view	The decimal field with an assumed decimals setting of 2 was only displayed in edit mode. Now the assumed decimals are also displayed in view mode.
EDM-3095	"Out of Bounds" error when Copy Record was used with View() during EDM upgrade	The "Out of Bounds" error is no longer generated during an EDM upgrade.
EDM-3103	Public ability to upload files to EDM server	Prior to this release, it was possible for anyone was able to upload a file to the EDM server.
EDM-3111	Buttons disappeared when editing properties	Menu buttons would disappear if the user double-clicked a button on the edit properties window, clicked Cancel, and then repeated these steps.
EDM-3124	Config Group Editor - MsgError: Record already exists at all selected sites	When using the Config Group Editor to setup new store locations, an error was generated stating "Record already exists at all selected sites". In response to this issue, the importer has been updated to ensure all Section and Key names use lowercase letters.
EDM-3127	Subform data disappears from cells in a new record added in spreadsheet view	Some data was disappearing when a new record was added in spreadsheet view. Now all values remain visible as the user inputs them.
EDM-3128	tbl_ItemSku not showing versioning differences	Two scenarios were observed: <ul style="list-style-type: none"> • When a package to update a SKU was committed, only the updated record remained in the table once the package was processed. • There have been cases where only part of a table was sent on a table refresh. The table would have to be refreshed multiple times.
EDM-3132	Problem saving ComboBoxes (and TextFields) from GridWin	The data in a form was not saved correctly. This issue was previously hidden by the fact that the combobox values were being lost before users were able to save the form.
EDM-3136	Error when editing fields in GridView	An error was generated when the user edited a field in a subform in GridView.
EDM-3139	Leading zeros in combo boxes	When the user edited a combo box the first field included a leading zero. The actual value did not display until the field was selected. Additionally, an error related to a datastring was generated when the user edited the field.
EDM-3141	Subform with GridView not updated	When the user added a new item in a subform in GridView, the grid was not updated and there was a bleed through between the forms.
EDM-3149	Setup - Application - Tables lockup	When the user navigated to Setup - Application - Tables to view a table, occasionally the UI locked up and no data was displayed. The browser had to be refreshed to exit.

Enhancements

This table describes the enhancements that were implemented in this release.

Issue Number	Issue Description	Issue Details
EDM-3061	Improved BulkInserter	BulkInserter now only sends the row values instead of a complete insert statement for each row, which significantly improves product performance.
EDM-3062	Log level server functions	Now when <i>Set Log Level</i> is selected, a new view will be displayed with two menu items: one for console level and one for file level. From here the user can select the desired log level (e.g. INFO, WARNING, SEVERE).
EDM-3064	Make log.txt file private	The log.txt file for EDM is now only accessible to EDM users.
EDM-3096	Cut and paste store list in a form	You can now cut and paste a store list during the package creation process. This functionality is available in the following areas: <ul style="list-style-type: none"> • Select Sites for Package form: In the <i>Find by Number</i> field • Select Sites to Commit form: In the <i>Find by Number</i> field

EDM-3108	Update EDM Java version from 1.8.0_60 to 1.8_92	Java 1.8_92 provides enhanced security for EDM.
EDM-3115	Expose REST Endpoint that provides Company/ConnectionString collection	EDM now exposes a collection of companies and their connection strings via the auth REST API. This enables a POS application to access companies configured via EDM.
EDM-3129	New EDM forms for Zingo Backoffice	EDM has been updated with forms for Zingo Backoffice.

Installation

To get the installation file for EDM (and the required license files), please contact your Professional Services Project Manager. Once you have the installer, please follow the steps detailed in the [Installation Guide](#).

IMPORTANT NOTE

If you are upgrading EDM from Apache 6/7 using an SSL connection, you must make the following change to the server.xml file located in the \EDMServer\conf folder. The headers in the Connector protocol section that include the certificate information must read as follows:

```
keystoreType="",keystoreFile="", and keystorePass=""
```

The EDM service will not start until after you make this change.

EDM 7.13 - Release Version - 2017-06-21

This page details the release of the Enterprise Data Manager (EDM) application. This update was delivered on 2017-06-21.

Click [here](#) to go back to the Release Notes page.

Contents

This release delivers an upgrade installation of EDM. This release upgrades EDM to v7.13.

Documentation

This release supports all EDM functions detailed in the latest version of our documentation:

- [Installation Guide](#)
- [User Guide](#)

Support for Xenial

Clients running EDM 7.12 builds with preliminary Xenial support should add the following to their Xenial configuration as noted here: [Configuring the Xenial Connector](#). Java, Tomcat, and SQL Server drivers have been updated.

- useTranRefresh= "true"

Release Notes

The issues listed in this section are addressed in this release.

Resolved Defects

This table describes the defects that were resolved in this release.

Issue number	Issue description	Issue details
EDM-3144	New menus were not appearing in search results	Menu and menu item searches now return the complete and accurate results.
EDM-3145	Package search returned wrong package	When a package was found and selected from a search, a different package opened.
EDM-3170	Exiting a menu screen returned user to the wrong position on list	When the user exited from a menu screen, they were taken to that menu's position on the list. Now users are returned to the original place they were on the menu list.
EDM-3173	Calendar Detail dates incorrect	Calendar Detail dates were two days behind the current date. This issue was fixed by the SQL Server driver updates.
EDM-3193	Amazon Web Services (AWS) incorrect path	With this release, EDM now creates the appropriate location directories in AWS.
EDM-3194	Multiple Inventory ID numbers linked to one ExternalItemID	EDM allowed multiple Inventory ID numbers to be linked to one ExternalItemID. EDM's handling of acting keys on updates now prevents this from occurring.
EDM-3197	Menu tab positioning issue	After an edit was made the menu tabs were not positioned correctly. Menu tabs are now positioned correctly after edits are made.
EDM-3198	Paste Store List function issue	The <i>Paste Store List</i> function failed when the store list was very long. List pasting has been modified to accommodate long store lists.
EDM-3199	Validate Data notification issue	The <i>Validate Data</i> function only notified the user when a failure occurred. A new <i>validateTablesAndVerify</i> function has been created to explicitly notify users when tables are successfully validated.
EDM-3200	Issue with selecting all available locations for a package	When the user double-clicked the <i>Available</i> folder when selecting sites for a package, not all sites were selected.
EDM-3220	Issue with editing a new record after it is saved	When items were added to a subform and saved, the parent form was also saved; since the form was opened as a new record, edit mode did not function properly until the form was closed.

Enhancements

This table describes the enhancement that was implemented in this release.

Issue number	Issue description	Issue details
EDM-2638	Increasing the default decimal scale	EDM has been updated to accept up to 10 decimal places in the schema for a Float column.

Installation

To get the installation file for EDM (and the required license files), please contact your Professional Services Project Manager. Once you have the installer, please follow the steps detailed in the [Installation Guide](#).

Release Notes Archive

This section contains the release notes for the XPIENT product, EDM. The release notes in this section are for versions of EDM prior to EDM 7.10 Click [here](#) to go back to the Release Notes home page.

Contents

EDM 6.0 Release Notes

This page provides access to the EDM 6.0 release notes. The release notes are sorted by version number in descending order. The release notes for the latest version of EDM are provided at the top of the list.

EDM 6.9 Release Notes

Enhancements

The following enhancements were made in this release

Added IRIS Price Change by Price report

Added report for IRIS users to view price transactions grouped by the item number, old value, new value, and effective date to make it easier for users to see what price changes are pending with less duplicate information. Transactions that are the same at multiple sites are only shown once instead of once for each site.

Added IRIS Tax Change report

Added report for IRIS users to view tax transactions grouped by the tax id, old value, new value, and effective date to make it easier for users to see what tax changes are pending with less duplicate information. Transactions that are the same at multiple sites are only shown once instead of once for each site.

Show and hide form controls at runtime

Form designers can now call functions to show or hide controls on a form at runtime.

For example, to hide a control based on the value of another control use an expression like this:

```
=iif( Price > 5, setControlProperty( 'tax3', 'visible', 'N' ), setControlProperty( 'tax3', 'visible', 'Y' ) )
```

Enable and disable form controls at runtime

Form designers can now call functions to enable or disable controls on a form at runtime.

For example, to hide a control based on the value of another control use an expression like this:

```
=iif( Price > 5, setControlProperty( 'tax3', 'enabled', 'N' ), setControlProperty( 'tax3', 'enabled', 'Y' ) )
```

Added remote host to HTTP request errors

Added the remote host to the error message logged by the server when it cannot process an incoming HTTP request to help the administrator find the cause of the problem.

Lock transaction processing once instead of once per site

Changed transaction processing so that it only tries to obtain the transaction processing lock once instead of once for each site. This solves the problem of the transaction processing loop taking a very long time when transaction processing is already locked because it was waiting up to 5 seconds per site for the lock request to time out.

Changed Site Manager to only display once

Changed Site Manager so that if it is already displayed on a tab and the user tries to open it again, the existing tab is brought to the front instead of creating a new tab.

Enable users to specify custom plugin for list control row source

Form designers can now use plugin functions to return the row data for list and combo box controls on forms by setting the 'Row source type' property to 'Table/Query' and the 'Row source' property to 'plugin:' followed by the plugin function name. The plugin function must be loaded as a server plugin function in config.xml and must take one parameter which is a comma-delimited list of site id's and must return a MemoryDataSet object. For example, to call the plugin function getColorDataSet to get the rows for a combo box, set the 'Row source' property to 'plugin:getColorDataSet'.

Request table refresh as scheduled task now supports 'all' sites

Changed the server-side requestTableRefresh function so that if the site list parameter is blank or '*' then a refresh will be requested from all sites that the user has permission to access.

Update transaction status date on commit

Changed commit process so that when transactions are committed the status is set to committed and the status date is set to the current date and

time.

Added 'editIrisItemMenus' function

editIrisItemMenus - Edit IRIS item menus and buttons.

Syntax:

editIrisItemMenus('useMacros', 'range')

Parameters:

useMacros - True (1) if using a version of IRIS that allows buttons to be assigned to menu macros.

range - The range of item menu numbers that the user is allowed to edit. An example would be '1,3,5-10' to exclude item menu numbers 2 and 4 in the range 1-10.

Returns:

Void.

Resolved Defects

The following defects were resolved in this release.

Allow user-initiated transaction processing to continue after logout

Changed logout procedure to not clear the session information about the users visit so that if there are background jobs such as transaction processing using the session information, they will be able to continue.

Fixed label control alignment

Fixed label controls on forms to properly display left, center, or right alignment in both design view and form view.

Fixed value popup on check boxes, radio buttons, and toggle buttons

Changed forms so that the popup window that enables users to edit values for multiple sites when they click on a check box, radio button, or toggle button is not displayed twice.

Stop allowing form and form design to be open at the same time

Changed form processing so that if a user opens a form from the menu or by switching to form view or spreadsheet view from the form designer, when the package and site selection screen is visible, the form cannot be opened or designed again. Instead, if the user tries to open or design a form while the package and site selection screen for the form is displayed, the package and site selection screen is brought to the front.

Fixed form design already locked error

Changed form designer so that if user clicks the 'Form View' or 'Spreadsheet View' button and then cancels out of the package and site selection, the form designer is redisplayed. This fixes the problem of leaving the form design locked when the user cancels the package and site selection.

Fixed Ctrl-Enter window to stay in front of browser

Changed the location value popup window to be made the front window immediately after being opened to prevent the it from occasionally being displayed behind the browser window.

Fixed Site Manager total calculations

Fixed Site Manager to correctly calculate the number of successful sites, error sites, and unresponsive sites and to only include the sites the user has permission to view. Also changed the count of sites and transactions with errors to not include failed transactions that have been marked fixed.

Fixed EOFError when deleting row with termination date

Fixed the EOFError that sometimes occurred when deleting a row from a form using a package with a termination date.

Fixed automatic refresh to only send related tables that are maintained for site

Fixed the automatic refresh that occurs when an insert, update, or delete transactions fails at a remote site so that it sends a refresh of the table and only the related tables that are actually maintained for the site in the central database. This eliminates the "Table X not maintained for site" error that occurs in this situation.

Fixed form designer to correctly support tab controls on the tab of another tab control

Fixed the form designer so that you can drop controls onto sub-tabs and they set the Tab Control and Tab Page properties correctly and so that controls on one subtab don't display on other subtabs.

Fixed form designer to scroll correctly while dragging controls

Fixed form designer so that when controls are dragged above or below the design area, the design area scrolls correctly under the controls and when the user releases the mouse the controls are correctly positioned.

Fixed IRIS button label line breaks

Modified the IRIS menu editor so that button labels will break to the next line if a CR/LF character combination is found in the string and also if a CR character is found by itself. This resolves the issue of IRIS breaking button labels when a CR character is found but EDM ignoring the CR by itself as a line break character.

Fixed Sybase support to handle embedded backslashes in strings

Changed the Sybase support functions to handle string constants that contain backslashes, such as file paths, by replacing any embedded backslash characters with two backslash characters.

Fixed form designer to set control name when a field is double-clicked

Changed the form designer so that when users select a control then double-click a field in the field list it sets both the control name and source to the selected field unless that control name is already being used by another control.

EDM 6.8 Release Notes

Enhancements

The following enhancements were made in this release

Prevent multiple package/site selection tabs for the same form

Changed the package/site selection wizard so that if a user clicks on a form and the wizard is displayed and then the user clicks on the form again, the original tab will be brought to the front instead of creating a new package/site selection tab.

Allow startup with invalid role names

Changed startup process so that if an invalid role name is detected while loading users and roles, the system logs the error and continues loading. All users and roles will still be loaded correctly except that users that were assigned the invalid role name will no longer have that role. If the log reports an invalid role name, the administrator can just re-create the role and re-assign it to the appropriate users to resolve the issue.

Simplified site selection screen

Removed the drop-down from the site selection screen that enabled users to filter the list to only show share groups or location groups. Since the locations and groups are already in a tree and users can show or hide any branch of the tree, this function was redundant and was removed to simplify the user interface.

Resolved Defects

Changed Transaction and IRIS Price Change report to only show sites user is allowed to see

The Transaction report and the IRIS Price Change report were changed so that if the user leaves 'All' sites selected, only the sites they have permission to see are included in the report.

Fixed site groups on IRIS Price Change report

Changed IRIS Price Change report to group the transactions correctly under the section header with the site number and name.

Fixed 'Copy Record' to put all transactions in selected package

Changed 'Copy Record' function so that if there are child tables, the transactions for the child tables are added to the selected package just like the transactions for the main table.

Fixed error when changing Role name

Fixed the "concurrent modification" error that sometimes occurred when users change the name of a Role and try to save the change. Roles may not be deleted if any users have that role and if a role is updated, all users with that role will immediately have the updated role name and permissions.

Fixed Copy Record error

Fixed Copy Record function so that if certain sites can't be updated because they are part of a share group that was not completely selected, it simply doesn't copy the data to those sites instead of displaying an error message.

EDM 6.7 Release Notes

Enhancements

The following enhancements were made in this release

Improved performance of Copy Record function

The Copy Record function on forms has been made faster by using bulk transaction inserts for central databases running on SQL Server. The bulk inserts are enabled by default but they may be turned off by setting `useBulkInserts="N"` in `config.xml`.

Changed site selection screen

Changed site selection screen so that if a site appears multiple times on the tree of available sites, such as in a share group and in the 'Locations' branch, if the user selects the site, all instances of the site will be moved to the selected tree. Moving sites from the selected tree to the available tree works the same way.

Improved effective/termination date and time entry

Changed the effective and termination date and time fields to different controls to make them more usable and to make them work with any version of Java. The date controls highlight today's date and allow the user to click today's date even if it is already selected and the time field is a simple pick list of times similar to the one used in Microsoft Outlook and the user can type in their own time if desired.

Changed transaction report site section headers

Change the transaction report so that when it is run at a central site, transactions both from and to any remote site will be grouped under a site section heading with the remote site number and name.

Resolved Defects

The following defects were resolved in this release.

Made effective and termination time fields wider

Increased the width of the effective and termination time fields to ensure that the entire time would be visible without scrolling. Also removed the seconds from the time fields so that the times are entered simply as hours and minutes with an AM/PM indicator.

EDM 6.6 Release Notes

Enhancements

The following enhancements were made in this release

Support encrypting database connection strings and passwords

Added option to enable users to encrypt the data source connection string and/or the data source password in the `config.xml` file so that passwords are not in clear text in the file where unauthorized personnel may be able to view them.

To cause the system to encrypt the connection string of a data source add `encryptedConnect = ""` to the data source in `config.xml`. To cause the system to encrypt the password of a data source add `encryptedPassword = ""` to the data source in `config.xml`.

When the system starts up, if it finds both a `connect` and an `encryptedConnect` attribute it will encrypt the value of the `connect` attribute and resave the `config.xml` file with the new encrypted value and without the original connection string. If both a `password` and an `encryptedPassword` attribute are found, the system will encrypt the value of the `password` attribute and resave the `config.xml` file with the new encrypted value and without the original password.

To change either the connection string or the password after it has been encrypted, simply add the `connect` or the `password` attribute back into the `config.xml` with the desired new value and restart the system, the new values will be encrypted and saved in the `config.xml` file.

Added encryption options to AddDataSource function

Added flags to encrypt the connection string and/or the password when adding a new data source to the system configuration using the `AddDataSource` function.

`addDataSource` - Add a new data source to the system configuration.

Syntax:

```
addDataSource( 'name', 'version', 'type', 'connectionString', 'encryptConnectionString', 'database', 'server', 'trusted', 'user', 'password', 'encryptPassword', 'isSystemDataSource')
```

Parameters:

name - Data source name used by the system.

version - Application-specific database version number.

type - Data source type, see `DbType` class.

connectionString - Connection string for data source. If specified, it will be used as-is to connect to the data source. If not specified, it will be built from other attributes. This allows users to customize the connection string as necessary. Care should be taken that the database and server names in the connection string match the database and server name from the other attributes.

encryptConnectionString - 'Y' if connection string should be encrypted when saved.

database - Database name.

server - Server name, may have :PORT or :SID suffixes.

trusted - 'Y' if this is ODBC data source that uses Windows login, else 'N'.

user - User name used to log into data source.

password - Password used to log into data source.

encryptPassword - 'Y' if password should be encrypted when saved.

isSystemDataSource - 'Y' if this is the system data source.

Returns:

Void.

Improved performance of copying records

Improved the performance of the 'Copy Record' function on all forms so that all the rows for the selected sites in each related table are copied with a single SQL statement. This is the default behavior but it can be disabled by setting useInsertSelect = "N" in the config.xml file.

Changed reset to not delete central transactions to remote sites

Change the Reset Site function so that in the central database, only the transactions from the remote site are deleted, not the transactions sent to the remote site. This allows the remote site to start over sending transaction id 1 again without causing a duplicate key error at central but preserves the outgoing transactions needed to roll back open and future effective date transactions to the remote site.

Use wizard to select packages and sites

Changed system to use a wizard when opening forms or screen editors so that users select the package on one screen, then click 'Next', then select the sites on a separate screen. This makes it easier to users to remember to select the package before selecting the sites and reduces crowding on the site selection screen.

Added package-level effective dates

Added an option to the system to set the effective dates when creating and committing packages instead of every time the user selects sites. This makes it possible for the user to update the effective date, termination date, and force flag when committing packages if desired and saves them from having to select effective dates each time they open a form and select an existing package. When selecting packages, if an effective date, termination date, or force flag is set, the dates or flags will be displayed after the package name.

Changed LDAP Test Button Label

Changed button label on Test LDAP Settings from 'OK' to 'Save Settings' to avoid confusion by users who don't have their LDAP settings working correctly yet. If users click 'Save Settings' the system will be configured to use LDAP for user authentication. The manually change the system back to using the built-in security, set useLDAP = "N" in config.xml and restart Tomcat.

Resolved Defects

The following defects were resolved in this release.

Fixed Login screen lockup

Fixed login screen so that when user clicks refresh in the browser, she can still enter a user ID and password and log in successfully.

Fixed Execute Command

Fixed Execute Command menu option to work correctly without displaying a class not found error.

Fixed IRIS Menu Editor to work at remote sites

Fixed the IRIS Menu Editor so that when used at a remote site, adding, updating, or deleting buttons or menus saves the users changes properly and sends the correct transactions to central.

Background process viewer expires tasks

Fixed background process viewer so that tasks are removed from the list 30 seconds after the task is complete even if the user closes the browser before the task completes.

Background process viewer shows correct date and time

Fixed background process viewer to show the correct month and show the time in the local format and to show the correct elapsed time and show time remaining as 0 for tasks that are complete.

Fixed transaction report filters

Fixed transaction report selection filters so that only selected packages are included in the report and so that when switching between the list of tables and the list of packages, each list is displayed correctly.

EDM 6.5 Release Notes

Enhancements

Added 'Notify Users' option

Added 'Notify Users' option to the 'Setup' menu that enables users to send a message to all the users that are currently using EDM. For example, the administrator may send a message to all users that the server is going to be restarted. The menu option uses the new 'notifyClients' function.

The following example shows a menu item with a predefined message warning users that the server will be restarted in 10 minutes followed by another warning 5 minutes later:

```
Name: Send 10 Minute Server Restart Notice
Action: notifyClients( "Notice", "Server restart in 10 minutes.", 0 ) +
notifyClients("Warning", "Server restart in 5 minutes", 5 )
```

Added 'notifyClients' function

notifyClients - Send a notification to connected clients.

Syntax	notifyClients('title', 'message', 'delayMinutes')
Parameters	title - Title of message to display to client (blank or ? means prompt user). message - Message to display to client (blank or ? means prompt user). delayMinutes - How many minutes to delay before sending message to connected clients.
Returns	Void.

Added package-level effective dates

Changed system to support package-level effective date, termination date, and force flag. When users create a new package they will be prompted for the effective and termination date and force flag for the package. When the user selects an existing package she will not be prompted for effective and termination date or force flag. When the user commits a package, the package effective and termination date and force flag will be displayed and the user can change them before committing if needed. To turn this behavior off and prompt users for effective dates and force flag every time, set the usePackageDates="N" to the config.xml file.

Added IRIS Price Change report

Added 'Run Price Change Report' to the 'Transactions' menu which creates a report of all the changes to the IRIS price table and displays them in a simple format that includes item names. Users can filter the report by transaction dates, packages, sites, transaction types and transaction status.

Save new package defaults

Modified the location selection screen so that when users create a new package and select their sites and effective/termination dates and force flag, the new package is saved with their selections. When a user selects a package instead of creating a new one, the default sites, dates, and force flag that were saved with the package will be pre-selected. After selecting a package, if the user modifies the sites, dates, or force flag, the new selections may be saved as the defaults for the package by clicking the 'Save as Package Defaults' button.

Simplified Transaction Report

Changed the transaction report to only group by 'Sites' instead of 'From Sites' and 'To Sites'. The new report enables users to select sites and then creates a report of the transactions that have been sent to or received from those sites. Internal transactions sent from the central site to

itself, such as transactions to restore a value in the central database when a termination date arrives, are included in the report only if all sites are selected.

Support Sybase database

Added support for Sybase 9 at both remote and central sites.

Resolved Defects

Fixed SQL Server connection recovery

Fixed automatic recovery of lost SQL Server connections to properly detect new error message formats and recover the connection when it is lost due to a temporary network error or a restart of the SQL Server service.

Fixed dialogs going behind browser

Changed dialog processing so that when a dialog is opened and the user clicks something on the browser, the dialog remains in front of the browser. This solves the problem of a dialog going behind the browser without the user noticing it making the system appear to be locked up.

EDM 7.0 Release Notes

This page provides access to the EDM 7.0 release notes. The release notes are sorted by version number in descending order. The release notes for the latest version of EDM are provided at the top of the list.

EDM 7.9 Release Notes

This release was tested with Java version 1.8 update 51.

The following changes were made in this release.

File Sharing

Users can now request files from the remote sites and create file share groups that enforce consistency with the remote files. By default, the new functionality can be found in the Sites menu's 'Edit File Sharing' item, or a new item can be added with the editFileSharing() action.

This functionality can be combined with the editServerTextFile() function to centrally edit and control remote files.

Click [here](#) for more information.

Installation Program Fixes

Various problems with the EDM installation program were fixed.

IRIS Menu Editor

- The *irisItemImageDir* variable in the [config.xml](#) file was corrected in the default IRIS installation.
- A bug was fixed in the Menu Editor that could cause it to freeze.
- The menu type items are now visible within the menu selection screen of the Menu Editor.

EDM 7.8.100 Release Notes

EDM Versions prior to 7.8.100 have an expired code signing certificate. You should upgrade to 7.8.100 as soon as possible.

This release was tested with Java version 1.8 update 40.

Enhancements

Updated EDM Signing Certificate

The EDM Signing certificate was updated to SHA256 with RSA security and is valid through March 2018.

Implemented an editor for IRIS Config Groups

The IRIS Config Groups editor is used to manage and edit IRIS INI files from a central location.

See [editIrisConfigGroups](#) function documentation for details.

Implemented a server text file editor

The [Server Text file Editor](#) allows the end user to select a text file on the server, edit it, and then update it on the server.

See [editServerTextFile](#) function documentation for details.

Resolved Defects

IRIS Tax Rules Editor fixes

The state of the "Show Detail" checkbox is now restored after editing a tax rule using the [Tax Rules Editor](#).

Fixed priority and sequence # assignment to new tax rules and to existing tax rules added as a top level tax rule. They now match the numbering of the original tax rules editor.

EDM 7.8.30 Release Notes

The following changes were made in this release.

IRIS Tax Rules Editor fixes

Creating new conditional tax rules was fixed.

EDM 7.8.27 Release Notes

The following changes were made in this release.

DeleteRowCmd

Added function: deleteRow

Delete a row or rows from a table for a given list of sites that match the given columns and values in the table with the given name. For example, when a form uses `addRow()` to add a row in a table that is not part of the form, it should use `deleteRow()` to remove the row from the other table when the form's data is deleted by the user.

```
deleteRow( 'IRIS_dbo_Table', getNamedControlValues( 'Id' ), getSelectedSites() )
```

See [deleteRow](#) function documentation for details.

Effective Date Fix

The effective date was still being set when inserting new transactions into the audit trail. This has been fixed.

EDM 7.8.24 Release Notes

The following changes were made in this release.

Fixes to the IRIS Tax Rules Editor

Editing a Tax Rule Condition no longer changes the underlying Tax Rule record's Active state. The sequence numbers of the Tax Rule Conditions are now set properly when adding an existing Tax Rule as a condition. Fixed interaction with the drop down controls when using both the keyboard and the mouse. Changed some of the text on the labels in the Tax Rule Condition form to better reflect that a condition is being edited.

Fixed a NullPointerException when opening certain forms when they were already open

Changed web services to use new deploy date in audit trail

The web services that get changes since a certain date (audit/changes) and that get a list of tables that have changed since a certain date (audit/changedTables) have been changed to use the new DeployDate column in the audit table to determine if a change has been deployed since a certain date. The deploy date is null for open transactions. When a transaction is committed, the deploy date is set to either the current date/time or the transaction's effective date/time, whichever is greater. This resolves the issue that occurred when users created a transaction one day then committed it the next day. When requesting changes at the end of the first day the transaction was ignored because it was not committed. When request changes at the end of the second day, the transaction was being incorrectly ignored because its effective date was earlier than the date of the previous request for changes.

The effective date is not updated when transactions are committed, only the deploy date is updated.

Fixed IRIS Menu Editor "Menu Bar" option

Fixed the Menu Bar option on the menu displayed when a user right-clicks on the IRIS Menu Editor so that it hides and displays the property window correctly.

Elstar Menu Editor

- Fixed a bug when selecting a different menu from within the menu button editor.
 - If the user had previously filtered the list of available menus, and chose a menu that was not in the matching list of menus, an exception would occur.
 - This fix also applies to the IRIS and Aloha menu editors.
- Fixed control alignment issues in button properties editor.
- The Category can now be selected via the keyboard and no longer shows the category ID.
- A reference to the Item table was fixed for Merchandise items. It was using the Elstar database, but should have been using the MenuMgr database.
- The prices listed on the button and in the table on the Prices tab have been ordered by the iPriceDescID column.

EDM 7.8 Release Notes

This release was tested with Java version 1.8 update 45.

The following changes were made in this release.

IRIS Menu Editor

- A "Multi Touch Button" group was added to the Appearance tab of the Menu Button properties window. The user can set the width of the Text Area and the Quantity Area of the button.
- A new "Use Configured Obj Pos" checkbox was added to the Image tab.
- A new preview of the button was added to the appearance tab.
- A "Button bounds" group panel was added to the property panel to the left of the menu panel (top / left / width / height). The values in the button bounds area are updated even when the user changes the values in the menu button's Property Dialog, and it updates in real time as you move the button. It can be used to precisely position the button in the menu.
- The menu properties panel was enhanced to include the ability to set the background image's location and size.
- The menu properties panel was enhanced to include the ability to set the WAV file that plays when the menu is opened.

SQL Server

Added support for SQL Server-specific TimeStamp column types.

Dinerware POS

Added support for the Dinerware POS.

PCAmerica POS

Added support for the PCAmerica POS.

EDM 7.7 Release Notes

The following changes were made in this release.

Added a Tax Rules Editor for IRIS

A new Tax Rules Editor has been created to manage taxes for IRIS. On new installations the editor can be opened by clicking POS - Tax Rules. If it is not on your menu, you can add it by creating a menu item where the action is 'editIrisTaxRules()'.

Improved performance of table refresh and current data web service

Improved the performance of rolling back transactions when sending a table refresh or when getting the current data values in response to a web

service call.

Added user ID when logging SQL statements

When the system is set to FINER logging to log all SQL statements, the ID of the user that initiated the query is now logged. This was done to help identify users who need training to avoid unknowingly running queries that adversely affect performance. For example, opening the CDMAudit table by selecting Application - Tables - CDMAudit then clicking a column heading to sort the data can cause performance to suffer if there are millions of rows in the audit table.

Added JDBC data source for MS Access databases

Users can now connect to MS Access databases using the JDBC-ODBC bridge instead of the older ODBC connection which is more flexible and works with both 32 and 64-bit operating systems. To use the JDBC connection to Access set the data source type in config.xml to ACCESS_JDBC and set the connect string like this example:

```
<systemDataSource
  name = "EDMWeb"
  version = "1"
  type = "ACCESS_JDBC"
  connect = "jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ=C:\EDMWeb\EDMWeb.mdb"
  database = "C:\EDMWeb\EDMWeb.mdb"
  server = "localhost"/>
```

Adjust UI to support Java 8

Made some minor adjustments to the user interface to support changes in Java 8 while remaining compatible with Java 7. When running Java 8 in the browser, the "Find by" combo boxes on forms did not move to the selected row when the enter key was pressed, right-clicking a button in the menu editor did not display the popup menu, and moving and sizing menu buttons did not work properly.

EDM 7.6 Release Notes

EDM Versions 7.6 and earlier do not function well with Java 1.8. Upgrade to the latest version of EDM for the best compatibility with Java 1.8

Resolved Defects

Fixed retry timeout when adding JMS receive listener

Fixed retry timeout when an error occurs in the JMS receive listener so that it resets to 0 after a successful connection instead of using the next higher timeout value. Previously, when a retry failed the timeout was doubled to avoid sending to many connection requests but when a retry succeeded the retry timeout was not reset to 0 so the next failure was waiting too long before retrying. Make value lock IDs company-specific

Changed registering lock ID's in the CDMValue table to be company-specific so that, if there are multiple companies on the server, each company registers its own lock ID's and removes them if they exist at startup. This fixes the problem of a lock row being left in the ValueDao table in the second company that is not deleted during during startup by the lock ID registration process and causing errors when the system tries to get the lock.

Fixed SendEvaluateTransaction function's site and package selection

Fixed the SendEvaluateTransaction to prompt for the sites and package only if the sites and package were not specified in the function parameters.

Fixed sendSynchronizeDirectoryTransaction function's site and package selection

Fixed the sendSynchronizeDirectoryTransaction to prompt for the sites and package only if the sites and package were not specified in the function parameters.

Fixed problem with setting values in the Value Popup window.

Fixed a problem with the Value Popup window causing the setting to not be saved. It affects both the standard forms and the Micros button editor which uses the Value Popup window for prices.

Change subforms to save properly in all cases

Fixed subforms when a single site is selected so that when the user clicks on a field to edit it the data will be saved properly. This will correct the problem seen on the Item Master form's Combo tab where entering combo information into the sub-form then entering components in the sub-sub-form did not save the changes.

Enhancements

Changed transaction web services to use transaction date if it is greater than the effective date

Changed the web services that get changes and that get changed tables since a specific date to use the transaction date instead of the effective date if the effective date is null or is less than the transaction date.

Changed system initialization to log a warning when onStart expression causes an error

Change the system initialization routine to just log a warning when an error occurs while processing an onStart expression for one of the companies. This will allow the system to finish its initialization process and be available for users instead of not allowing users to access the system until the onStart expression error is corrected.

Changed addProcessReceivedTransactionsListener to log processing errors

Change the addProcessReceivedTransactionsListener function so that if errors occur during the initial transaction processing, they are logged but the error does not prevent the listener from being added.

Changed transaction loader to ignore columns that don't exist

Changed the process that loads transactions from the audit trail to log a warning and ignore any columns that are not in the table schema. This will enable the transaction processing to proceed instead of preventing it from completing altogether.

Enabled central to receive transactions from sites that have not been added yet

Transactions will now be received when they arrive in ActiveMQ and left in the incoming directory even if the site ID has not been added yet. Once the site is added, transactions from the site may be processed. When transactions are received from a site whose ID is not recognized, a warning will be added to the log. This is secure because the remote site has to know the user and password to send messages to the queue and the transactions are not processed, just saved in the incoming directory.

Log start and end of web service calls

Added logging to show when web service calls start and end. Users can see what web services, scheduled tasks, and background jobs are currently running using the View Server Processes menu option.

Update effective date on commit

When transactions are committed, if the effective date is null or less than the current date/time it is updated to the current date/time. This enables web service calls that get the changes since a specific date/time to get the transaction even if the transaction was created before the specific date/time. If the effective date is in the future it is not modified.

Increased size of IRIS Menu Editor's 'Find Item' dialog

Increased the size of the 'Find Item' dialog so that in browsers where the users have increased the font size the fields are not wrapped to the next line or hidden. The dialog has also been made resizable.

New Functions

Function Name	Description	Example
setConfigProperty	The setConfigProperty function enables callers to set the value of a property in the config.xml file. See setConfigProperty function documentation for details.	To set the awsCredentials property to the encrypted AWS credentials call: <pre>setConfigProperty('awsCredentials', 'myencryptedawscredentials')</pre>
fileExists	The fileExists function returns true if the given file or directory name exists. See fileExists function documentation for details.	<pre>fileExists('myfile.txt')</pre>

addTransactionsProcessedListener	<p>The addTransactionsProcessedListener function evaluates an expression whenever transaction processing completes. This function is normally called in the onStartup property of the configuration to cause the system to do something every time transactions are processed.</p> <p>See addTransactionsProcessedListener function documentation for details.</p>	<p>To open Notepad after transaction processing is complete call:</p> <pre>addTransactionsProcessedListener('runApp('notepad.exe', false)')</pre>
removeTransactionsProcessedListener	<p>The removeTransactionsProcessedListener function removes a listener that was previously added with the addTransactionsProcessedListener function and the given expression.</p> <p>See removeTransactionsProcessedListener function documentation for details.</p>	<p>To remove a listener that opens Notepad after transaction processing is complete call:</p> <pre>removeTransactionsProcessedListener('runApp('notepad.exe', false)')</pre>
getConfigProperty	<p>The getConfigProperty function enables users to get the value of a property in the config.xml file. This can be used to pass values from config.xml to functions.</p> <p>See getConfigProperty function documentation for details.</p>	<p>If the encrypted Amazon AWS credentials are in the config.xml in a property called awsCredentials then you can call the Amazon transaction file transfer function:</p> <pre>receiveTransactionFilesFromAmazon('*', 'mybucket', 'incoming', getConfigProperty('awsCredentials'))</pre>
getConfigPropertyWithDefault	<p>The getConfigPropertyWithDefault function enables users to get the value of a property in the config.xml file or a default value if the property is not found. This can be used to pass values from config.xml to functions.</p> <p>See getConfigPropertyWithDefault function documentation for details.</p>	<p>If the encrypted Amazon AWS credentials are in the config.xml in a property called awsCredentials then you can call the Amazon transaction file transfer function:</p> <pre>receiveTransactionFilesFromAmazon('*', 'mybucket', 'incoming', getConfigPropertyWithDefault('awsCredentials', 'defaultCredentials'))</pre>
clearSessionPackage	<p>The clearSessionPackage function removes the selected session package on the client to users have to select a package when making changes.</p> <p>See clearSessionPackage function documentation for details.</p>	<pre>clearSessionPackage()</pre>
publishXCEDDataTables	<p>The publishXCEDDataTables publishes current data values with open and future transactions rolled back into CSV files. The function creates one export file per table and uses share group numbers instead of site numbers when data is shared. The share group data is only included in the file once and the individual sites in the share group are not included. The data for the lowest site number in the share group is used for the share group data.</p> <p>See publishXCEDDataTables function documentation for details.</p>	<pre>publishXCEDDataTables('', '', 'PublishedData')</pre>

publishXCEDDataTableList	<p>The publishXCEDDataTableList function works just like the publishXCEDDataTables function except that instead of passing in a list of table names, the name of a table list is passed in and the function gets the table names from the snapshot file.</p> <p>See publishXCEDDataTableList function documentation for details.</p>	<pre>publishXCEDDataTableList('', '', 'PublishedData')</pre>
--------------------------	--	--

EDM 7.5 Release Notes

The following changes were made in this release.

Enhancement

Improved Performance of changedTables Web Service

Reduced the time required to get a response from the changedTables web service by a factor of 100.

EDM 7.4 Release Notes

The following changes were made in this release.

Enhancement

Improve the form view function to support system tables

The "view" function can now be used to open forms that have a system table, such as the site table, as their source. This provides users with a better row selection screen including full filtering and sorting capability as well as eliminating the multiple "Find By ..." combo boxes used to select rows on forms opened with the "open" function. The "view" function has better performance than the "open" function due to eliminating the need for multiple "Find By ..." combo boxes whose queries are executed separately. The "view" function is also more scalable also because it uses newer data access methods that do not keep database queries open on the server while the user is using the form.

A new version of the site form is available to use with the view function by adding the following action to the menu:

```
view( 'Form', 'CDMLocationsAndGroups View' )
```

Resolved Defect

Fixed slow form performance on forms with SQL joins

Corrected the SQL generation of queries contains joins to eliminate redundant join processing which was causing slow performance for forms containing queries with joins especially on databases with a large number of sites.

EDM 7.3.86 Release Notes

The following changes were made in this release.

Enhancements

Added automatic transaction recovery

If any transaction files are missing when the remote site receives updates, the missing transactions are automatically recovered from central. A new configuration option "processUnexpectedFiles" was added to control this functionality. It is enabled by default.

Added function: sendSynchronizeDirectoryTransaction

The "sendSynchronizeDirectoryTransaction" function synchronizes an entire directory between the central location and a remote location. Either the remote directory can be updated to match the central directory or the central directory can be updated to match the remote directory. It is optional to include sub-directories and to delete files in the destination directory if they are missing in the source directory.

See [sendSynchronizeDirectoryTransaction](#) for details.

Added web service: changedTables

The "changedTables" web service returns a list of tables that have been updated since a given date along with the sites that were affected by the updates.

Example

http://myserver/EDM/rest/audit/changedTables?sites=12&table=IRIS_dbo_tbl_ItemMaster&since=2013-08-15%2016:33:45.321

The "sites" parameter is a comma-separated list of site IDs to include in list of changed tables ("*" means all sites)

The "table" parameter is the table name for to check for changes (null or blank means all tables).

The "since" parameter is the earliest effective date of transactions to be include.

Use checkboxes to update IRIS Monitor and Printer Routing

Updated Item Master forms so that users can simply check the box next to each monitor or printer to which the item should be routed instead of selecting from a list of all possible combinations.

Added plug-in for Nucleus POS

Added plug-in functions for the Nucleus POS system to enable editing House, Vendor, and User accounts and to transfer report files to central for central access to reports from remote sites.

EDM 7.3.36 Release Notes

The following changes were made in this release.

Resolved Defects

Fixed effective date error

Changed package selection screen to not display a message telling the user that an effective date must be entered when there is no effective date field on the screen where the user can enter the date. This resolves the issue for users that require effective dates and don't use the default package but want to send a refresh which does not prompt for the effective date.

Fixed an error in the IRIS Menu Editor regarding menu button arrangement

When the IRIS Availability Editor is used to activate buttons, the button that is activated may have a sequence number that is higher than the number of buttons on the menu. This was causing an error because the sequence number of the button was used in the calculation of the Z order of the button on the menu. The sequence number is no longer used, so the error has been eliminated.

Transaction viewer "Mark as fixed" button

The transaction viewer no longer clears the transaction list when the "Mark as fixed" button is clicked.

Enhancements

EDM Client Applet requires all permissions

Oracle changed the requirements for the "accessEventQueue" permission, which is used by EDM's client UI. The change to this permission will cause a sandboxed applet to fail with an "access denied" error. Due to this change, we have changed EDM's client applet to request all permissions, which is the only other available security setting. EDM's philosophy of not altering anything on the client computer has not changed, however. It will not write anything to the client computer's filesystem or registry.

Optimized the Select Sites dialog

The dialog used to select sites in the Routing Wizard, the Transaction Report Dialog, among others has been optimized to work faster with systems that have many thousands of sites.

Allow joins in CDMRecordSelector row source

Fixed the form viewer so that if the CDMRecordSelector has a join in it the form operates correctly. This is useful when a subform needs to be sorted by a name field that coming from another table such as in the Xpress subform that edits the SecurityGroupMap table where the CDMRecordSelector is "SELECT b.name, a.authId, a.securityGroupId FROM TNG_dbo_SecurityGroupMap a, TNG_dbo_SecurityGroups b WHERE a.securityGroupId=b.securityGroupId order by b.name".

Default column value now defaults to NULL

The database schema reader was changed so that if no default value was specified for a column, the default column value is set to NULL instead of a blank string.

This addresses issues like the XPRESS menu editor where after starting up a new remote system and adding a button to the menu editor was generating foreign key violations due to '0' being used for Popup Panel IDs.

EDM 7.2 Release Notes

The following changes were made in this release.

Resolved Defects

Copy record fix

When opening up a menu editor or a form using the view() function, copying a row and then editing a value in the row that appears in the selection list, the value would not be updated the selection list after saving the row and returning to the list.

Fixed the cancel button in the Mobile Item Sequence screen

A problem was fixed where the Mobile Item Sequence screen would close when the user pressed the cancel button. It now returns to the previous screen.

Fixed a problem when requesting a Table List from a site and "Require Effective Date" was set to true

When requireEffectiveDate was set to "Y", requesting a Table List from a site would display an error message. This has been fixed.

Fixed display of versioned sites

A problem was fixed with the UI that displays sites to the user when there are multiple versions of a record at the sites in the package.

Fixed saving records when closing the form by clicking the "X" at the top of the tab

Clicking the "X" at the top of the tab prompts the user to save any changes that were made. Clicking the "YES" button will now correctly save the changes as expected.

Fixed task scheduling so that task is rescheduled on error

Change the task scheduling procedure to check for errors when scheduling the task and, if an error occurs, reschedule the next execution of the task anyway. This resolves the issue of getting an error while scheduling a periodic process, such as a daily transaction file transfer, and not rescheduling the next execution of the task.

Fixed NullPointerException that caused 404 error on login

Fixed the remote table load function so that if the cache of remote tables needs to be reloaded and a user browses to the server, the remote tables are loaded without throwing a NullPointerException because the user is not logged in yet and returning a 404 error to the user's browser.

Fixed JMS commit listener to handle new sites

Change the commit listener that sends committed transaction files to ActiveMQ so that if it is listening for all sites ("*") and a new site is added via the site form, transactions for the new site will be sent to ActiveMQ without restarting Tomcat or EDM.

Changed reset transaction to not delete .FIL files

After a remote site receives a reset transaction and any transactions sent after the reset, it now deletes the remaining .XML files from the incoming directory but leaves any .FIL files. This corrects the problem where users would send a reset followed by a refresh or other transaction that comes with a .FIL to a site and the .FIL was deleted before the transaction was processed.

Enhancements

Sort packages alphabetically when committing

Changed the commit wizard to sort the list of packages by name to make it easier for users to find the package they want to commit.

Check for committable packages in background thread

The startup process was changed to check for committable packages in a background thread so that users don't have to wait for that server call to complete before using the system when they first log in. When the call is complete, the Commit button will flash if there are packages to commit.

Added 'About' to menu item context menu

To see the system name, version, and copyright info users can now right click on any menu item and select 'About'.

Optimized the loading of Master Data Management records

The logic for limiting the records shown to the user was moved from the Application Layer to the Database Layer.

Reduce scrolling on Elstar button properties window

Changed the layout of the check boxes on the Elstar button properties window to use 3 columns and be closer together vertically so that users can edit them with less scrolling.

Saving records in the form viewer is now an atomic operation

Records in the form viewer are all saved as a single unit. If any of the records fails to save, the records that did save are rolled back so that the user can fix the problem(s) and save them again.

Changed difference report to put quotes around values

In order to help users see differences in white space between column values, difference reports now put quotes around all values so a value like Kansas is shown as "Kansas" and a value consisting of three spaces is shown as " ".

EDMclient package reorganized

New sub packages were created in the EDMclient package to help better organize our client related classes.

Improved automatic database reconnection

Change database reconnection to support additional errors, such as "Connection timed out", which trigger the system to automatically try to reconnect to the database. Also made it possible for users to add additional errors by adding them to config.xml with property names made up of a prefix and a sequence number beginning with 0. The messages are not case-sensitive and if a database operation returns an error that contains one of the messages then the database connection will be retried.

For example to add two new errors to cause SQL Server to automatically attempt to reconnect, add properties like the following:

```
sqlServerRetryableConnectError0 = "Connection reset "  
sqlServerRetryableConnectError1 = "Software caused connection abort "
```

Oracle servers use prefixes like the following:

```
oracleRetryableConnectError0 = "Connection reset "  
oracleRetryableConnectError1 = "Software caused connection abort "
```

If your system has both SQL Server and Oracle data sources, you can add messages for each one by using the appropriate prefix.

A new "After Save" event was added to EDM Forms

EDM Form Designers can now include calls to EDM functions in a new "After Save" event that is triggered once a record (and its sub records) are saved. This event will not be triggered if there was an error while saving.

Added new Taco Bell BLIMP specific plugin

Taco Bell BLIMP specific functions and commands are now located in EDMclient.plugin.tacobell.ClientPlugins. The config file for Taco Bell will need to load this plugin.

displayMModSeq Plugin to Support Mobile Modifier Display Sequence

This plugin is used in store environments where customers place orders using the mobile phone.

The *displayMModSeq* plugin enables the user to drag-and-drop existing records on the *Mobile Modifier Display Sequence* form to set the desired sequence for the records. This action updates the *sequenceID* field of each record in the *MobileAddOn* table. Records in the *MobileAddOn* table are sorted in ascending order by the *sequenceID* field.

The plugin requires the user to select sites, a package, and if there are differences in the data, the desired version of the data to edit.

EditMobileCategoryItemSequence Plugin for Mobile Orders

This plugin is used in store environments where customers place orders using the mobile phone.

The *EditMobileCategoryItemSequence* plugin enables the user to drag-and-drop existing records on the *Mobile Menu Item* form to set the desired sequence for the records. This action updates the *sequenceID* field of each record in the *MobileCategoryItems* table. Records in the *MobileCategoryItems* table are sorted in ascending order by the *sequenceID* field.

The plugin requires the user to select sites, a package, and if there are differences in the data, the desired version of the data to edit.

The user is presented with a list of mobile categories. The mobile categories list is populated with records from the *MobileCategory* table where the *usedForMobile* field value is *True*.

BLIMP Mobile Categories can be resequenced

A new function was added that will resequence Mobile Categories. If no Mobile Items are assigned to a Mobile Category, the category's sequence will be set to NULL and the other Mobile Categories will be resequenced so that no gaps in the sequence order will exist.

Note

If Share Groups are used, Mobile Items and their Mobile Categories must all reside in the same group or the resequencing operation will fail.

See [resequenceMobileCategories](#) for details.

The "Copy Record" button has been removed from the Mobile Modifier Sequence screen

The "Copy Record" button was removed from the Mobile Modifier Sequence screen as the screen is meant only to manage the sequence of the records in the table, not to add/update/delete mobile modifier records.

Updated splash screen

Updated splash screen with latest logo.

New login screen

EDM has a new login screen that features a background image and version information as well as the username and password text fields.

This screen is brandable via new attributes in the Config node in the [Config.xml](#) file.

The following table describes the new Config node attributes.

Config Node Attribute	Description
loginBackgroundImage	Name of the image file to display in the login screen
loginPanelX	X location of the login panel
loginPanelY	Y location of the login panel
versionPanelX	X location of the version panel
versionPanelY	Y location of the version panel

EDM will use default values if these entries are omitted from the [Config.xml](#) file.

Since the image file will be streamed to the client from the server, the size of the image will affect how quickly the client displays the login screen.

The default login image is provided in the EDMclient.jar, which is cached on the client after the initial load of the applet.

Process Transactions now only reports transactions processed for sites that the current user has access to

The message box that appears after processing transactions no longer contains all processed transactions, but only the transactions processed for the sites that the currently logged in user has access to.

SQL Server 2012

EDM now supports SQL Server 2012.

New Functions

getSelectedSites()

This function returns a comma-delimited list of site IDs that the user currently has selected in the active form.

getServerTime

The `getServerTime` function returns the current system time on the server in milliseconds since 1/1/1970 UTC. This can be used to synchronize system times.

See [Client Functions - getServerTime](#) and [Server Functions - getServerTime](#) for details.

getServerCallsPerSecond

The `getServerCallsPerSecond` function returns the average number of calls per second that can be made to the server after sending as many calls to the server as possible for 5 seconds.

See [Client Functions - getServerCallsPerSecond](#) and [Server Functions - getServerCallsPerSecond](#) for details.

synchronizeWithOpenText

The `synchronizeWithOpenText` function calls a special API provided by OpenText to compare the item, category, category item, and modifier data in OpenText with the existing data in the tables: MDM_dbo_MobileItemTable, MDM_dbo_MobileCategory, MDM_dbo_MobileCategoryItem, and MDM_dbo_MobileModifierGroup. If the data exists in OpenText but not in the table it is added to the table. If the data exists in the table but does not exist in OpenText it is deleted from the table. If the data exists in both OpenText and the table but some of the column values are different, the data in the table is updated to match the OpenText table.

This function can be called from the menu or from a scheduled task that runs periodically. An example of the call is shown below.

```
synchronizeWithOpenText( 'https://myserver/opentext/api', 'user',  
'password', '777777' )
```

addAmazonCommitListener

The `addAmazonCommitListener` function adds a system listener that automatically sends transaction files to Amazon S3 as soon as they are committed. This enables users to send files to Amazon immediately instead of waiting for a scheduled transfer. The function can be called in the `onStartup` property of `Config.xml` as shown below:

```
xxx  
onStartup = "addAmazonCommitListener( '*', 'com.xpient.EDM.transactions',  
'outgoing', getPropertyFileValue( 'credentials', 'aws.properties' ) )"
```

See [Client Functions - addAmazonCommitListener](#) and [Server Functions - addAmazonCommitListener](#) for details.

EDM 7.1.12 Release Notes

The following changes were made in this release.

Optionally log company names

Some servers have multiple companies defined in their `Config.xml` so to make it easier for users to know which company generated a particular log message, users can now set a flag in `Config.xml` to tell the system to include the company name in every log message.

To include the company name in log messages, change the root element in `Config.xml`

From this: `<config version = "4">`

To this: `<config version = "4" logCompanyName = "Y">`

Fixed session packages to keep separate site lists for each form

Changed the session package processing so that when a user has a session package selected and he opens one form and selects a list of sites then, while the first form is still open, he opens another form and selects a different list of sites, each form remembers the sites that were selected for it.

Added function: updateTransferSettings

The `updateTransferSettings` function updates the transfer host, user ID, password, and company name on the central site record. This information is used when connecting to the central system to transfer transaction files. To have the information automatically saved when the system starts up, the function can be called in the `onStartupProperty` of `Config.xml` as shown in the following example.


```
onStartup = "updateTransferSettings( 'localhost:8080/EDM', 'remote',  
'remote', 'HQ' )"
```

See [updateTransferSettings](#) for details.

Added function: createRemotelInstaller

The createRemotelInstaller function creates an installation program that can be downloaded and used to install EDM at a remote site with pre-configured settings for the current company.

See [createRemotelInstaller](#) for details.

Fixed SQL Server driver to always turn IDENTITY_INSERT property off

Change the SQL Server driver so that the IDENTITY_INSERT property is always turned off for a table even if the SQL INSERT statement fails. This resolves the problem of transactions failing because they can't insert into a table with an identity column because the IDENTITY_INSERT property is not on.

EDM 7.1.11 Release Notes

The following changes were made in this release.

Bit Editor now supports hard coded bit text

The form designer now allows the entry of "column headings" for bit editor controls. These headings are separated by semicolons and will be displayed next to the checkbox that represent the bit. Clicking the info icon in the attributes tool in the form designer will show a message box that contains the informative text.

Fixed form "save" function when only one site is selected

Fixed the form processing so that forms that do not have CDMRecordSelector controls still save changes properly when the user only selects a single site.

EDM would not timeout

Fixed a situation where EDM would not timeout when the sessionTimeoutMinutes config file option was specified.

EDM would not enforce required fields

Fixed a validation error when EDM would not prompt the user to enter a value into blank fields that were marked as 'required' by the form designer.

EDM would save changes even when user chose not to

Fixed a condition where EDM would sometimes save changes when the user clicked "NO" to the save prompt.

Fixed table painting

Fixed painting so that when user first browses to EDM and opens a table the rows and columns are displayed correctly.

Eliminated redundant logging of InvalidClassException errors

Changed server so that when an InvalidClassException occurs while deserializing a client request, indicating that the client browser has an old EDMclient.jar in the cache, the error is only logged once for the clients IP address.

Eliminated syntax error on quote character in search field

Changed form viewer selection screen so that the user may enter a '\' character (in the process of entering a quoted character such as '\.') without immediately getting a 'Bad Syntax' error message.

Eliminated 'Set to Null' button for those fields that require a value

Changed the field value editor pop-up such that it will provide a 'Set to Null' button only if a null value is allowed for this field.

Added "Show Locations" button to IRIS, Aloha, POSitouch and Elstar menu editors

Clicking this button will show a panel containing the locations in the package that have and do not have the selected menu. This is useful for when you want to see which locations in the package do not have any version of a selected menu without editing the menu.

Modified "Show Locations" button when the versions of a selected menu are being displayed

Clicking this button will show a panel containing the locations in the package that have the selected version of the menu and locations in the package that do not have any version of the selected menu.

Improved estimated time remaining display when copying records

Improved the status display of the copy record function so that the estimated time remaining is much more accurate on very long copying jobs.

Improved performance of copy record function

Improved performance of copy record function when very large number of transactions are being inserted in the audit trail with the bulk insert function.

The EDM Applet is now signed.

The EDM applet is now signed with a certificate issued by the [COMODO](#) Trusted Publisher. This will eliminate the warning that users receive when they connect to the EDM server that they cannot confirm who published the app.

Modified the Commit Wizard to use the full Site Selector

The Commit Wizard now uses the full site selector by default, giving the user a much better visual overview of the sites that are to be committed. A simplified site selector can be used by setting the "useSimpleSiteSelection" config property to "Y". This will indicate to EDM that simple site selection is desired in the package creation wizard as well.

Fixed StringIndexOutOfBoundsException in query designer

Fixed query designer so that it no longer get a StringIndexOutOfBoundsException error when running a query where the table name is longer than the sum of the unique index column names.

Fixed Elstar screen editor to only display "may not move" message once

Change the Elstar screen editor so that if the user tries to move a button on a screen where the Auto Sort box is checked, a message is displayed only once telling the user that they cannot move buttons when Auto Sort is enabled.

Fixed Elstar screen editor NullPointerException

Fixed error that occurred when selecting a screen after enter a value in the filter window.

Fixed Elstar screen editor price editing

Fixed NullPointerException that occurred when clicking the Edit button on the Price tab of the button properties dialog.

Improved performance getting package list

The system can now get lists of packages to select when opening forms or committing transactions much more quickly.

Added single site update button option

The update buttons in EDM forms opened using the "open()" function can optionally be displayed when editing data for a single site using the "useUpdateButtonForSingleSite" config file option. It is off by default.

Fixed Elstar screen editor "New" button

Corrected the problem that was causing "Error adding new row" to be displayed to the user when they click the "New" button on the Elstar screen editor.

Fixed "invalid syntax" error on reports

Fixed the "invalid syntax" error that was displayed when non-administrator users tried to filter reports, such as the Transaction Report, by selecting packages.

Changed Elstar screen editor to only warn users about changing shared button text once per button

Changed Elstar screen editor so that when user begins editing the text of a button that shares a merchandise record with other buttons on the same or different screens, he is only prompted once to confirm that it is ok to change the text on the shared buttons as well instead of being prompted on every keystroke.

Fixed Elstar screen editor difference report

Fixed Elstar screen editor so that when user edits a screen with multiple versions and selects two of them and clicks "Show Differences", all the differences are properly displayed.

Fixed IRIS menu editor to not change unknown button types to -1

Changed the button handling in the IRIS menu editor so that if a button is loaded from the table with an unknown button type, it is left unchanged. If the user selects the button, they will see a message that there was an error setting the button properties for the unknown button type along with the button type ID. Users can still edit the button text and position without the button type being changed. This was done so that if a new button type is introduced in IRIS, the IRIS menu editor can still be used to edit the menus with the exception of being able to select the button value for the unknown types and the button types will not be changed to -1.

Fixed functions that allow '*' for the list of sites to not display error if user can't access all sites

Updated the functions that accept a list of sites as a parameter where '*' (asterisk) means all sites so that they only affect sites for which the user has 'view' permission. This eliminates the error message displayed to users that do not have access to all sites when they execute a function with '*' (asterisk) as the site list parameter. This does not affect menu options with '*' (asterisk) as the site list parameter, which means prompt the user to select the sites, which is the normal case. It also does not affect functions called from remote sites which only have their own site and the central site in the site list.

The affected functions are:

- processTransactions
- sendRemoteFiles
- receiveTransactionFilesFromAmazon
- sendTransactionFilesToAmazon
- receiveTransactionFilesFromJMS
- sendTransactionFilesToJMS
- transferSiteFilesViaFTP
- transferLocationFilesViaFTP
- transferSiteFilesViaRAS
- transferLocationFilesViaRAS
- transferSiteFilesViaWeb
- transferLocationFilesViaWeb
- sendSiteFilesViaWeb
- receiveSiteFilesViaWeb
- sendEvaluateTransaction
- sendExecuteTransaction
- validateTables
- copyFoodCostPeriodData
- deleteFoodCostPeriodData
- exportFoodCostFiles
- sendXpressImages
- Transaction Status Viewer
- Master Data Management screens that display sites
- Report dialogs that display sites

Added function: deleteTransactionFilesFromAmazon

This function deletes transaction files (*.xml and *.fil) from the Amazon Simple Storage Service (S3). See [deleteTransactionFilesFromAmazon](#) for details.

Added server function: addDeleteAmazonTransactionsOnResetListener

This function adds a listener that deletes transaction files from the Amazon incoming and outgoing directories when a reset site transaction is committed. To enable the listener call it in the onStartUp property of config.xml as shown in the following example:

```
onStartup = "addDeleteAmazonTransactionsOnResetListener( '*',  
'com.xpiend.EDM.christransactions', 'incoming', 'outgoing',  
getPropertyFileValue( 'credentials', 'aws.properties' ) )"
```

See [addDeleteAmazonTransactionsOnResetListener](#) for details.

Added "regularExpressionFiltering" config option

Set to 'Y' if the system will perform regex searches during filter operations by default, else 'N' to perform literal searches. The default is "N".

No longer require permission to view the Application menu to use the Elstar screen editor

Changed the Elstar screen editor and other functions to not look up table and query names in the Application menu. This resolves the problem of users who do not have permission to see the Application menu getting an error when opening the Elstar screen editor.

Reduced memory usage for remote table cache

Changed the data structure used to cache the remote table records loaded from the database so that it takes up much less memory and improves the performance of cache searching operations.

Fixed Master Data Management to not build IN clause with over 1000 site id's

Changed the query that looks for a site's group ID so that the query does not use an IN() clause with more than 1000 site id's. This was causing an error in Oracle systems which do not allow more than 1000 elements in an IN clause.

EDM 7.1.10 Release Notes

The following changes were made in this release.

Resolved Defects

Fixed site reset to leave future-effective transactions in remote audit table

The site reset function has been changed so that if there are transactions that have future effective dates, when the reset is processed at the remote site, the future transactions are added to the audit trail.

Fixed "0-Default Package does not exist" error

Change package cache to not be shared across companies so that if the Default Package is deleted at one company we don't get "0-Default Package does not exist" error at other companies when the SelectSessionPackage function is called.

Fixed Copy button screens in Aloha, IRIS, and POSItouch editors

The Copy Menu/Panel buttons in Aloha, IRIS, and POSItouch were not working properly. They were supposed to prompt for the sites to copy to, but were prompting for a whole new package and were not copying the menus properly. It now just prompts for the sites to copy to, copies the menu to the selected site(s) that don't already have it, and informs the user when the copy has completed successfully.

Enhancements

Added function: `updateTacoBellMasterDataFromOpenTextData`

This function queries OpenText data via a web service provided by Taco Bell to get item and category images and other data which is then saved in the master data tables for the mobile system. The data is then made available to callers of the Menu Service API.

See `updateTacoBellMasterDataFromOpenTextData()` for details

EDM 7.1.9 Release Notes

New Functions

The following functions were added in this release:

<code>attachMasterTables</code>	The <code>attachMasterTables</code> function can be used to create a new master data table in the EDM central database by loading the table schema from another database. When the central table is created, remote table records and remote data sources will be created for the central site and every remote site so that the table may be immediately edited and all remote sites will be supported. See attachMasterTables for details.
<code>insertTableList</code>	The <code>insertTableList</code> function can be used to add a new table list to the system. See insertTableList for details.
<code>updateTableList</code>	The <code>updateTableList</code> function can be used to update an existing table list. See updateTableList for details.

deleteTableList

The deleteTableList function can be used to delete a table list from the system. See [deleteTableList](#) for details.

Fixed update error in IRIS menu editor

The IRIS menu editor would generate error code 515 when saving changes to a menu due to columns that had been added to the schema. This has been fixed.

Fixed high memory usage bug in Aloha, Elstar, and IRIS menu editors

EDM would use too much memory when switching between menus in these editors. This has been fixed.

EDM 7.1.8 Release Notes

The following changes were made in this release:

Elstar Menu Editor

Updated the Elstar menu editor to allow users to assign items to buttons even if the items have an All number of 0.

Commit Button

A new *Commit* button has been added to the EDM client user-interface. The *Commit* button replaces the prompt to commit packages when forms and menu editors are closed. The button's text will animate for a short time when there are packages to commit. Clicking the button will prompt the user to select packages and sites to commit to the selected package. The button is hidden from the user-interface if the user does not have permission to commit packages.

EDM 7.1.7 Release Notes

The following changes were made in this release:

Added exportSecurityPolicyToXml function

The exportSecurityPolicyToXml function can be used to save the users, roles, and other security objects in an XML file which can be used to load the security tables when creating a new company's system database.

See [exportSecurityPolicyToXml](#) for details.

Added new client functions for storing/retrieving cached information.

Added the following client functions:

- [setClientPropertyString\(String name, String value \)](#) - Sets a client side property to the given String value with the given name.
- [getClientPropertyString\(String name, String defaultValue \)](#) - Gets a client side String property value with the given name or defaultValue if the property has not been set.
- [setClientPropertyBoolean\(String name, Boolean value \)](#) - Sets a client side property to the given Boolean value with the given name.
- [getClientPropertyBoolean\(String name, Boolean defaultValue \)](#) - Gets a client side Boolean property value with the given name or defaultValue if the property has not been set.
- [setClientPropertyInteger\(String name, Integer value \)](#) - Sets a client side property to the given Integer value with the given name.
- [getClientPropertyInteger\(String name, Integer defaultValue \)](#) - Gets a client side Integer property value with the given name or defaultValue if the property has not been set.

These functions can be used in an EDM form to store and retrieve values without requiring the use of another control on the form. The values will be removed once the EDM session has terminated.

Added ability to edit location data from all sites in package

If you have selected multiple sites in a package and have opened an EDM form using the "[view\('FORM', 'formName' \)](#)" function, you will be prompted to select a version if differences are found between the sites in the package.

Once you have selected a version and are editing a field, you can press the "CTRL+ENTER" keys on the keyboard or click the wrench icon and select "Edit all values" from the "Edit" menu to get a popup window where the data for all sites in the package can be set for that field.

Fixed problem with transfer_web.bat hanging on exit

The transfer_web process was getting stuck at the exit after processing a reset site transaction. This has been fixed.

The screen that shows versions of a row being edited has been changed

A versions screen is displayed when a row is selected to be edited and the row has multiple versions across sites in the package. This screen has

been simplified to display the row information in the title of the screen and the list displays the locations that contain differing versions.

Fixed problem with remote form viewer updates

Fixed problem with modifier images not being updated reliably. This fixed a problem with row updates being misapplied at central from remote due to missing column information.

EDM 7.1.6 Release Notes

The following changes were made in this release:

Fixed "IndexOutOfBounds" error when renaming Elstar screens

Fixed the Elstar screen editor to correctly save the data when the user renames a screen without displaying an IndexOutOfBounds error.

Added "getVersion" function

Using the function "getVersion()" will return the current version of EDM. This function is available via the EDM web service as well.

Fixed MDM deploy functionality

Fixed MDM deploy to handle deleting rows in IRIS tables in EDM central db when master row is deleted.

Fixed problem with deleting rows in forms

When a form was set to not allow the deletion of rows, pressing the delete key on the keyboard would still delete the row. This has been fixed.

EDM 7.1.5 Release Notes

The following changes were made in this release:

Forms support assumed decimal places for display and editing

When numeric values such as prices are stored in the database as integer values but the form designer wishes to display and edit the data formatted as a number with a decimal place then set the 'Assumed decimal places' property of the control to the number of digits that should be shown to the right of the decimal point.

For example, if prices are stored as integers such as 123 and the assumed decimal places is set to 2 and the format is set to GC (general currency) then the value will be displayed as \$1.23 (in the United States) and when the user is editing the value it will be displayed as 1.23. The user can change the value by entering a new value with decimal places such as 4.56 and the value will be saved to the database as 456.

Added replaceAll function

The replaceAll function can be used in expressions to replace portions of a string. For example, to replace single quotes in a string with two single quotes before using the string in a SQL statement use the expression `replaceAll("Mom's House", "'", "''")`.

See [replaceAll](#) for details.

Fixed exception thrown when attempting to edit Elstar screen

Fixed the Elstar screen editor to no longer throw an exception when the user selects a screen to edit.

Fixed concurrent modification error in reloadCachedData function

Fixed the concurrent modification error that occurred in the reloadCachedData function if any of the companies had registered JMS connections.

Fixed JMS message listener to receive all messages

The JMS listener sometimes did not receive all the existing messages in the queue before listening for incoming messages when there were thousands of messages queued before starting the listener. This works correctly now.

Changed Xpress create transaction triggers to ignore updates that don't change any values

Changed the createTransactionTriggers function that generates an SQL script to install triggers on Xpress tables whose updates need to be sent to central so that when a row is updated but none of the column values are changed it no longer generates an update transaction in the EDM audit table. The script was also changed to get the site ID from the site table so that the same script could be used at all sites without regenerating it.

EDM 7.1.4 Release Notes

The following changes were made in this release.

Added "clientBackgroundColor" config option

The background color used in the main UI of EDM is now configurable. Valid entries are a comma separated list of RGB values or a valid Java color name.

Java color names are white, lightGray, gray, darkGray, black, red, pink, orange, yellow, green, magenta, cyan, and blue. <i>Some of these colors will look better than others.</i>

The default is "black".

Fixed combo pricing problem in Micros menu editor

The pricing data could be updated for the wrong combo items that were in more than 1 group. This has been fixed.

Added "nullValue" config option

The text displayed when a database field is NULL can now be set in config.xml. The default is "<NULL>".

Added "useSimpleSiteSelection" config option

Valid values for this option are "Y" or "N", with "N" being the default. If set to "Y", the Site Selection screen is greatly simplified. The Share and Location groups, the Site Properties and Search controls and the Save Package Defaults button are removed from the screen.

Moved executeSQL() method

The executeSQL() method was moved from ReportServer to ExecuteQueryCmdExecutor

Fixed Aloha panel editor parameter panel for Smart Item and Smart Select

The Aloha panel editor now correctly loads and saves the function parameter values for buttons with the Smart Item and Smart Select function type.

Change override refreshes to roll back open and future transactions

When a remote site reports a transaction error and the override setting for the table is for central to override remote, the refresh transaction that is sent to the remote site now rolls back open and future transactions.

Changed site reset to re-commit transactions with status "send" and "received"

Changed the resetSite command to re-commit any committed transactions plus any transactions with the status "sent" or "received" to the site to ensure the site gets them. Transactions with future effective dates will have the status "received" if the site responds that it has received the transaction but has not yet committed it.

Change transaction processing to sort by effective date

Changed the "Process Transactions" functions to sort the transactions to be processed by the effective date, if it is not null, otherwise sort by the transaction date instead of sorting by the transaction ID. This resolves the problem of sending a transaction to a site with a future effective date, then just before the effective date arrives, sending a refresh to the site. When transactions were sorted by ID, the refresh would be processed after the transaction with the effective date which had now arrived overwriting the effective change. By sorting by the effective date, if the refresh is sent before the effective date of the other transaction arrives, the refresh transaction will be processed before the transaction with the effective date and the data will have the correct values.

Support processing transactions as they are received

The system can now be configured to process transactions as soon as the transaction file is received by adding the following line to the plugin section of config.xml.

```
<serverPlugin class =  
  "EDMserver.plugin.ProcessTransactionsOnReceivePlugin" />
```

Support auto-commit of changes made in forms

The system can now be configured to automatically commit changes made in forms instead of requiring the user to commit them. To enable auto-commit, add the following attribute to config.xml.

```
autoCommit = "Y"
```

Support managing data without packages

The system can now be configured so that the users don't have to select packages when managing data. All changes will be stored in the default package. To remove the need to select packages, change the onLogin attribute in config.xml to include a call to `selectSessionPackage('0-Default Package', false)` as shown in the following example.

```
onLogin = "iif( hasPermission( 'execute', 'function', 'showSiteManager' ),
showSiteManager(), 0 ) + iif( hasPermission( 'execute', 'function',
'selectSessionPackage' ), selectSessionPackage('0-Default Package', false),
0 )"

```

Added function: `getNamedControlValues`

The `getNamedControlValues` function gets a list of name-value pairs for a given list of controls on the active form. This function is useful for getting a list of names and values used to insert rows in the database with functions such as the `addNLSRows` function.

See [getNamedControlValues](#) for details.

Added function: `getFormNamedControlValues`

The `getFormNamedControlValues` function gets a list of name-value pairs for a given list of controls on the given form. This function is useful for getting a list of names and values used to insert rows in the database with functions such as the `addNLSRows` function.

See [getFormNamedControlValues](#) for details.

Added Xpress function: `addNLSRows`

The Xpress plugin function `addNLSRows` inserts a row in the given NLS table for every languages defined in the Locales table using a list of column name-value pairs. The function automatically populates the `CDMLOCID` and `locale` columns and gets the values for the other columns from the list of name-value pairs. For example, to add rows to the `TNG_dbo_NLSHiringSources` table after the user inserts a row in the `TNG_dbo_HiringSources` table, set the `afterInsert` property of the Hiring Sources for to:

```
=addNLSRows( 'TNG_dbo_NLSHiringSources', getNamedControlValues(
'hiringId,name,description' ), getSelectedSites() )

```

See [addNLSRows](#) for details.

Add "isView" flag to tables

Added an `isView` attribute to table objects so that EDM can distinguish between database tables and views. Users can update a table entry in the schema file to include the `isView` flag if they have created a view that they wish EDM to be able to use for reading data as shown in the following example:

```
<table name = "TNG_dbo_VIEW_Price" dbName =
"[dbo].[TNG_dbo_VIEW_Price|TNG_dbo_VIEW_Price]" dsn = "EDMWebHQ" isView =
"Y">
  <col name = "CDMLOCID" desc = "Site id" type = "Number" len = "9"
required = "Y"/>
  <col name = "ID" type = "Number" len = "10" required = "Y"/>
  <col name = "PRICE" type = "Number" len = "10" scale="2" required = "Y"/>
  <key name = "PK" type = "PRIMARY">
    <seg name = "CDMLOCID"/>
    <seg name = "ID"/>
  </key>
</table>

```

Prevent copying views when copying location data

Modified the **copyLocationData** function to ignore tables with the flag **isView="Y"** since data cannot be copied to a view.

Support 'onStartup' event

Users can add an 'onStartup' property to the config.xml file to specify an expression to evaluate when the system starts. The expression may only contain server functions since it is executed by the server. For example, the following expression will start a JMS listener when the system is started.

```
onStartup = "addJMSCommitListener( '*',  
'com.xpient.EDM.outgoing.%REMOTELOCATIONID%', 'tcp://192.168.1.65:61616' )"
```

Added function: addRow

The plugin function addRow inserts a row in the given table for the selected sites using a list of column name-value pairs. The function automatically populates the **CDMLCID** column and gets the values for the other columns from the list of name-value pairs. For example, to add rows to the TNG_dbo_ProductType table after the user inserts a row in the TNG_dbo_Products table, set the **afterInsert** property of the Product form to:

```
=addRow( 'TNG_dbo_ProductTypes', getNamedControlValues(  
'Id,name,description' ), getSelectedSites() )
```

See [addRow](#) for details.

Added function: createTransactionTriggers

Creates a SQL script called EDM_Transaction_Functions.sql that creates functions in the EDM database that can be called by triggers in the remote application database to generate transactions. Also creates a script called App_Transaction_Triggers.sql to create triggers on the tables in the given table list that generate transactions in the EDM audit table on insert, update, and delete. Currently this function only supports SQL Server databases.

```
=createTransactionTriggers( 'Xpress Transaction Tables' )
```

See [createTransactionTriggers](#) for details.

Support Java Message Service (JMS) Queues for Transaction Transfer

The system now supports transferring transaction files between remote and central installations via JMS queues. JMS queues provide fast, reliable transport of transaction files between central and remote sites and JMS brokers can be clustered behind a load balancer for high volume installations. Transaction files can be transferred at specific times or can be triggered by user actions or the system can be configured to automatically send transactions as soon as they are committed and to receive transactions as soon as they are received for near-real-time data transfer. JMS offers guaranteed delivery and also guarantees that the messages are delivered in the order they were sent and will not be delivered more than once.

Added function: sendTransactionFilesToJMS

The sendTransactionFilesToJMS function will send outgoing transaction file for selected sites to a Java Message Service (JMS) queue such as ActiveMQ. Use this function when you want manual control over when transactions are sent from the outgoing folder to the queue.

See [sendTransactionFilesToJMS](#) for details.

Added function: receiveTransactionFilesFromJMS

The receiveTransactionFilesFromJMS function will receive incoming transaction file for from a Java Message Service (JMS) queue such as ActiveMQ. Use this function when you want manual control over when transactions are received from the queue to the incoming folder.

See [receiveTransactionFilesFromJMS](#) for details.

Added function: addJMSCommitListener

The addJMSCommitListener function will install a listener that automatically sends transaction files to a Java Message Service (JMS) queue as soon as they are committed. This enables systems with real-time data transfer requirements to instantly send and receive changes as soon as they occur.

See [addJMSCommitListener](#) for details.

Added function: [addJMSReceiveListener](#)

The [addJMSReceiveListener](#) function will install a listener that automatically receives transaction files from a Java Message Service (JMS) queue as soon as they arrive. This enables systems with real-time data transfer requirements to instantly send and receive changes as soon as they occur. To configure the system to immediately process the received transaction files, call the [addProcessReceivedTransactionsListener\(\)](#) function in the `onStartup` event of `config.xml`.

See [addJMSReceiveListener](#) for details.

Added Allow Insert, Allow Update, and Allow Delete form properties

Added form properties to enable the form designer to control whether users are allowed to insert, update, or delete rows by simply putting an expression in the Allow Insert, Allow Update, or Allow Delete property. If the expression is blank, then the action is allowed. If the expression evaluates to false, for example `'=false'`, then the action is not allowed.

EDM 7.1.3 Release Notes

The following changes were made in this release:

Optionally permit only a single site to be selected in a package

EDM can now optionally allow a user to select only a single site (or a single Share Group) when opening an EDM Form or a menu editor.

Just add the following setting to the `Config.xml` file:

```
allowMultiSiteEditing = "N"
```

Multi site editing is allowed by default.

Certain properties of the IRIS Menu Editor can now be limited

It is now possible to restrict which buttons and popup menu items can be seen by the user in the IRIS Menu Editor.

Just add any combination of the following settings to the `Config.xml` file:

- `limitIrisPopupOptions = "Y"`
- `limitIrisToItem = "Y"`
- `removeIrisEditButton = "Y"`
- `removeIrisCommandButton = "Y"`
- `removeIrisFindItemButton = "Y"`
- `removeIrisDeleteItemButton = "Y"`

None of these settings are active by default.

Changes to `selectSessionPackage` function

`selectSessionPackage()` now supports a second parameter, `"requireVersionedPackages"`. If it is set to `'true'`, then only packages with versions will be displayed to the user to select. If it is not set or `'false'`, then all packages will be displayed.

See [selectSessionPackage](#) for details.

Changes to when Packages are deleted

Packages are no longer deleted until they are older than the `auditHistoryDays` setting in the `Config.xml` file.

Fixed problem with the Micros Menu Button Editor's Price Editor Popup

The Menu Button Editor was always using the complete list of the current package's sites when bringing up the price editor popup. It has been changed to use the current version's sites if a version has been selected.

Added MDM function: `copyConfigurationVersionData`

The `copyConfigurationVersionData` function copies the master data for a configuration from one version of a configuration to another version. If the destination configuration version does not exist, it is created. Data for all the tables in the configuration type's shared table group is copied.

Existing data, if any, is overwritten.

See [copyConfigurationVersionData](#) for details.

Added MDM function: copyVersionData

The copyVersionData function copies the master data for all the configurations of one version to the same configurations in another version. If the destination configuration versions do not exist, they are created. Data for all the tables in the configuration type's shared table group is copied. Existing data, if any, is overwritten.

See [copyVersionData](#) for details.

Added MDM function: copySiteSubscriptions

The copySiteSubscriptions function copies the configuration subscriptions from one site to one or more other sites. The version ID's of the new subscriptions are left blank and users must schedule the version assignments for the new subscriptions.

See [copySiteSubscriptions](#) for details.

Support using IRIS Menu Editor with master tables

Master data management users can use the IRIS Menu Editor to edit master tables if they create the appropriate master tables and views and set the table name overrides in [Config.xml](#).

There will need to be master tables with the same structure as the IRIS tables that are updated by the Menu Editor:

- IRIS_dbo_tbl_MenuMaster
- IRIS_dbo_tbl_MenuBtns
- IRIS_dbo_tbl_ItemMaster - updated when a menu ID referenced by this table is changed
- IRIS_dbo_tbl_Concept - updated when a menu ID referenced by this table is changed
- IRIS_dbo_tbl_ItemMenus - updated when a menu ID referenced by this table is changed
- IRIS_dbo_tbl_ItemComponents - updated when a menu ID referenced by this table is changed
- IRIS_dbo_tbl_ItemAttachments - updated when a menu ID referenced by this table is changed

There will also need to be master tables or views with the same structure as the IRIS tables that are read, but not updated, by the Menu Editor:

- IRIS_dbo_tbl_ItemModifiers
- IRIS_dbo_tbl_MenuCmds
- IRIS_dbo_tbl_MenuMacros
- IRIS_dbo_tbl_MenuObj
- IRIS_dbo_tbl_Tare
- IRIS_dbo_tbl_MenuResolutionCurrent
- IRIS_dbo_tbl_PayType
- IRIS_dbo_tbl_ItemPeriods
- IRIS_dbo_tbl_CfgSetting
- IRIS_dbo_tbl_UpsizeType
- IRIS_dbo_tbl_Discounts
- IRIS_dbo_tbl_DiscountCoupon
- IRIS_dbo_tbl_Msg
- IRIS_dbo_tbl_Destination

To tell the system the name of the master table or view that is to be used in place of each IRIS table, add override table names to the config.xml file as shown below using your own MDM table names:

```

overrideTableName_IRIS_dbo_tbl_MenuMaster = "MDM_dbo_tbl_MenuMaster"
overrideTableName_IRIS_dbo_tbl_MenuBtns = "MDM_dbo_tbl_MenuBtns"
overrideTableName_IRIS_dbo_tbl_ItemMaster = "MDM_dbo_tbl_ItemMaster"
overrideTableName_IRIS_dbo_tbl_ItemModifiers = "MDM_dbo_tbl_ItemModifiers"
overrideTableName_IRIS_dbo_tbl_MenuCmds = "MDM_dbo_tbl_MenuCmds"
overrideTableName_IRIS_dbo_tbl_MenuMacros = "MDM_dbo_tbl_MenuMacros"
overrideTableName_IRIS_dbo_tbl_MenuObj = "MDM_dbo_tbl_MenuObj"
overrideTableName_IRIS_dbo_tbl_Tare = "MDM_dbo_tbl_Tare"
overrideTableName_IRIS_dbo_tblMenuResolutionCurrent =
"MDM_dbo_tblMenuResolutionCurrent"
overrideTableName_IRIS_dbo_tbl_PayType = "MDM_dbo_tbl_PayType"
overrideTableName_IRIS_dbo_tbl_ItemPeriods = "MDM_dbo_tbl_ItemPeriods"
overrideTableName_IRIS_dbo_tblCfgSetting = "MDM_dbo_tblCfgSetting"
overrideTableName_IRIS_dbo_tbl_UpsizeType = "MDM_dbo_tbl_UpsizeType"
overrideTableName_IRIS_dbo_tblConcept = "MDM_dbo_tblConcept"
overrideTableName_IRIS_dbo_tblDiscounts = "MDM_dbo_tblDiscounts"
overrideTableName_IRIS_dbo_tblDiscountCoupon = "MDM_dbo_tblDiscountCoupon"
overrideTableName_IRIS_dbo_tblMsg = "MDM_dbo_tblMsg"
overrideTableName_IRIS_dbo_tblDestination = "MDM_dbo_tblDestination"

```

Support using IRIS Combo and Shake Price Editor with master tables

Master data management users can use the IRIS Combo and Shake Price Editor to edit master tables if they create the appropriate master tables and views and set the table name overrides in [Config.xml](#).

There will need to be master tables with the same structure as the IRIS tables that are updated by the editor:

- IRIS_dbo_tbl_ItemPricing

There will also need to be master tables or views with the same structure as the IRIS tables that are read, but not updated, by the Menu Editor:

- IRIS_dbo_tbl_ItemMaster
- IRIS_dbo_tbl_ItemModifiers

To tell the system the name of the master table or view that is to be used in place of each IRIS table, add override table names to the config.xml file as shown below using your own MDM table names:

```

overrideTableName_IRIS_dbo_tbl_ItemMaster = "MDM_dbo_tbl_ItemMaster"
overrideTableName_IRIS_dbo_tbl_ItemModifiers = "MDM_dbo_tbl_ItemModifiers"

```

Fixed error in Smart menu builder

Fixed Smart menu builder so that it opens properly instead of throwing an error that it needs a Frame or Dialog parent window.

Fixed error in WebEDM

When using WebEDM for conversational ordering, some exported database tables did not contain data.

Prior to 7.1.3, the system checked the site ID list that gets passed to ensure that each listed site accepts transactions and has data in the table. If either condition is false, then the site is removed from the list of sites to export. If the system found a site that did not accept transactions, it removed the site from the list, but then tried to check the same site to see if it has data in the table. Since the site had been removed from the list, an error occurred.

In 7.1.3, this logic is changed to only check for data in the table if the site accepts transactions.

Updated form viewer to support selection lists with joins

The form viewer can now support CDMRecordSelector source queries that contain joins such as when the designer wants to display the name of the item from the item table when showing a list of prices from the price table.

API CHANGES:

- Moved DataSourceInfo to EDMcommon.dao package so client-side code can access it.
- Moved RemoteTable to EDMcommon.dao package so client-side code can access it.
- Added AddSiteCmd that adds a site to the cache and site table.
- Added DeleteSiteCmd that deletes a site from the cache and site table.
- Added AddRemoteTableCmd that adds a remote table to the cache and remote tables table.
- Added DeleteRemoteTableCmd that deletes a remote table from the cache and remote tables table.
- Added GetDataSourceCmd that gets data source connection information from the server configuration.

Added command-line client to call web service

Added a command-line client to call the EDM Evaluate Expression web service.

The command-line arguments are:

- **expression** Expression to evaluate. For example "processTransactions(**)".
- **userid** ID of the user making the call.
- **encryptedPassword** Encrypted password for the user from the CDMUser table.
- **company** Name of company where expression is to be evaluated.
- **wsdlUri** URL of the WSDL for the web service. For example "http://localhost:8080/EDM/service?wsdl".

Run it from the DOS command line with a batch file like the following (ensure paths are correct):

```
@C:\EDMWeb\jre1.6.0_07\bin\java.exe" -cp C:\EDMWeb\lib\system.jar
EDMserver.EvaluateExpressionWebMethodMain "processTransactions('*')"
"remote" "1b06345acc9ffc3d26e6019fc94c3696a2d286cee6a80bfe00000010" "HQ"
"http://localhost:8080/EDM/service?wsdl"
```

Or, allow the caller to pass the expression to the batch file as a parameter like this:

```
@echo off
if ""=="%1" goto usage
"C:\EDMWeb\jre1.6.0_07\bin\java.exe" -cp C:\EDMWeb\lib\system.jar
EDMserver.EvaluateExpressionWebMethodMain %1 "remote"
"1b06345acc9ffc3d26e6019fc94c3696a2d286cee6a80bfe00000010" "HQ"
"http://localhost:8080/EDM/service?wsdl"
goto end
:usage
echo Usage: web_service "expression"
:end
```

EDM 7.1.2 Release Notes

The following changes were made in this release.

Changes to the Package Selection Screen

EDM no longer appends the user and date information to user entered package names, so the screen where new packages are created has been changed to verify that the package name is unique.

The list of packages has been modified to be easier to read and display more information to the user, such as the user who created the package and the date that it was created.

Added isOnNewRow function

Added function to return true or false if the active form is on a new row.

This can be used to perform logic on a new record that you wouldn't want to perform on an existing record.

See [Functions.htm#isOnNewRow](#) isOnNewRow function documentation for details.

See [isOnNewRow](#) for details.

Fixed problems with the Micros Menu Button Editor

1. The pagination calculation code was off by 1 in some situations.
2. The left side panel has been made a little wider to accommodate long names.
3. Activating and Deactivating menu buttons now works.

Added `getValueListOfUsersWithRole` function

This function, which is available on the client side as well as from the web service, will return a list of User Ids and Names with the given role separated by semicolons.

See [getValueListOfUsersWithRole](#) for details.

Added `view` and `viewForm` functions

The `view` and `viewForm` functions provide an alternative form interface that prompts the user to first select from a list of rows, then compares the selected rows from all the selected sites and, if there are multiple versions of the row values, prompts the user to edit each version of the row separately. When the form is displayed it behaves as if the user were editing the row at a single site where there will never be more than one value in a field so the form no longer requires the use of the location value popup window. This also allows advanced form processing such as disabling or updating controls when a control is updated.

See [view](#) for details.

See [viewForm](#) for details.

Added `maximizeContentArea` and `restoreContentArea` functions

Added functions to maximize and restore the content area in the main EDM window.

See [maximizeContentArea](#) for details.

See [restoreContentArea](#) for details.

Modified the package selection screen

The package selection screen that is shown when opening an EDM form (such as "Item Prices") has been modified to be a table rather than just a list.

The table contains 3 columns:

1. Package Name
2. Created By
3. Create Date

The packages in the table can be sorted by these columns if the column header is clicked.

There is also a new "Search" field that can be used to filter the list of packages by any of the values in these 3 columns.

Added `publishMasterDataTables` function

Added function to roll back open and future transactions from selected tables and write the data to .csv files.

See [publishMasterDataTables](#) for details.

Added `hasPermission` function

The `hasPermission` function returns true if the current user has the given permission type for the given object type and name.

See [hasPermission](#) for details.

Perform an action on login

The system can be configured to perform an action every time a user logs in by setting the **onLogin** property in config.xml to an expression. The expression will be evaluated by the EDM client so any client functions may be used in the expression such as:

```
onLogin = "open( 'form', 'MyForm' )"
```

Removed automatic display of Site Manager

The Site Manager is no longer automatically displayed when the user logs in. If you want the system to continue to display the Site Manager for all users with permission to use it, add the following option to your config.xml file:

```
onLogin = "iif( hasPermission( 'execute', 'function', 'showSiteManager'
), showSiteManager(), 0 )"
```

Changes to when the update button is displayed

When editing a form (such as pricing or items) with a package that has just 1 site, the forms framework no longer requires you to click an update button in order to change a value. You can now edit it in place just like if you were on a new record.

Added new EDM Function: selectSessionPackage()

A new function has been added that will set the current session package.

See [selectSessionPackage](#) for details.

Added new EDM Function: openPlayer()

A new function has been added that opens a Table, Query, or Form on the client.

See [openPlayer](#) for details.

Added config option to turn form colors on or off

EDM Forms are transparent by default to promote a more harmonious and integrated experience. This can be changed by adding the option 'useFormColors = "Y"' to the config.xml file, which will change the forms to use the background colors set by the designer.

Changes to the Micros Menu Button Editor

The radio buttons to switch between active and inactive menu buttons have been removed and all menu buttons that are assigned to the current screen are shown. Inactive menu buttons are now grayed out and are placed at the end of the screen. The reason why the menu button is inactive is displayed as a tooltip when the mouse pointer is placed over the button for a moment. Some menu buttons were also being displayed as inactive when they should have been active. This has been fixed.

Fixed setValue()

The setValue() method was not flagging the data as changed, so saving a record was not saving the data properly. This has been fixed.

Added a function to get the next unused ID for a table column

A new function has been added that will return the next available ID for a given table column at all sites.

See [getNextUnusedId](#) for details.

Fixed a problem with type to select from combo boxes in the Location Values window

When the Location Values window is displayed and the underlying control is a combo box, the user could not type to select items in the drop down list. This has been fixed.

Fixed processing committed transactions on master data and subscriptions that have become effective

Retrieving the key values for the DataSet was erroring with duplicate keys in some cases. This has been fixed.

Changed partition function to use absolute value of partition ID calculated by partition expression

In order to prevent the user from having to specifically handle negative values when creating a partition expression, the partition function now uses the absolute value of partition ID calculated by partition expression. For example, when partitioning on site ID's, the partition ID for a negative site number will be greater or equal to 0. This prevents unnecessary errors in partitioning caused by negative values returned from partition expressions such as TOLOC % 3 which evaluates to -1 if TOLOC = -1.

Fixed Amazon file transfers to not leave incomplete files

Changed the Amazon file transfer process, used to transfer transaction files and to synchronize directories, to transfer each file to a temporary name and not rename it to the real name until the transfer successfully completes. This prevents the file transfer function from leaving a partially transferred file in the directory which the system cannot process.

Fixed combo box row source to allow function calls in SQL

Fixed error that was generated when the row source property of a combo box contained an SQL statement with an EDM function call in it such as:

```
select trim(Description) as Name, UpsizeType from tbl_UpsizeType where  
trim(Description) <> 'King'
```

Fixed memory leak in scheduled tasks

Fixed the task scheduler so that when it re-schedules a task, the thread for the previous task ends gracefully instead of staying in memory.

Added functions to set control values to null

Added functions to set control values to null. To set controls that won't have site-specific value, such as controls without sources or controls whose source is a system table, use `setValueToNull('ctlNames')` or `setFormValueToNull('formName', 'ctlNames')`. To set controls whose source is a column in an application table that may have multiple values use `setSiteValueToNull('ctlNames')` or `setFormSiteValueToNull('formName', 'ctlNames')`. See `setSiteValueToNull` function documentation for details.

Fixed scheduled task timer to continue running repeating tasks even after error

For example, to display the currently selected package and its version in the welcome bar add the following line to config.xml:

```
welcomeMessageExpressionPanera = "iif( getSessionPackageName() = ", 'No Package Selected', ' Package: ' & getSessionPackageName() & iif( getSessionPackageVersionName() = ", ", ", ' ; Version: ' & getSessionPackageVersionName() ) )"
```

If no expression is found in config.xml, the default expression is:

```
welcomeMessageExpression = "getCompanyName() & iif( getSessionPackageName() = ", ", ", ' - Package: ' & getSessionPackageName() )"
```

Added function to unlock application objects

The unlockAppObjects function displays a list of locked application objects so that they can be manually unlocked if necessary. This enables administrators to unlock a form or query that was locked when it was opened in design view and then never unlocked due to a browser crash or other reason. To use the function, create a new menu item with the action set to `unlockAppObjects()`.

See [unlockAppObjects](#) for details.

Fixed form/query designer to release lock on error

Changed the process of opening a form or query in design view so that if an error occurs in opening the object in design view, the object is unlocked instead of being left locked.

EDM 7.1.1 Release Notes

The following changes were made in this release.

Support partitioning the audit table for improved performance

The audit table can now be partitioned across multiple database servers which process queries in parallel for much faster performance. Transaction records are divided among the partitions based on a partitioning expression. Queries on the audit table are sent to each partition server for parallel processing and the results are merged by EDM so that the partitioning is transparent to EDM users. The primary partition will continue to reside in the EDM central database, additional partitions can be stored on other servers. See [partitionTable](#) for details.


Added function to copy and refresh site data

The new function `copyAndRefreshSiteData` enables users to tell the system to copy data from a source site to one or more destination sites and send a table refresh to each destination site at a specific date and time. Users select the source site, the destination sites, the effective date, and the tables to copy and refresh. If the effective date is left blank or is the current date and time then the system will perform the copy and refresh immediately. If the effective date is in the future, then the system will schedule the copy and refresh to be performed on the effective date. This function is useful for setting up data for a future rollout and then scheduling when the sites should receive the new data. See [copyAndRefreshSiteData](#) for details.

Created simpler bulk insert process for SQL Server

Change the SQL Server bulk insert process, used to add large numbers of rows to the audit table during record copying, to send blocks of insert statements instead of using the BULK INSERT command to import the data from a file. This makes it easier and safer for users to use the bulk insert capability without having to concern them with themselves with writing the data file to a shared location and enabling OLE Automation on their SQL Server installation. The new process is also compatible with SQL Server 2005 and 2008.

Edit Scheduled Tasks

Added "Edit Scheduled Tasks" menu option to the "Setup" menu to enable users to edit the scheduled server tasks. Scheduled tasks are saved in the config.xml file. Users can choose the frequency of the tasks, the first date and time the task should run, which days of the week the task should run on if it is a recurring task, and an expression to evaluate. The expression may call any server-side function in the system and multiple functions may be called by putting a plus sign  between them. For example, users may wish to schedule transaction processing to occur at 3am every day.

Changed Amazon directory synchronization to use MD5 file comparison

When synchronizing a local directory with a directory on Amazon S3, the system will now calculate the MD5 hash code for the local file and the Amazon file to determine if they are the same or not instead of using the file timestamp. This is a more accurate means of comparing files and eliminates the challenges of interpreting file time stamps across multiple time zones.

Added function: updateWebDirToMatchLocalDir

The `updateWebDirToMatchLocalDir` function can be used to update a directory on the server to match a local directory. This can be used to keep a directory of image or document files at remote sites synchronized with a directory on the server. See [Web Transfer - Transferring Directories](#) for details.

Log server job start time, end time, and duration

Added log statement for all jobs that run as background jobs on the server to show their start time, end time, and duration to make it easier to see what jobs are running at any moment. The log statements look like the ones below:

```
2012-08-28 16:45:39 INFO Starting server job 3: Partition Table EDMserver.BackgroundJob$1.run:202
```

...

```
2012-08-28 16:45:46 INFO Finished server job 3: Partition Table in 0:00:07 EDMserver.BackgroundJob$1.run:210
```

Fixed remainder (%) operator in expressions

Form designers can now use the remainder operator (%) in expressions to calculate the remainder of a division operations. For example, $5 \% 3$ will return the value 2, the remainder when dividing 5 by 3.

Fixed problem closing temp table when using expression in form source query

Corrected the temp table handling so that forms may have expressions in the where clause of their source query such as 'trim(columnname)'.

Fixed IRIS menu editor button sequencing problem

The button sequencing method was not being called when the menu was saved. It is now, which will fix any problems with buttons being saved out of sequence.

Fixed type-ahead error

Fixed error that sometimes occurred when typing a value into a combo box to try to find the correct item by typing the first few characters.

Eliminated excess logging from transaction import process

Change import transaction function to not log the start and end of the audit table update job for each transaction imported. These log statements were not needed and made the log file too large.

The Role Editor was fixed so that permissions were not deleted when renaming a role.

A rare issue was fixed where permissions would be lost when renaming a Role.

The Policy file has been migrated to the database.

The current policy file (which contains Users, Roles, Permission Types and Permissions) will be migrated to the system database and archived. Going forward, the system will use these new database tables when validating access to and usage of EDM. The major benefit is that security

date, just the file information. The system sends or requests the file from the central site. See `sendFile` and `requestFile` function documentation for details.

Added client function: `setSiteValue`

Added a client function "setSiteValue" so that users can set values for sites while retrieving values from other controls. The function takes 2 parameters, a Control's name, and an **expression**.

Example

```
setSiteValue( 'S1','"Hello"' )
```

This sets the value of the control named 'S1' to be the given expression, which is evaluated by the expression parser to the string "Hello". It required 3 apostrophes on each side so that EDM's expression parser knows that it is a string expression.

Example of getting a value from another control

```
setSiteValue( 'S1', S2 )
```

This sets the value of the control named 'S1' to be the value of control named S2. There are no apostrophes because we want the value of the control, not the string constant 'S2'.

Example of getting a value from another control, with concatenation

```
setSiteValue( 'S1','"A"&S2' )
```

This sets the value of the control named 'S1' to be the given expression, which is evaluated by the expression parser to the string "A" concatenated (&) with the value of the control named S2.

Added client function: `setFormSiteValue`

This function is just like `setSiteValue`, but you can pass in the name of another form. This is how you can set the value of a control on another form, which **must be open**.

Example

```
setFormSiteValue( 'OtherForm', 'S1','"Hello"' )
```

Control Value functions changed to be more site specific

The `setValue()`, `setSiteValue()`, and `setFormSiteValue()` functions will only update the sites that are being updated in the Location Values popup. If you have a package with sites 1, 2 and 3 in it, but are updating a value in the Location Values popup that is only at sites 2 and 3, only sites 2 and 3 will be updated.

EDM 7.1.0.70 Release Notes

The following changes were made in this release.

Changed site reset to re-commit transactions

Changed the `resetSite` command to re-commit any committed transactions to the site to ensure the site gets them.

Changed table refresh to continue sending to other sites after error

Changed the send table refresh process so that if an error is encountered when sending a refresh to one site in the list, the process logs the error and continues on to the next site. The system will attempt to send the refresh to all the requested sites and log any errors in the log file. The first 10 errors, if any, will be reported to the user if the refresh was started from the menu.

Added server function: `resetSites`

Added a server function "resetSites" so that users can trigger site resets from the web service that evaluates expressions or from a scheduled task. The function takes a single parameter which is a string of comma-delimited site id's.

Example

```
resetSite( '1001,1002,1003' )
```

Added server function: `executeDirectSQL`

Added function to execute a query on a data source and return the result rows. This server function can be called in the Source property of form controls.

Syntax	<code>executeDirectSQL('dataSourceName', 'sql'</code>)
Parameters	dataSourceName - Name of the EDM data source where the query is to execute. sql - SQL statement to execute.
Returns	Data set containing the result rows from the query.

Added server function: runApp

Added server function to execute a command line so that users can schedule commands to be executed.

Example

```
<schedule frequency = "TIMED" time = "1/1/2001 12:15 AM" task = "runApp('cmd /c copy c:\EDMweb\go.bat c:\EDMweb\test.bat', true )" />
```

Syntax	runApp('commandLine', 'wait')
Parameters	commandLine - Command line to execute.
	wait - True if system should wait for command to finish before continuing.
Returns	Exit code from command, 0 typically means successful.

Added client functions to send and receive transaction files

Added client functions to enable remote sites to separately send and receive transaction files so that if there is a reset sent from central to a remote site, the reset will be processed and the outgoing directory will be cleared before sending any files back to central.

sendSiteFilesViaWeb(siteldList)	Sends transaction files via HTTP.
receiveSiteFilesViaWeb(siteldList)	Receives transaction files via HTTP.

Added server functions to transfer transaction files

Added server functions to enable transaction files to be transferred as a scheduled task.

transferSiteFilesViaWeb(siteldList)	Transfers transaction files via HTTP.
transferSiteFilesViaFTP(siteldList)	Transfers transaction files via FTP.
transferSiteFilesViaRAS(siteldList)	Transfers transaction files via direct network connection.
sendSiteFilesViaWeb(siteldList)	Sends transaction files via HTTP.
receiveSiteFilesViaWeb(siteldList)	Receives transaction files via HTTP.

Log transaction files

Added logging at remote sites to record received, processed, and sent transaction files to aid in diagnosing transaction processing issues.

Improved commit locking

Improved commit locking so that when a large multi-site commit process is started, the commit lock is obtained and released for each site instead of obtaining the lock at the beginning and not releasing it until all sites have been committed. This causes the commit lock to only be held for the duration of a commit for a single site so that other commit processes can execute without waiting a long time for the lock. Also, when automatically committing table refreshes, the system can now wait indefinitely for the commit lock instead of timing out after 5 seconds. This prevents the system from generating an unnecessary timeout error when the refresh commit needs to wait more than 5 seconds for the commit lock.

Added server functions to online help

Added a section that describes all the server-side functions to the online help. Server-side functions are available to be called in expressions executed by calling the web service that evaluates expressions and by tasks scheduled in config.xml.

See [Server Functions](#)

EDM 7.0.8 Release Notes

The following changes were made in this release.

Added web service to evaluate functions

Added an evaluateExpression web service so other processes can ask EDM to evaluate an expression such as processTransactions(*). This is useful for advanced users that want to develop their own applications that integrate with EDM.

Parameters:

expression - Expression to evaluate.

userId - ID of the EDM user.

encryptedPassword - Encrypted password for the user from policy.xml.

company - Name of the company for which the expression is evaluated..

Returns:

Results of evaluating the expression.

The WSDL for the service can be found at URL <http://EDMserver/EDM/service?wsdl>. The XSD for the service can be found at URL: <http://EDMserver/EDM/service?xsd=1>.

Improved cache reload performance

Change the Copy Location Data and Delete Location Data process to only update the remote table cache version number once instead of once per table so that, for users with multiple servers, one server performing one of these processes does not trigger all the other servers to reload the cache every second.

Changed ping timer to not display wait cursor

Change the ping timer that periodically checks for commands from the server to not display the wait cursor while communicating with the server. This resolves the issue of some user keystrokes being lost due to the wait cursor being displayed by the background ping process.

Added a Micros Menu Button Editor

Added function: editMicrosButtons()

Added function to launch the Micros Menu Button Editor. EDM users can now change how the Menu Buttons will look on the Micros POS Terminal before sending the changes to the store.

Added function: setPropertyFileValue

The setPropertyFileValue function enables users to update the value of a property in a property file on the server. For example, to set the mail.smtp.host property in the mail.properties file call: setPropertyFileValue('mail.smtp.host', 'smtp.gmail.com', 'mail.properties', false, 'EDM E-Mail Settings')

See [setPropertyFileValue](#) function documentation for details.

Reduced cache reloading

Changed the remote table cache to only update the remote table records in the cache when the values have actually changed instead of replacing an old value with a new identical value and triggering a cache reload.

Fixed insert error caused by inserting same column name multiple times

Fixed a problem in the form insert process that sometimes generated an error that the INSERT statement was invalid because the same column name was listed multiple times. An example of the error is: Caused by: com.microsoft.sqlserver.jdbc.SQLServerException: Column name 'MeasID' appears more than once in the result column list.

Fixed Edit Shared Table Groups

Fixed the Edit Shared Table Groups option to properly save new and updated shared table groups.

EDM 7.0.7 Release Notes

The following changes were made in this release.

Support Amazon Simple Storage Service (S3) for file transfer

EDM users can now take advantage of the Amazon Simple Storage Service (S3) to leverage the power of cloud computing to transfer files between the central site and remote sites. S3 gives users access to the same highly scalable, reliable, secure, fast, inexpensive infrastructure that Amazon uses to run its own global network of web sites. The system can transfer transaction files via Amazon S3 or synchronize entire directories with directories on Amazon S3.

See [Amazon S3 Transfer](#) for details.

Added function: sendTransactionFilesToAmazon

Added function to send transaction files to the Amazon Simple Storage Service (S3). Remote sites can send transaction files to the central site by sending them to the incoming directory on S3 where the central site can retrieve them. Central sites can send transaction files to remote sites by sending them to the outgoing directory on S3 where the remote sites can retrieve them. Users must have a valid Amazon Web Services account with access to S3 to use the function.

See [sendTransactionFilesToAmazon](#) function documentation for details.

Added function: receiveTransactionFilesFromAmazon

Added function to receive transaction files from the Amazon Simple Storage Service (S3). Remote sites can receive transaction files from the central site by receiving them from the outgoing directory on S3 after the central site has sent them to S3. Central sites can receive transaction files from remote sites by receiving them from the incoming directory on S3 after the remote sites have sent them to S3. Users must have a valid

Amazon Web Services account with access to S3 to use the function.
See [receiveTransactionFilesFromAmazon](#) function documentation for details.

Added function: encryptAmazonCredentials

The encryptAmazonCredentials function can be used to encrypt a property file containing the Amazon Web Services (AWS) access key and secret key. The encrypted credentials can then be passed to the functions that send and receive transaction files from Amazon S3 so that they can successfully log into AWS. The encrypted property file can be sent to remote sites so that they can also connect to AWS to send and receive transaction files without making the AWS access key key and secret key visible to personnel at the site.
See [encryptAmazonCredentials](#) function documentation for details.

Added function: updateLocalDirToMatchAmazonDir

The updateLocalDirToMatchAmazonDir function updates a local directory to match a directory on Amazon S3. This can be used to easily update large numbers of files at remote sites such as image or document files. For example, if remote sites have a directory with all the image files used by the POS system, a directory can be created on S3 that contains the latest image files and the remote site can compare the directories periodically or on demand to download any new or updated files and delete any files that no longer need to be kept.
See [Amazon S3 Transfer](#) for details.

Added function: updateAmazonDirToMatchLocalDir

The updateAmazonDirToMatchLocalDir function updates an Amazon S3 directory to match a local directory. This can be used to easily update large numbers of files on Amazon S3, such as image or document files, so that remote sites can retrieve them with the [updateLocalDirToMatchAmazonDir](#) function. For example, if the central EDM web server has a directory with all the image files used by the POS system at remote sites, a directory on Amazon S3 can be updated to match the server directory.
See [Amazon S3 Transfer](#) for details.

Added function: getPropertyFileValue

The getPropertyFileValue function returns the value of a property in a property file. This function can be used to get the encrypted Amazon Web Services credentials from the property file to be passed to the functions that send and receive files from Amazon S3. For example, to receive transaction files at a remote site from S3 and get the AWS credentials property value from a property file called 'aws.properties' call: **receiveTransactionFilesFromAmazon(" , 'mybucket', 'outgoing', getPropertyFileValue('awsCredentials', 'aws.properties'))**{*}.
See [getPropertyFileValue](#) function documentation for details.

Added function: updateLocalDirToMatchWebDir

The updateLocalDirToMatchWebDir function can be used to synchronize a local directory with a directory on the server. This can be used to keep a directory of image or document files at remote sites synchronized with a directory on the server.
See [Web Transfer - Transferring Directories](#) for details.

Added 'Evaluate' transaction

Added a new type of transaction that tells the receiving EDM server to evaluate an expression. The expression can contain any valid EDM server functions but will not be able to call functions that are only available on the client side since the expression is evaluated on the server. An example of how this transaction may be used would be to tell remote sites to update a local directory to match a directory on the server when you have added or changed the contents of the server directory. An example EDM menu item action that sends this transaction after prompting the user to select sites is:
sendEvaluateTransaction('updateLocalDirToMatchWebDir("files/images", "c:\images", true)', " , " , '0-Default Package', " , true)

Support comet notifications in Tomcat 6 and Tomcat 7

The system now supports using Comet (HTTP push) to notify users instantly when notifications are sent out in both Tomcat 6 and Tomcat 7. If Comet notifications are not enabled, then the applet pings the server every 15 seconds checking for notifications. To enable Comet notifications add either the CometServlet6 or CometServlet7 to the webapps/EDM/web-inf/web.xml file as follows:

```
<servlet>
<servlet-name>cometservlet</servlet-name>
<servlet-class>EDMserver.CometServlet6</servlet-class>
<load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
<servlet-name>cometservlet</servlet-name>
<url-pattern>*.comet</url-pattern>
</servlet-mapping>
```

Support copying Positouch screens to other sites

Added 'Copy screen to sites' button to the Positouch screen editor that enables users to copy the selected screen to other sites that do not already have the screen. If users try to copy a screen to a site that already has it, a duplicate key error will be displayed.

Support grouping subform records by non-key columns

Changed form processing so that if the designer sets the Acting Keys property of the subform to a list of columns that contain columns that are not in the primary key, the subform will group the records by the column values specified in Acting Keys. For example, on the Micros Item form, the price subform has the Acting Keys property set to 'mi_seq;effective_from' so the price records are grouped by the 'Effective From' date instead of by the 'mi_price_seq' column which makes it easier to use.

Added form name to all form error messages

Added the name of the form to all error messages displayed by a form to help form designers diagnose problems more quickly when a form has multiple sub-forms.

Changed Positouch import to work with NewScreens XML element

Changed Positouch import process to look up tables in the data map XML file by table name instead of by XML element names so that it will work with either 'UpdateScreens' elements or 'NewScreens' elements in the screens.xml file. Also added log entries for starting and successfully completing the import process.

Added function: getConfigProperty

The getConfigProperty function enables users to get the value of a property in the config.xml file. This can be used to pass values from config.xml to functions. For example, if the encrypted Amazon AWS credentials are in the config.xml in a property called **awsCredentials** then you can call the Amazon transaction file transfer functions like this:

```
receiveTransactionFilesFromAmazon( '*', 'mybucket', 'incoming', getConfigProperty( 'awsCredentials' ) )
```

See [getConfigProperty](#) function documentation for details.

Added function: setConfigProperty

The setConfigProperty function enables callers to set the value of a property in the config.xml file. For example, to set the **awsCredentials** property to the encrypted AWS credentials call:

```
setConfigProperty( 'awsCredentials', 'myencryptedawscredentials' )
```

See [setConfigProperty](#) function documentation for details.

Added function: generateFunctionHelp

The generateFunctionHelp function generates an HTML help file for a list of PluginLoader classes. This is useful for users that develop their own plug-in functions and would like to generate help information describing the functions and their parameters in the same format the system uses.

See [generateFunctionHelp](#) function documentation for details.

Improved session removal for forwarded commands

Changed forwarded command processing to immediately remove the session information after processing the command even for GetJobStatus commands which were not having their sessions removed previously. This reduces memory usage on the server and eliminates the need to time out user sessions to avoid filling up memory.

Fixed table refresh to roll back transactions with status "Received"

Change the transaction roll back process used when sending table refreshes to also roll back transactions with future effective dates that have the status "Received" in the central database. This resolves the problem of committing a transaction with a future effective date to a site then sending a table refresh after the site has received the transaction but before the future effective date arrives and causing the change to take effect immediately instead of on the future effective date.

Fixed key value prompts when using the form Copy Record option

Fixed the dialog that prompts users to enter new key values for tables that have multiple key values such as the IRIS Child Item Price table so that the prompt displays correctly for each key value instead of being blank for some of the values.

Fixed "File already exists" error when committing packages

Changed the commit process so that when it creates a transaction file or temp file, if the file already exists it logs the fact in the system log and then renames the file to have a '.backup' extension. If there is already a file with the same name and a '.backup' extension it is deleted. This solves the problem of failures after a transaction file is created but before it is finished writing from requiring manual intervention to delete the file and synchronize the sent file number between central and the site.

Fixed "Company already exists" error

Changed the startup process to display the original error when the server fails to start correctly instead of a "Company already exists" error. In previous versions, if the server failed to start because of a license or database error or some other error, instead of reporting the original error, it was reporting "Company already exists" and users had to check the log to find the original error.

Fixed Cut, Copy, Paste popup menu in form designer

Fixed clipboard options in form designer that appear when the user right-clicks a text field to properly cut, copy, and paste text values.

Fixed bulk insert operation to display message on error

Changed bulk insert function used when copying records on a form to display an error message when it fails so the users don't have to check the log to find out if there was a failure.

Fix index out of bounds error when showing differences in Elstar screen editor

Fixed Elstar screen editor so that users with version 4 or later of the POS don't get an error when they select a screen that has multiple versions and they click the Show Differences button. The differences are now displayed correctly.

Fixed transaction viewer scrolling

Fixed the 'View Transaction Status' window to correctly scroll the transactions in the center window if the user selects a site with many error or delayed transactions.

Fixed 'Allowed Form Views'

Fixed form windows to not display the 'View form' or 'View spreadsheet' menu options on the menu bar when the form designer has not set the 'Allowed Views' property to all.

Fixed 'Limit to List' property

Fixed the 'Limit to List' property of combo box controls on forms to correctly prevent the user from entering a value that is not in the list when it is set to 'Yes'. If the control's 'Required' property is set to 'No' then a null value will be allowed in the control whether or not the user set the 'Null Item' property or not. If the 'Required' property and the 'Limit to List' property are both set to 'Yes' then the user may not enter a null value or a value that does not match one of the values in the list into the combo box text entry field.

Fixed CancelEvent to stop user from going to another row

Change the form processing so that in both grid view and form view if the CancelEvent() function is called during the Before Update or Before Insert event that is called when the user edits data and then tries to move to a different row, the user will stay on the original row. This resolves the issue of designing forms to validate row data before allowing updates and calling CancelEvent() when the data is not valid but the user was still able to go to a different row and the updates were lost.

EDM 7.0.6 Release Notes

Enhancements

The following enhancements were made in this release.

Support single sign-on with Tomcat 7

Changed system to enable request headers to identify users accessing the EDM central server through a portal so that the login information is preserved when running Tomcat 7.

Support disabling URL login

Changed system to check the servlet parameter 'enableUrlLogin' in web.xml to determine if users should be able to pass their user ID and password in on the URL. URL login is enabled if the servlet parameter is missing or set to Y. It can be disabled by setting it to N as follows:
<init-param><param-name>enableUrlLogin</param-name><param-value>N</param-value></init-param>

Support displaying URL when single-sign on user logs out

Changed logout so that when user access the system via browser using single sign-on, such as going through WebSeal which passes an "authorization" header to EDM, when they log out the system will jump to the configured URL, such as the single sign-on URL. To enable this, single sign-on must be enabled and add the following parameter to web.xml and specify the URL to display on logout:
<init-param><param-name>headerLoginLogoutURL</param-name><param-value><http://www.yahoo.com></param-value></init-param>

Added IRIS form to update tax category for selected items.

Added form 'Update Item TaxCatNum' that enables users to select an existing tax category number, preview a list of all the items that currently have that tax category number, then update all the items with the selected tax category number to use a new tax category number. To add the form to the menu, add an option with the action **open('form', 'Update Item TaxCatNum')**.

Added function: updateRows

Update selected rows for selected sites and generate transactions for the changes.

Syntax:

updateRows('tableName', 'columnNames', 'columnValues', 'whereClause', 'siteIdList', 'packageName', 'packageDescription', 'effectiveDate', 'terminationDate', forceProcessing)

Parameters:

tableName - Name of the table to be updated.

columnNames - Comma-delimited list of column names to be updated.

columnValues - New column values to be updated.

whereClause - Where clause to limit rows that are updated of the form or blank to update all rows for the selected sites. E.g. 'TaxCatNum = ' & Forms:[Update Item TaxCatNum]:OldTaxCatNum

siteIdList - Comma-delimited list of site ID's to update or blank for all sites.

packageName - Package name.

packageDescription - Package description.

effectiveDate - Date/time transaction becomes effective.

terminationDate - Date/time transaction terminates.

forceProcessing - True if transactions should be processed when only processing forced transactions.

Returns:

Void.

Added function: getControlValueRow

Create row with column values from the given list of controls on the active form.

Syntax:

getControlValueRow('controlNames')

Parameters:

controlNames - Comma-delimited list of names of the controls to use for row values.

Returns:

String containing delimited ascii values.

Added function: getPackageDescription

Get description of package that contains transaction for the active form.

Syntax:

getPackageDescription()

Parameters:

none

Returns:

Package description.

Added function: getEffectiveDate

Get date/time transactions becomes effective for the active form.

Syntax:

getEffectiveDate()

Parameters:

none

Returns:

Effective date in default system format.

Added function: getTerminationDate

Get date/time transactions terminate for the active form.

Syntax:

getTerminationDate()

Parameters:

none

Returns:

Termination date in default system format.

Added function: getForceProcessing

Get flag for whether transactions for the active form should be processed when only processing forced transactions.

Syntax:

getForceProcessing()

Parameters:

none

Returns:

True if transactions for the active form should be processed when only processing forced transactions.

Fixed Positouch screen editor to display error message properly

Fixed Positouch screen editor to display an error message to the user when an error occurs on one of the menu options displayed when the user right-clicks on a button.

Fixed Positouch screen editor error that occurred when user cancels creation of a new screen

When user creates a new screen then chooses not to save it, the system goes back to the list of screens without displaying a Null Pointer Exception error message.

Reduced memory usage by clearing visits on forwarded commands

Changed command forwarding to immediately remove the session/visit object once the forwarded command has completed its processing which may be in a background job.

EDM 7.0.5 Release Notes

Enhancements

The following enhancements were made in this release.

Changed user editor

Changed user editor to not show users that have permissions to view sites that the current user is not permitted to view.

Fixed role editor

Fixed role editor to not delete permissions when renaming role.

Fixed policy file loader

Fixed policy file loader to correctly assign role when loading the original version of the policy file that only supported one role per user.

Fixed problem with transactions getting saved in wrong package

Fixed package purge process to not delete packages with no transactions unless they were created at least one day ago. This prevents transactions from getting into the wrong package because the user created the package with id N, then the package was purged before the user could save his transactions, then a different packages was created with id N, and the first users transactions are saved with package id N.

Fixed 'Label control' property

Fixed problem with controls that have the 'Label control' property set so that when the control is made visible or invisible, the label control's visibility is set to match regardless of the order of the controls saved in the form file.

Fixed "Attempt to mutate in notification" error

Fixed "Attempt to mutate in notification" error that sometimes occurred when editing a table directly or using a form to edit a system table such as the site table.

Fixed a problem loading IRIS images

Clicking the load button in the Menu Button Editor for IRIS would sometimes cause a NullPointerException. This has been fixed.

EDM 7.0.4 Release Notes

Enhancements

The following enhancements were made in this release

Save forms and queries in separate XML files

Forms and queries are now saved as separate XML files in the 'forms' or 'queries' directory in the directory where the schema file is saved. This makes it easier to replace individual forms or queries by simply replacing the file.

New "Allowed Views" EDM Form property

EDM Forms now allow for the restriction of which view(s) are available to the user at runtime. You can specify Form, Spreadsheet, or All.

Support reloading cached data without restarting Tomcat

Added a new function **reloadCachedData** that can be added to the menu or called as a scheduled task. Calling the function causes the server to re-initialize itself and reloading the configuration data, the users, roles, permissions, sites, site groups, remote tables, menus, schema, table conversions, transient fields, snapshots, and the application forms and queries. The eliminates the need to restart the Tomcat server when any of this data is updated manually or by another process. Note that the command will stop users from interacting with the server until all currently

running requests and background jobs have been processed at which time the cached data will be reloaded so it is best not to run it while you have a long background job running.

Save security settings in XML file

Security settings are now saved in policy.xml with just the passwords encrypted instead of the entire file being encrypted. Encrypted policy files from older versions are automatically converted to the new format and renamed to 'policy_system_generated_backup'.

Automatically reload cached data if updated by another server

Changed system to check to see if sites, site groups, remote tables, users, roles, and permissions have been changed while the server is running and, if so, reload the changed data. This enables multiple servers to keep their settings synchronized. If one server updates the settings, the other servers will detect that their data is out of date the next time they try to access the data and reload the latest data.


Increased performance of allocating transaction ID's on SQL Server

When using SQL Server for the central database, the system now uses a single call to the database server to allocate a block of transaction id's using SQL Server transactions to prevent other processes from allocating the same ID's instead of multiple calls to lock and unlock the allocation process

Fixed error when sorting grid column

Fixed error that sometimes occurred when sorting a column of a form in grid view so that it no longer displays a NullPointerException error.

Fixed problem with EDM Form controls not respecting numeric data entry

Form controls that edit underlying numeric data now prevent the entry of data that will cause an exception upon save. Only digits, the minus sign , and the decimal point may be entered in numeric fields. The Max Decimals property, when set, is enforced.

New Report - Price Change by Price Group

There is a new report available that is similar to the existing Price Change by Price report. This new variation simply puts the sites below the price change detail in a collapsible list.

runPriceChangeByPriceGroupReport - Display a Pricing Report with the Site list below the price group detail.

Syntax:

```
runPriceChangeByPriceGroupReport()
```

Parameters:

None

Returns:

Void

New Search Functionality in forms

Users can now search for values in forms by typing in some text in a new search box in the toolbar. Find Previous and Find Next toolbar buttons have been added to the toolbar as well.

Location Value Popup Panels respect the boundaries of the browser window

When a Location Value Popup Panel comes up, it will no longer display off of the right side of the browser window.

Scheduled Tasks now show up in the Background Jobs Viewer

If you have scheduled tasks defined in config.xml, they will show up in the Background Job Viewer when they execute.

Acting Key Fields Enhanced

When a form designer specifies an Acting Key Field, such as "obj_num" for Micros, the user will be prompted to supply that value when copying records.

Fixed a background image error in the IRIS Menu Editor

When a background image was specified in a menu, but the image file did not exist on the server, a "NullPointerException" was generated. This has been fixed.

Support 'Label control' property to hide form labels when form controls are hidden

Added 'Label control' property to form controls so designers can specify which label is associated with a control. For example, when the designer adds a label control named 'idLabel' and a text control named 'id' to a form, she can also set the 'Label control' property of the text control to 'idLabel' so that if the 'id' text control is hidden, the label will be hidden also.

Improve Row Set Editor

Changed label on selected rows to "Rows included in row set:"

Fixed form so that when you select a different row set, the correct selected rows are shown.

Fixed form so when you create a new row set, you can add rows immediately without leaving the row set screen.

Changed editor to display row set ID number in list of row sets so they can be used when adding row set permissions.

Changed editor to not change the row set ID number when the row set is updated so existing row set permissions still work.

Changed editor to enable user to scroll through list of table columns and check viewable and editable columns.

Forms that hide data entry controls due to row set permissions can now also hide the label by setting the 'Label control' property of the control to the name of the label control on the form that is associated with the data entry control.

Changed forms to save blank numeric values as null

Changed forms so that when users are editing a numeric field, if they delete all the characters from the field, the saved value is NULL instead of 0.

EDM 7.0.3 Release Notes

Enhancements

The following enhancements were made in this release

Improved performance

Improved overall performance of system by streamlining memory management routines.

Enable non-administrators to edit users and roles

User and role editors have been improved so that users who have permission to use them can edit users and roles but cannot grant permissions that they do not already have. For example, a franchise owner can be granted permission to use the user and role editors so that he can set up users and roles for his employees but he will not be able to give them any permissions that he does not have himself.

Support Row-Level Security

Administrators can now control which sets of rows within a table users have permissions to view. For rows that the user can view, administrators can control which column can be viewed and which can be updated.

To create row sets, add a menu option with the name **Edit Row Sets** and the action **openRowSetForm()**. Users can create one or more row sets for a given table by giving them different names. Each row set has a set of select columns, a set of columns, and a set of selected rows. Select columns control which columns are displayed when selecting the rows for the row set. For example, to select items you would include the item number column and the item name column so that when selecting the items for the row set you can see both the item number and name. The filter flag on select columns determines whether the column is used to see if a row is in a row set. In our item example above, we would only set the filter flag on the item number column because we want items with specific numbers in the row set and don't care what the name is. The set of columns in a row set determines which columns in the table the user will be able to view and edit. If you want the user to be able to see the column, add it to the column list. If you want the user to be able to edit the column, check the Editable checkbox. To select the rows in the row set, click the **Add** button beneath the list of selected rows and click one or more rows to add to the row set and click OK.

To restrict a user to only seeing certain rows and columns add a **rowset** permission to the user's role for each row set you want them to have.

When users open a form, if the user has been given row set permissions for the table being edited, then she will only be able to view the rows defined in the row sets. In form view, if a field is for a column that is not in the list of row set columns then the control will be hidden (or filled with ### in grid view). If the field is visible but not editable then the user will not be allowed to change the value in the field.

Enhanced 'setValue' expression to accept a comma separated list of control names.

For example, to set the value of 3 controls based on the value of another field, use an expression like this:

```
=iif( tax1, setValue( 'tax3,tax4,tax5', 'T' ), setValue( 'tax3,tax4,tax5', 'F' ) )
```

Enhanced 'setControlProperty' expression to accept a comma separated list of control names.

For example, to hide 3 controls based on the value of another control, use an expression like this:

```
=iif( Price > 5, setControlProperty( 'tax3,tax4,tax5', 'visible', 'N' ), setControlProperty( 'tax3,tax4,tax5', 'visible', 'Y' ) )
```

Forms enhancements

Disabled columns are skipped when navigating with the TAB or Arrow Keys when a form is in Grid View.

Disabled columns (Text Entry fields, CheckBox fields, and ComboBox fields) are "grayed out" when a form is in Grid View.

There is now an 'update button' in form fields that, when clicked, brings up the Location Values editor. Pressing Ctrl+Enter still works as expected.

There is now a 'Locations' button on the Location Values editor that, when clicked, brings up the Locations window for the selected row. Pressing Ctrl+Enter still works as expected.

Form Designer enhancements

Toolbox tools are now "pluggable", meaning that new tools can be created and plugged into EDM via the ServerPlugins and ClientPlugins classes.

Null entries can now be added to ComboBox controls. Set "required" to "no" and then set the "Null Item" property to whatever you want the null entry to say in the drop down at runtime.

Added support for a ColorChooser control. The Color Chooser control, available in the Forms Designer Toolbox, will prompt the user for a color using the standard Color Picker window. When the color is chosen, the underlying field is saved as an integer representation of an RGB color value.

- NOTE: Micros uses a special color format, BGR instead of RGB in their database. To accommodate this, if you add "bgr" to the format field in the attributes window of the Color Chooser control, it will format the field accordingly.

Added support for a Micros specific Icon Chooser control. The Icon Chooser control, available in the Forms Designer Toolbox when designing forms for the Micros POS, will prompt the user for an icon that resides on the server. Once the icon is chosen, the icon ID is written to the underlying field.

The Forms Designer now opens in a Tab, rather than in separate windows.

You can now ctrl+click multiple controls in the designer window.

You can now edit common attributes of multiple selected controls in the attributes panel.

You can now change a control's type (or the type of multiple selected controls of the same type) to a different type. Select a control or controls, click the Edit menu and select the "Change control type" menu item.

Added filter field to help find fields by typing a partial name.

Find control with a certain source column by highlighting the field in the field list and pressing F3.

Auto-commit refreshes after each site

Refresh transactions are now committed as soon as each FIL is generated so if users send a refresh to 10 sites and get an error on site number 8, they do not need to resend the first 7 sites since they were successfully generated and committed.

Stop prompting for commit when closing user and role editors

System no longer prompts users to commit packages when closing the user or role editor.

Improved IRIS menu editor

IRIS menu editor now shows in a tab and support resizable placement grid.

Resolved Defects

The following defects were resolved in this release.

Fixed Validation Rules not being checked when making changes in the Location Values popup window.

When editing a field that has a validation rule in an EDM Form, the validation rule was not being respected when making changes in the Location Values popup window. This has been fixed.

Ensure difference reports are not cached in Edit Sharing

Changed Edit Sharing window to reload all difference reports from the server every time to avoid the problem of the report being cached and not updating the display when the report changes.

Fixed 'Index out of bounds' error on RequeryControl function

Fixed the requery control function so that if the control being requeryed is a list box or combo box with the same query as other list or combo box controls, all the controls have their list of rows updated at the same time and the 'Index out of bounds' error does not occur.

Fixed refresh file renaming

Refresh transactions now rename the .FIL.TMP when the XML is created and only for the transaction being committed. This solves the problem of all the .FIL.TMP files in the outgoing folder being renamed at once even if not all the refreshes are being committed and getting a 'File not found xxx/FIL.TMP' error.

Support bitmaps on IRIS menu tabs and backgrounds

Users can now set the background bitmap on IRIS menus and menu tabs.

Fixed transaction viewer

Transaction viewer no longer displays transactions from sites the the user does not have permission to see.

EDM 7.0.2 Release Notes

Enhancements

The following enhancements were made in this release

Changed reset and refresh to not commit entire system package

Changed the Reset Site, Send Table Refresh, Request Table Refresh, and Send Table Refresh to Test Site functions to only commit the requested transactions instead of all the transactions in the system package to prevent system processes from committing transactions concurrently.

Made auto-committing refreshes configurable

Added autoCommitSendRefresh and autoCommitRequestRefresh properties to config.xml enabling users to turn the auto committing for table refreshes on or off. Set the properties to "N" to disable auto committing or "Y" to enable it. If the properties are not found auto committing is enabled.

Resolved Defects

The following defects were resolved in this release.

Fixed error editing IRIS menu button properties

Fixed error that occurred when users tried to edit button properties in the IRIS menu editor.

Removed Edit Users and Edit Roles permissions from all roles

The system updates the policy file on startup if it is from version 7.0.1 or older to remove the permission to Edit Users and Edit Roles from all roles except Administrator. Administrators can give roles permissions to edit users or roles by granting the role view permission on the menu item in the role editor.

EDM 7.0.1 Release Notes

Enhancements

The following enhancements were made in this release

Configurable retries for server communication

The system can now be configured to retry communications to the web server if there is a communication error. The number of retries and the number of seconds between retries can be configured for both remote sites communicating to a central server and for users accessing the central server via a browser. The default number of retries is 4 for remote sites connecting to a central server and 3 for users connecting via browser. The first retry occurs immediately after the communication error, the default time between retries after the first one is 5 minutes for remote sites and 1 second for users accessing the central system via a browser.

To change the default the retry count and time between retries, add lines like these to config.xml

```
maxRequestRetries = "3"
```

```
secondsBetweenRequestRetries = "60"
```

To configure the retries at a remote site, add the lines to the config.xml at the remote site. To configure the retries for users accessing the central system via a browser, change config.xml on the central server.

Improved temp file handling

When transferring files between remote and central sites, the temp files are now named with the real file name plus the extension ".EDM_TEMP". This means that if the file transfer does not complete, when it restarts during the next web transfer, the same temp file will be overwritten. This prevents temp files from building up in the incoming folders and makes it easier to know what file failed.

Added Commit Wizard

Change the 'Commit Selected Packages' function to use a wizard to select the packages, sites, and effective dates. The wizard is displayed within a tab instead of in popup windows and gives the users the ability to move back and forth between the screens.

Added showDocument function

Added the function showDocument(URL) that causes the system to display a document in a separate browser window. This is useful for displaying help or other information when users select a menu option or a button on the form. If the document URL does not contain a complete path it will be relative to the webapps/EDM directory on the web server. For example, showDocument('help/mydoc.htm') will display the webapps/EDM/help/mydoc.htm file in a separate browser window.

showDocument - Display a document in a browser window. Applets can only display documents on the same server they were started from.

Syntax:

```
showDocument( 'url'
```

Parameters:

url - URL of the document to display relative to the web application directory.

Returns:

Void

Added queryValue function

Added function to directly execute an SQL statement on a data source without any parsing by EDM. The SQL statement can be any SQL statement that is valid on the server and database identified in the data source.

queryValue - Directly execute the given SQL statement on the given data source and return the value of the first column in the first row or a blank string if no rows returned. If the query is not a select statement, execute the SQL statement and return empty String. Use caution to avoid exceeding available memory.

Syntax:

queryValue('dataSourceName', 'sql')

Parameters:

dataSourceName - Name of the EDM data source where the query is to execute.

sql - SQL statement to execute.

Returns:

Value of the first column in the first row or a blank string if no rows returned.

Example menu action:

Msg('The site count is ' & queryValue('EDMWebHQ', 'select count(siteid) from cdmsite'), 'Site Count', 1)

Added getSelectedSites function

Added function to return the selected sites for the currently active form as a comma-delimited list of site ID numbers.

getSelectedSites - Return selected sites for the active form.

Syntax:

getSelectedSites()

Returns:

List of sites.

Support subforms without links to parent form

Changed subform controls so that in cases where the data on the parent form is not actually related to the data on the subform, the Master Link Fields and Child Link Fields properties may be left blank. This can be useful for creating a single parent form that has subforms for unrelated tables.

Stop browsers from caching transaction reports

Changed transaction reports so that they use a different file name each time they are run so that browsers won't cache an older version of the report and prevent users from seeing the latest version without refreshing the browser by pressing Ctrl-F5. Previous files used for the same report by the same user are deleted so that the report files do not fill up the hard drive over time.

Enable permissions for Application and User Menus

Administrators can now decide whether or not a user can see the menu options for Users, Change Password, Edit Users, Edit Roles, Application, Tables, Queries, and Forms. For example, to limit the Central user role to only allowing users to change their password but not be able to work with the Application menu or its sub-menus, you would add permissions like this to the Central role:

Object Type	Object Name	View
menu	*	Yes
menu	Users	Yes
menuitem	Change Password	Yes
menuitem	Edit Users	No
menuitem	Edit Roles	No
menu	Application	No
menu	Tables	No
menu	Queries	No
menu	Forms	No

Clarified label in transaction viewer

Changed "Sites Have Warnings" to "Sites late in responding" on transaction viewer to make it clearer to users why the sites are listed.

Resolved Defects

The following defects were resolved in this release.

Replaced popup windows

Changed popup windows to appear within the tabs of the main EDM window in the browser to eliminate problems with popups going behind the browser window or not responding to user actions.

Don't allow saving query with same name as table

Changed system so that you cannot save or rename a query to the same name as a table (with or without the prefix such as 'IRIS_dbo') instead of deleting the table when a query of the same name was saved.

Prevent users from seeing packages created by other users

Fixed problem with background jobs that temporarily set the permission level to Administrator, such as running a transaction report or processing incoming transactions, so that the user that started the job does not have admin permissions while the job is running. This prevents them from seeing and committing packages created by other users when they should not be allowed.

Prevent users from reporting transaction packages created by other users

Fixed problem with transaction reports that enabled users to report on packages created by other users that don't share a site role by leaving 'All' packages selected. Now, if a user's role only includes certain sites she will only be able to select and report on packages she created or packages created by other users that have the same site role. A site role is any role that defines site permissions.

Support binary 0 in text columns

Added support for databases that contain binary 0 characters in text columns in the database. This eliminates the error when reading transaction XML files that the binary 0 is an invalid character.

Fixed Transaction Status Viewer

Fixed Transaction Status Viewer so that a user that is only allowed to see certain sites will not be able to see transaction statuses from other sites.

Fixed IRIS Menu Editor Copy

Fixed IRIS Menu Editor so that when copying menus to other sites the transactions are placed in the package selected by the user instead of in the default package.

Fixed Aloha Submenus Item Properties editor

The Item Properties Editor in the Aloha Submenus now fully displays the OK button.

Fixed POSitouch Button Repaint Issue

The display of the Cell Panel Buttons would sometimes be cut off. This has been fixed.

Fixed POSitouch Coupon Category Issue

The Coupon Categories in the POSitouch menu editor now properly save to the database.

User Guide

This is the home page for the EDM User Guide.

Getting Started	Reference the pages in this section for information on how to get started using EDM
Managing Sites	Reference the pages in this section for information about creating new sites and managing site databases
Data and Transactions	Reference the pages in this section for information about maintaining data, managing transactions, data recovery and import/export features
Editing Data	Reference the pages in this section for information about adding/editing/deleting records on forms or spreadsheets
Server Text File Editor	The <i>Server Text File Editor</i> is used to create or edit text files on the EDM Server.

Queries	Reference the pages in this section for information about queries, which are requests to read or write data in the database
Forms	Reference the pages in this section for information about forms, which are used to enter and edit data in tables. You can create your own forms and design the field layout, the fonts and colors, and the types of controls.
Expressions	Reference the pages in this section for information about expressions, which are used to calculate a value in a query column or a form control
Client Functions	Reference this page for a list of all functions that may be used on a client system. Functions can be called in Expressions, and on Control or Form Events.
Server Functions	Reference this page for a list of all functions that may be used on the server. Functions can be called in Expressions, and on Control or Form Events.
Format Strings	Reference the pages in this section for information about format strings, which are used to control how data is displayed on spreadsheets and forms, define fields in a table and controls on forms
File Transfer	Reference the pages in this section for information about how to configure the system to transfer transaction files between central and remote sites.
Polling Data	Reference the pages in this section for information about how to retrieve transactional data (sales, inventory usage, cash management) from central using EDM's powerful and flexible polling capabilities
Security	Reference the pages in this section for information about Internet and application security
Supported Databases	Reference the pages in this section for information about the different databases that EDM supports
Configuration Files	Reference the pages in this section for information about the EDM configuration files. You can modify these files to customize your installation.
Upgrading EDM or Your Application	Reference the pages in this section for information about upgrading EDM and/or your application
Frequently Asked Questions	Reference this page for frequently asked questions about EDM
Glossary	Reference this page for definitions of EDM terms and concepts
Extending EDM Functionality	Reference this page for information about the client and server plug-ins that you can use to extend EDM functionality
Translating EDM System Text	Reference this page for information about translating the text used within EDM to other languages, or to match the terms used by your organization

IRIS Editors

The following editors are related to the IRIS product.

Tax Rules Editor for IRIS	Use the <i>Tax Rules Editor</i> to define tax rules and corresponding tax rule conditions.
IRIS Config Group Editor	The <i>IRIS Config Group Editor</i> is used to create or edit Config Groups on the EDM Server.

Getting Started

Reference the pages in this section for information on how to get started using EDM. To return to the User Guide Home, click [here](#).

Starting a Central System	Reference this page for information about starting the Apache Tomcat Service to access the central system
Starting a Remote System	Reference this page for information about starting a remote system
Using the Menu	Reference this page for basic information about using the EDM Main Menu
Modifying the Menu	Reference this page for information about modifying the EDM Main Menu
Using an LDAP with EDM	Reference this page for information about maintaining security for EDM in an LDAP

Starting a Central System

Overview

The central system is normally running all the time as a service on the server. The system runs within the Apache Tomcat web server, which automatically starts up when the server is started. As long as the Apache Tomcat service is running, you can access the central system from any computer that has access to the server by opening a browser to:

```
http://server:8080/EDM
```

Where *server* is the name or IP address of the server where the Apache Tomcat web server is running. You can leave out the *:8080* if the system configuration has been changed to use the default port 80.

See [Apache Tomcat Web Server](#) for information on configuring the system to use the default port 80.

To Start Apache Tomcat Service

1. Go to *Start>Control Panel>Administrative Tools>Services*.
2. Select the *Apache Tomcat* service.
3. Click *Start*.

To Restart Apache Tomcat Service

1. Go to *Start>Control Panel>Administrative Tools>Services*.
2. Select the *Apache Tomcat* service.
3. Click *Restart*.

Starting a Remote System

Overview

To start the standalone system, type the following from a command-line:

```
"_JAVA_HOME_\bin\java.exe" \-jar _EDM_HOME_\lib\system.jar
```

Where *JAVA_HOME* is the folder where the Java Runtime Environment is installed and *EDM_HOME* is the folder where WebEDM is installed.

Java Runtime Environment Options

The following optional parameters can be added before the "-jar" option on the command-line.

Parameter	Description
"-Duser.dir=EDM_HOME"	Sets the current directory for the program (<i>EDM_HOME</i> is the folder where WebEDM is installed)
"-Dsun.java2d.noddraw"	Disables use of the Microsoft Direct Draw library

Command-Line Parameters

The following optional parameters can be added to the end of the command-line.

Parameter	Description
"-user=USER_ID"	Sets the login user ID
"-pwd=PASSWORD"	Sets the login password
"-expr=EXPRESSION"	Executes an expression on startup
"-nosplash"	Disables the system splash screen on startup
"-nomenu"	Disables the main menu on startup (do not use without an expression that ends with the Exit() function)

Example

The following command-line starts the system in the *C:\EDMWeb* folder, processes transactions and exits after Startup with:

- Direct Draw disabled
- The user ID set to *remote*
- The password set to *remote*
- The splash screen and menu disabled

```
"C:\Program Files\Java\jre1.5.0_02\bin\java.exe" "-Duser.dir=C:\EDMWeb"
"-Dsun.java2d.noddraw" \-jar C:\EDMWeb\lib\system.jar "-user=remote"
"-pwd=remote" "-nosplash" "-nomenu"
"-expr>ShowWarnings(0)+ShowErrors(0)+ProcessTransactions(' '*')+Exit()"
```

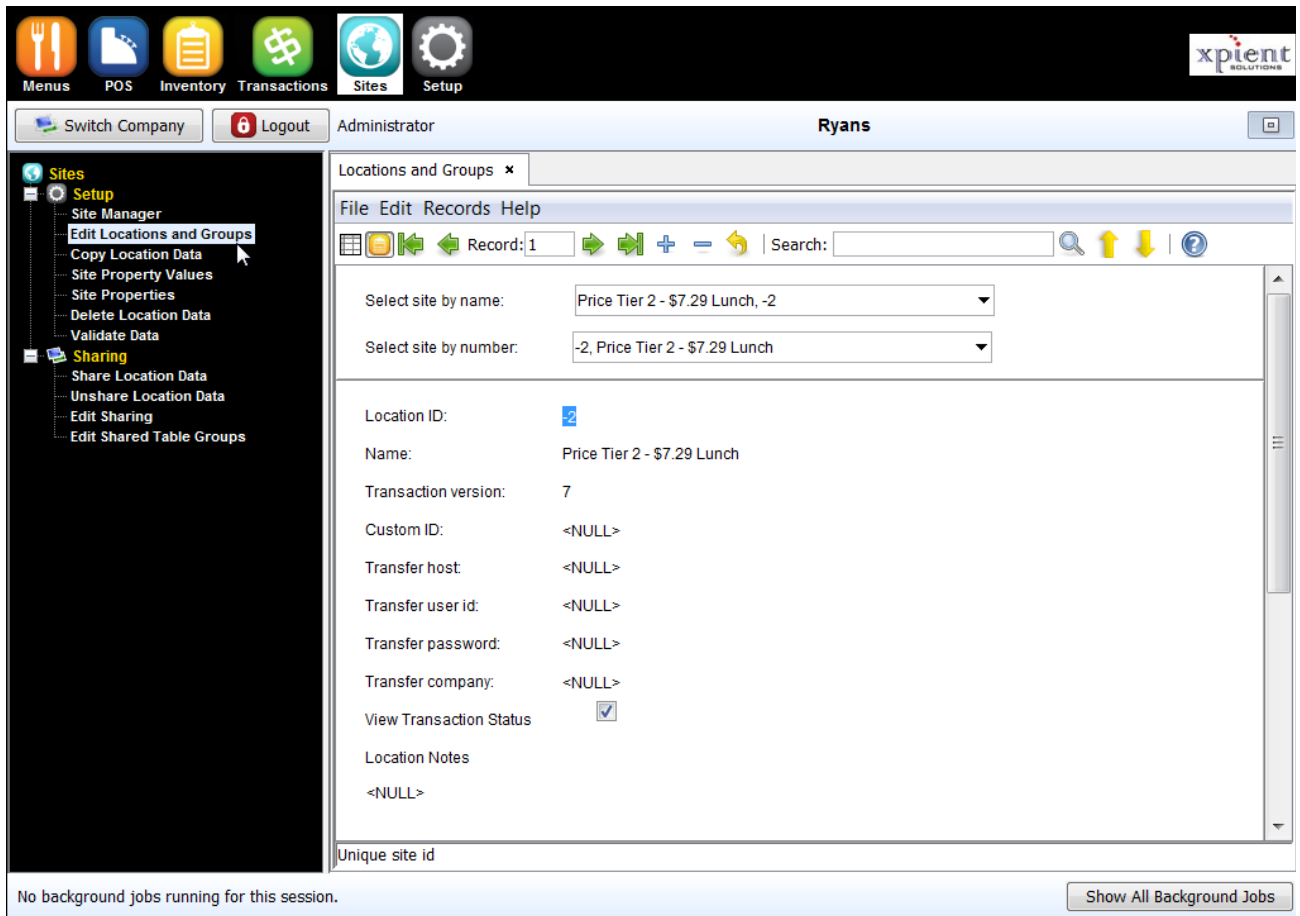
Using the Menu

The EDM Main Menu provides access to all the functions of the system. The menu can be modified to include any desired submenus and items (See [Modifying the Menu](#)).

At a central location, the EDM Main Menu typically includes data editing forms and system functions. At remote locations, the EDM Main Menu may only include a few buttons such as *Process Transactions* and *View Audit Trail*.

Performing a Task from the Main Menu

1. Select the module from the menu at the top of the browser. The modules include *Menus*, *POS*, *Inventory*, *Transactions*, *Sites* and *Setup*.
2. When you select a module, the left column of your browser is updated to list all the functions that are included in that module. Click the plus/minus sign to expand/collapse the submenus in the left column. Select the desired function from the left column. The form for the selected feature appears in the window on the right.



At the bottom of the EDM Main Menu, there is a small area showing the information on the background jobs that are running. Click the *Show All Background Jobs* button to view detailed information on all of the processes that are running.

Getting Help

To access the help system at any time from the EDM Main Menu or other windows, press *F1* on your keyboard. You can also right-click any menu option, and then click *Help* from the menu that appears.

Displaying System Information

To display the system splash screen, right-click the module name from the left column, and then click *About* from the menu that appears. The splash screen provides the following information:

- The system name
- The version number
- The copyright information

Menu Security

The options displayed on the menu are controlled by the security system. Some users may not see all the options on the menu because they are not authorized to use them. For example, the only menu options that you see when you log in as the default remote user are *Process All Transactions* and *View Transactions*.

To change which menu options are available to a user, either create a new role or modify the user's existing role to have different permissions for the menu options. See the *Remote* role for an example of a role with very limited menu option permissions.

Modifying the Menu

Overview

This page describes modifying the EDM Main Menu. The menu options are stored in a menu configuration file and may be edited there. See [Main Menu](#) for detailed information on the structure of the menu file.

Users with the proper permissions can change the menus and items on the EDM Main Menu by simply right-clicking the menu (or item) and selecting *Edit*.

Module Options

The following menu editing options are available when you right-click a module from the left window. Modules include *Menus*, *POS*, *Inventory*, *Transactions*, *Sites* and *Setup*.

Option	Select it to
New Menu Above...	Insert a new menu above the selected menu
New Menu Below...	Insert a new menu below the selected menu
New Submenu...	Insert a new submenu of the selected menu
New Menu Item...	Insert a new menu item as the last child of the selected menu
Edit Menu...	Edit the menu name
Cut	Cut the menu to the clipboard
Copy	Copy the menu to the clipboard
Paste Above	Paste the menu from the clipboard above the selected menu
Paste Below	Paste the menu from the clipboard below the selected menu
Paste Child	Paste the menu from the clipboard as a child of the selected menu
Delete Menu...	Delete the selected menu
Help	Access the help documentation
About	View EDM version and copyright information

Function Options

The following menu editing options are available when you right-click a function from the left column.

Option	Select it to
Do action	Open the form for the currently selected feature
New Item Above...	Insert a new menu item above the selected item
New Item Below...	Insert a new menu item below the selected item
Edit Item...	Edit the item name and action
Cut	Cut the item to the clipboard
Copy	Copy the item to the clipboard
Paste Above	Paste the item from the clipboard above the selected item
Paste Below	Paste the item from the clipboard below the selected item
Delete Menu...	Delete the selected item
Help	Access the help documentation

Using an LDAP with EDM

Introduction

8 To disable the LDAP configuration:

- Login with a valid LDAP ID
- Go to the *LDAP Configuration* tool
- Click *Disable LDAP*



Usage

Users will not be able to use the default policy file or passwords to login after the LDAP connection is configured. User information, role and permission information will be maintained entirely by the LDAP.

Managing Sites

Reference the pages in this section for information about creating new sites and managing site databases. To return to the User Guide Home, click [here](#).

Creating a Site	Reference this page for information about creating a new remote location
Location Data	Reference the pages in this section for information about managing data for remote locations
Creating a Database for a New Remote Location	Reference this page for information about creating a database for a new remote location
Sending a Database to a New Remote Location	Reference the pages in this section for information about sending a database or data for a new remote location
Keeping Central and Remote Data Synchronized	Reference this page for information about synchronizing the data at central and remote locations

Creating a Site

This page describes how to create a new remote location.

1. Select the *Sites* module.
2. Select *Edit Locations And Groups* from the *Setup* menu in the left column..
3. Click



- on the tool bar.
4. Enter the site data.
5. To enter more site records, click



again and enter another site's data.

Select site by name:	2105 - Columbia #1, SC, 2105
Select site by number:	2105, 2105 - Columbia #1, SC
Location ID:	2105
Name:	2105 - Columbia #1, SC
Transaction version:	7
Custom ID:	<NULL>
Transfer host:	<NULL>
Transfer user id:	<NULL>
Transfer password:	<NULL>
Transfer company:	<NULL>
View Transaction Status	<input checked="" type="checkbox"/>
Location Notes	<NULL>
Locations (or groups) in this group:	
	Location
+	<NULL>

Add Locations To This Group

The following table describes the fields on the *Locations And Groups* form.

Field name	Description
Select site by name	This drop-down list enables the user to select the site record to view or edit by name
Select site by number	This drop-down list enables the user to select the site record to view or edit by site number

Location ID:	This field contains a unique ID number for the site. 0 is usually used for the central site and negative numbers are usually used for share groups and location groups.
Name	This field contains the name of the site
Transaction Version	This field contains the highest transaction file version number accepted by this site. When set to a negative number, no transaction files will ever be generated for the site.
Custom ID	This field can be used to customize the folders and filenames when sending and receiving files. For example, to place the outgoing files for store 1234 in a folder called <i>Outgoing\001234</i> , set the <i>Custom ID</i> to 001234 and set the <i>localSendDir</i> in the <i>config.xml</i> to "Outgoing%REMOTECUSTOMID%"
Transfer host	This field contains the FTP (or HTTP) host URL for communicating with site. For example, if the central site communicates with an FTP server at each remote site to deliver transaction files, the <i>Transfer host</i> field for each remote site should be set to the URL or IP address of the FTP server. For HTTP communications (such as with <i>transfer_web.bat</i>), the <i>Transfer host</i> field should be the name of the Web server followed by the port number (if necessary) and the Web application name. For example, if the system is running on <i>EDMServer1</i> on port 8080, then set the <i>Transfer host</i> field to <i>EDMServer1:8080/EDM</i> . This field value is case-sensitive and the forward slash character must be used.
Transfer user id	This field contains the User ID for the FTP or HTTP server when using the file transfer functions
Transfer password	This field contains the Password for the FTP or HTTP server when using the file transfer functions
Transfer company	This field contains the company name for the HTTP server when using the file transfer functions, in case the central server being contacted has more than one company configured
View Transaction Status	Select this option to display the status of the transaction
Location Note	This field contains any comments or notes about the location
Locations (or groups) in this group	This field contains a list of locations (or locations groups) contained by this location group. Real central and remote locations should not have other locations assigned to them. Location Groups are basically saved lists of locations that can be selected when opening a form to make changes. For example, if you often select locations 2, 4, 6, 8, and 10, you could create a new location group with ID=-50 and NAME="Even Locations" and Transaction version=-1 (To prevent the generation of transaction files, add locations 2, 4, 6, 8, and 10 to it in this subform). When a form is opened, the user can select the "Even Locations" group instead of individually selecting locations 2, 4, 6, 8, and 10.
Add Locations To This Group	This drop-down list enables the user to select multiple sites to add to the site group

Location Data

Reference the pages in this section for information about managing data for remote locations.

Copying Location Data	Reference this page for information about copying remote location data
Sharing Location Data	Reference this page for information about sharing remote location data with other locations
Unsharing Location Data	Reference this page for information about unsharing remote location data
Deleting Location Data	Reference this page for information about deleting remote location data

Copying Location Data

In order to create data for a new location, you can copy data in the central database from one location to another, or set up a data location that will share its data with other locations.

To copy data in the central database from one location to another:

1. Select the *Sites* module.
2. Select *Copy Location Data* from the *Setup* menu.
3. Select the location with the data to copy from the *Available Sites* field.
4. Click the right arrow to move the selected location to the *Selected Sites* field.

- Click *OK*. The database tables you can copy are listed.
- Select the tables to copy. Hold down the Ctrl key on your keyboard to select multiple tables.

Option	Description
Overwrite existing data	If this option is selected, the data in the existing table is replaced with the copied data
Copy Table Sharing	If this option is selected, the table is shared at the destination site. Otherwise, the shared data is copied to the destination site, but the table is unshared.

- Click *OK*.




Sharing Location Data

When more than one location has the same data in one or more tables, it will improve performance and consistency to share one set of data between the locations. Whenever the shared data is edited, the changes are sent to all of the locations that share the data. For example, if five locations in the East Region all use the same set of items, you could create a share group named "East Region" and add the five locations to the share group. If an item is added, changed or deleted in the share group, the change will be sent to all five locations.

- [Setting Up Sharing](#)
- [Working with the Share Tree](#)
- [Creating a New Share Group](#)
- [Adding Locations to a Share Group](#)
- [Moving a Location to a Different Share Group](#)
- [Removing Locations from a Share Group](#)
- [Renaming a Share Group](#)
- [Deleting a Share Group](#)
- [Showing Recommended Share Groups](#)
- [Viewing Data Differences](#)
- [Accepting Recommended Share Groups](#)

Setting Up Sharing

To view or modify data sharing:

- Select the *Sites* module.
- Select *Edit Sharing* from the *Sharing* menu. Any existing packages are listed.
- Select an existing package from the list or click *New Package* to create a new package. If you choose to create a new package, a form opens where you define properties for the package including: name, description, effective and terminating date/time; select *Force Processing* if transactions should be processed when only processing forced transactions.
- Click *Finish*.
- Click the  beside a table name.
- Click the  beside 'Share Groups' to see the existing share groups, or click the  beside 'Unshared Locations' to see the locations that are not in a share group.

To setup sharing:

- Set the transaction version to -1 for any existing share locations and location groups that are not real remote locations. This will let the system know they are not real locations and transaction files will not be generated in their outgoing directories.
- Right-click a table and select *Show recommended groups*.
- Right-click a table and select *Accept all recommended groups* (they already have matching data).
- Click each group and location starting with the first one and review the differences between the recommended group in the difference window.
- If the differences are not important or if one group is incorrect, move the group with the incorrect data to the group with the correct data.
- Review the locations in each share group and rename the groups as appropriate by right-clicking the group and selecting *Rename share group*.

Working with the Share Tree

Click the  next to an item to display its contents. For example, click the  next to a table to display the share groups and unshared locations.

Click the  next to an item to hide its contents.

Right-click an item to see the options available for that item.

Creating a New Share Group

1. Right-click an unshared location, a recommended share group or a location in a share group.
2. Select *Create Share Group*.
3. You will be prompted to enter a name for the new share group. The selected locations will be moved into the new share group. The new share group will be saved as a location and will automatically be assigned a unique, negative location number.

Adding Locations to a Share Group

1. Select one or more unshared locations or recommended share groups.
2. Right-click one of the selected locations and select *Move to share group* from the popup menu.
3. Select the share group from the displayed list and click *OK*.

You can also click and hold the left mouse button on a location, drag the location to the desired share group and release the left mouse button. If there are any differences between the location's current data and the share group's data, transactions will be generated to make the remote location's data match the share group's data.

Moving a Location to a Different Share Group

To move locations to the recommended group, right-click the location and select *Move to recommended share group*. If the recommended group is another remote location, you will be asked if you want to create a share group from the recommended location and add this location to the new share group.

To move locations to a different share group:

1. Select one or more locations in a share group.
2. Right-click one of the selected locations and select *Move to share group* from the popup menu.
3. Select the share group from the displayed list and click *OK*.

You can also click and hold the left mouse button on a location, drag the location to the desired share group and release the left mouse button. If there are any differences between the location's current data and the share group's data, transactions will be generated to make the remote location's data match the share group's data.

Removing Locations from a Share Group

1. Select one or more locations in a share group.
2. Right-click one of the locations and select *Unshare*. Locations removed from share groups will be displayed in the *Unshared Locations* list.

Users can also click and hold the left mouse button on the location, drag the location to *Unshared Locations* and release the left mouse button. Location data is not changed when a location is removed from a share group. The location will now have its own copy of the table data and may be edited separately from the other locations in the share group.

Renaming a Share Group

1. Right-click a share group and select *Rename*.
2. Type in the new name and press *Enter* on your keyboard.

Note

To avoid confusion, do not use the same name for more than one share group in the same table.

Deleting a Share Group

Right-click the share group and select *Delete*.

If there are any locations in the share group, they will be removed from the share group and returned to the list of unshared locations. Location data is not changed when a location is removed from a share group.

Showing Recommended Share Groups

To view the share groups the system recommends for all locations for a table:

Right-click the table and select *Show recommended groups*. The analysis may take several minutes depending on the number of locations and the amount of data in each table.

To show recommended share groups for more than one table:

1. Select the desired tables.
2. Right-click one of the selected tables, and select *Show recommended groups*.

To show recommended share groups for selected unshared locations and share groups for a table:

1. Select the desired unshared locations and share groups.
2. Right-click one of the selected locations or groups, and select *Show recommended groups*.

The system will examine all the unshared locations and all the existing share groups to see which ones already have the same data and can be shared without changing the data. The unshared locations will be grouped into recommended share groups if they have the same table data as other unshared locations. For all share groups, recommended share groups and unshared locations, the name of the most similar location and the number of differences between the location and the most similar location will be displayed next to the location or group name.

Viewing Data Differences

After showing the recommended share groups, you can view the differences between a location or group and the most similar location or group. Click the location or group and the differences will be displayed in the report window below the tree.

You can also right-click a location or group and select *View Differences*. The data differences will be displayed so you can determine which locations should share data.

You do not have to manually change the data to make locations match before sharing them. The system will automatically make the necessary changes when a location is moved into a share group where the data does not match exactly.

Accepting Recommended Share Groups

To accept a recommended share group and make it a real share group:

1. Right-click the recommended share group and select *Create share group*.
2. Enter a name for the new share group when prompted. The new share group will be moved to the list of share groups for the table.

You can also click and hold the left mouse button on a recommended share group and drag it to *Share Groups* to accept the recommended share group.

To accept all recommended share groups: Right-click the table (or the *Unshared Locations* node or a recommended group) and select *Accept all recommended groups*.

Unsharing Location Data

If the data at one or more locations needs to be changed, but the data is shared by other locations that should not be changed, you will have to unshare the locations to be changed before updating the data.

To unshare data for one or more locations:

1. Select the *Sites* module.
2. Select *Unshare Location Data* from the *Sharing* menu. The *Unshare Location Data* form opens.
3. Select the locations that are no longer to get data from a data location. Hold down the *Ctrl* key on your keyboard to select multiple locations.
4. Click *OK*.
5. Select the tables that will no longer be shared.
6. Click *OK*.

Deleting Location Data

If a location was set up as a test location, you can delete some or all of the tables for the location. When deleting data, only the data for the selected locations will be deleted. Data for unselected locations will remain in the tables.

To delete location data:

1. Select the *Sites* module.
2. Select *Delete Location Data* from the *Setup* menu. The *Delete Location Data* form opens.
3. Select the locations to delete. Hold down the *Ctrl* key on your keyboard to select multiple locations.
4. Click *OK*.
5. Select the tables to delete.
6. Click *OK*.

Creating a Database for a New Remote Location

When you want to create a database for a new remote location, you can copy the data from a similar location and then make location-specific changes to the new data. The data for the various tables at the new location can come from different existing locations. Some of the new location's data may be specific to the new location, while other tables are shared with other existing locations.

To create a database for a new remote location:

1. Select the *Sites* module.
2. Use the *Copy Location Data* function from the *Setup* menu to copy location-specific tables from existing locations to the new location. You can use this function more than once to copy different tables from different locations to the new location.
3. Use the *Share Location Data* function from the *Sharing* menu to share data from existing locations with the new location.
4. Make sure all of the tables at the new location have either been either copied or shared from another location.
5. Use forms to make any location-specific changes to the data at the new location.

Sending a Database to a New Remote Location

Once you have set up the data for a new location in the central database, the data needs to be sent to the new location. This can be done in two ways depending on your requirements.

The first way is to create an application database from the data in the central database and physically send the entire application database to the new remote location.

The second way is to send the data to the new remote location in a transaction. Data can be sent to a location in a transaction if the location already has an application database, even if it is empty. Click [here](#) for instructions.

Sending Data to a New Remote Location in a Transaction

Follow these steps if the new remote location already has an application database installed and you just want to replace the data with the data from the central database using a transaction.

1. Select the *Transactions* module.
2. Select *Send Table Refresh* option on the *Processing* menu.
3. Select the location to refresh from the *Available Sites* field. Hold down the Ctrl key on your keyboard to select multiple locations.
4. Click the right arrow to move the selected location(s) to the *Selected Sites* field.
5. Click *OK*. The database tables you can refresh are listed.
6. Select the tables to refresh. Hold down the Ctrl key on your keyboard to select multiple tables.
7. Click *OK*.

Keeping Central and Remote Data Synchronized

This page how to synchronize the data at the central and the remote locations. Perform the following steps on a regular basis.

1. Transfer all the transaction files (.XML and *.FIL) between the central *Outgoingstorenum* and the remote *Incoming\0* directories, and between the remote *Outgoing\0* and the central *Incomingstorenum* directories.
2. At both the remote and central locations, select *Process All Transactions* from the *Processing* menu of the *Transactions* module, OR execute *ProcTran.bat* to process the transactions and update the database.

Usually these operations are performed daily during end-of-day processing. The system automatically creates the *Outgoing* directory and its subdirectories when it writes the transaction files. The *Incoming* directory and its subdirectories must be created manually or by using the file transfer program.

Data and Transactions

Reference the pages in this section for information about maintaining data, managing transactions, data recovery and import/export features. To return to the User Guide Home, click [here](#).

Maintaining Remote Data	Reference this page for information about maintaining the data for each table in the application database
Override Flags	Reference this page for information about override flags, which are used to specify how to handle database table conflicts
Transactions	Reference the pages in this section for information about transmitting/receiving/processing transactions and audit trails
Import-Export Features	Reference this page for information about the powerful and flexible import/export capabilities
Central Data Import	Reference this page for information about importing data into the central database from outside sources
International Language Support	Reference this page for information about maintaining multiple language versions of each text string
Data Recovery	Reference the pages in this section for information about recover data that is lost due to a system failure or error

Database Purge Process	Reference this page for information about the configurable database purge process. Purging the database prevents the database from reaching its maximum capacity and adversely affecting operations.
Validating Data	Reference this page for information about data validation

Maintaining Remote Data

You can choose who maintains the data for each table in the application database. Data for a table may be maintained by central users, remote users, central and remote users, or the remote system.

For example, you might setup EDM, so that the item table is maintained by central users, the employee table is maintained by remote users, the price table is maintained by both central and remote users, and the order table is updated by the remote system.

[Override Flags](#) can be assigned to each table to help the system enforce the decision about which users will maintain the table data. See [Override Flags](#) for more information.

Central Maintenance

When a table is to be maintained strictly by central users, changes made to the central database are transmitted to remote locations and applied to the remote database. Data changes may be made with forms at the central location or by using third-party applications, such as a menu editor. The [override flag](#) for the table should be set to either *Central Overrides on Conflict* or *Central Always Overrides*.

Remote Maintenance

When a table is to be maintained strictly by remote location users, changes may be made either with system forms or with the user's application tools, but not both.

If the data changes are made with system forms, the changes are automatically sent to the central location as transactions to update the central database.

If changes are made using the user's application tools, the changes must be captured by comparing snapshots of the data. In order to detect any changes made by third-party tools, a snapshot of the data is compared to the previous snapshot before the remote system communicates with the central system. Snapshots and system forms should not be used on the same table because the changes to the data would be reported by the forms, and then detected as additional changes by the snapshot comparison.

Central and Remote Maintenance

When a table is maintained by both central and remote users, the data *must* be maintained by using the system forms. By using the system forms, users can be assured that changes made either at the central or remote location are transmitted and processed, and the databases remain synchronized. The [override flag](#) for tables maintained by both central and remote users may be set to anything except *Central Always Overrides* as this would prevent remote changes from being accepted at the central location.

System Maintained Tables

Transaction tables, such as orders, are maintained by the application at the remote location and are not normally updated by direct user editing. Capturing changes to these tables and transmitting them to the central location is called polling. See [Polling Data](#) for more information.

Override Flags

Users can specify an override flag for each table to tell the system how to handle conflicts. The override flag for each table should be selected based on whether the table is maintained at the central or remote location, and how conflicts should be handled.

The following table describes the override flags.

Log	Logs conflicts in the audit trail and takes no further action
Undo	When a conflict occurs, the change is undone in an effort to resynchronize the databases. For example: The name of item 5 is changed from <i>Hamburger</i> to <i>Cheeseburger</i> at the central location. When the transaction is processed at the remote location, where item 5 is named <i>Milkshake</i> , a conflict message is sent to the central location where the name of item 5 will be changed back to <i>Hamburger</i> .
Central Overrides on Conflict	When a conflict occurs, the values in the central database override the values in the remote database. For example: The name of item 5 is changed from <i>Hamburger</i> to <i>Cheeseburger</i> at the central location. When the transaction is processed at the remote location, where item 5 is named <i>Milkshake</i> , item 5 will be changed to <i>Cheeseburger</i> anyway.

Central Always Overrides	This flag is the same as <i>Central Overrides on Conflict</i> with the additional feature that if a change is made at the remote location, then the transaction will be rejected at the central location and sent back to the remote location to be undone. For example: The name of item 5 is changed from <i>Hamburger</i> to <i>Cheeseburger</i> at a remote location. When the transaction is processed at the central location, it is rejected and a conflict message is sent to the remote location, where the name is changed back to <i>Hamburger</i> .
Remote Overrides on Conflict	When a conflict occurs, the values in the remote database override the values in the central database. For example: The name of item 5 is changed from <i>Hamburger</i> to <i>Cheeseburger</i> at a remote location. When the transaction is processed at the central location, where item 5 is named <i>Milkshake</i> , item 5 will be changed to <i>Cheeseburger</i> anyway.

Transactions

Overview

The flow of data and transactions through the system generally follows this progression:

1. *Editing* data using a central table or form updates the central database and saves the transaction in the audit trail.
2. *Committing* transactions writes the transactions in the outgoing folder as XML files.
3. *Transferring* files moves transactions from the outgoing folder on one system to the incoming folder on another system.
4. *Processing* transactions moves all transactions from XML files to the audit trail, then applies all transactions in the audit trail that either have no effective date or whose effective date has arrived. As transactions are applied, status transactions are automatically committed to notify the sender about the outcome of the transaction.

Topics

[Data Changes and Transaction Files](#)
[Generating Transactions](#)
[Transmitting Transactions](#)
[Receiving Transactions](#)
[Processing Transactions](#)
[Recording Conflicts](#)
[Supporting Multiple Application Versions](#)
[Audit Trails](#)
[Transaction File Numbers](#)
[Reset Transactions](#)

Data Changes and Transaction Files

Whenever changes are made to the data at a central location the changes need to be sent to all the affected remote locations. Changes made at remote locations are communicated to the central location. The system accomplishes this by recording the data changes in transaction files.

Generating Transactions

When records are added, changed or deleted, the record information is written to a transaction file, which is stored in the location directory for the location where the data is to be sent.

For example, if a new menu item is added at remote location 21 and it needs to be sent to central location 100, then a transaction file is created in the directory for central location 100.

Transmitting Transactions

The system provides tools to transfer transaction files between central and remote sites. You may also elect to use your existing file transfer tools.

The system provides three transfer tools:

- One works when there is a network connection between remote and central sites
- One works by using File Transfer Protocol (FTP)
- One uses HTTP when the central system Web server is visible to the remote sites over the network, or over the Internet

See [File Transfer](#) for more information.

Receiving Transactions

As part of the End-of-Day procedure (or more frequently if necessary), the system needs to process the received transaction files.

This is accomplished by running the *Receive Transactions* function, which can be run in one of the following ways:

- From a user-designed form

- From a command-line as a startup expression

The *Receive Transactions* function reads all the transaction files in the current location's directory and loads them into the pending transaction file, where they will wait until their specified effective date.

Processing Transactions

As part of the End-of-Day procedure (or more frequently if necessary), the system needs to read through the pending transaction file and apply the transactions to the database that have reached their specified effective date.

Processing All Transactions

1. Select the *Transactions* module.
2. Select *Process All Transactions* from the *Processing* menu.

Processing Selected Transactions

1. Select the *Transactions* module.
2. Select *Process Selected Transactions* from the *Processing* menu.
3. Select the locations with the transactions to process from the *Available Sites* field. Hold down the Ctrl key on your keyboard to select multiple locations.
4. Click the right arrow to move the selected locations to the *Selected Sites* field.
5. Click *OK*. The transactions you can process are listed.
6. Select the transactions to process. Hold down the Ctrl key on your keyboard to select multiple transactions.

The processing transactions function can also be accomplished:

- From a user-designed form
- From a command-line as a startup expression

Recording Conflicts

Sometimes conflicts can arise because the same record was changed at a central location and a remote location at the same time. When this occurs, a conflict record is added to the [audit trail](#) and the change is undone.

For example, the following sequence generates a conflict:

1. Location 1 changes the price of item 50 from 1.99 to 1.89
2. Central location changes price of item 50 at location 1 from 1.99 to 1.79
3. Transaction files are transmitted

When the transaction from the central location is processed, it fails because the price for item 50 is not 1.99 as expected, but 1.89. A conflict record is added to the [audit trail](#) and a status transaction is sent to notify the central location of the error.

Supporting Multiple Application Versions

When a new version of a user application is installed, it often includes some changes to the structure of the database, such as new tables and new fields. Since a company may have many locations and may not upgrade all of them in a single day, data coming from a location with an older version of the database needs to be converted to the current version. Conversely, when transactions are sent from the central location to a remote location with an older version of the database, the data needs to be reverted from the current version to the older version.

Note

The current version of the database must be installed at the central location first before it can be installed at any remote locations.

The system includes a data conversion file that needs to be updated whenever a new version of the database is installed at the central location. This file contains information about how to convert a record from a table from the older version to the newer version and vice versa. This conversion file is usually developed by the application vendor.

Audit Trails

Every change to the database is recorded in an audit trail, including inserted/updated/deleted records, as well as conflict and conflict undo records, which are generated when a conflicting transaction is removed from the database.

Viewing Audit Trail Data

1. Select the *Transactions* module.
2. Select *View Transactions* or *Run Transaction Report* from the *Processing* menu.

Transaction File Numbers

Transactions sent between central and remote locations are written in files called transaction files. Each transaction file name is a number with either a .XML or .FIL extension. For example, the first transaction sent after installing the system is usually called 1.XML.

Transaction Processing Sequence

The system must process the transactions in order to avoid errors. For example, if you change the price of an item to 1.89 and then again to 1.99, and the transactions are processed in the wrong sequence, then the final price would be 1.89 instead of the expected 1.99.

In order to process the transactions in the proper sequence, the system remembers the last transaction file number that it received and sent to each location. Central systems remember the received and sent transaction file numbers for each remote location. Remote locations remember the received and sent transaction file numbers for the central location.

Transaction File Numbers

The transaction file numbers are stored in the *CDMLocation* table in the fields *SENTTFN* (last sent transaction file number) and *RCVDTFN* (last received transaction file number). At the central location, the last sent and received transaction file numbers are stored in the location records for each remote location. At remote locations, the last sent and received transaction file numbers are stored in the location record for the central location (typically the record where *CDMLOCID* is 0).

If the system is expecting transaction file number 1, but there is no 1.XML or 1.FIL in the incoming directory, then the system will report that there are no transactions pending even if there are other transaction files in the incoming directory. Whenever you see the message "No transactions pending" when there are transactions in the incoming directory, it means that the system is expecting a transaction file number that is not there. EDM processes transactions in order, so it will not process transaction file 2.XML before it has found and processed transaction file 1.XML. If a transaction file is lost or deleted (or EDM is reinstalled and loses track of which transaction file to expect next), then the system will report that there are no transactions pending. In this situation, you need to adjust the transaction file number it is expecting.

See Also

Changing the Incoming Transaction File Number at Remote Location

1. At the remote location, start EDM by running *go.bat*.
2. Login as a user with permission to edit the *CDMLocation* table.
3. Select the *Setups* module.
4. Expand the *Application>Tables* menu, and then select the *CDMLocation* table
5. Scroll to the *RCVDTFN* (last received transaction file number) column on the *Headquarters* row (usually ID 0).
6. Change the *RCVDTFN* value to 1 less than the lowest-numbered transaction file in the *Incoming\0* directory. For example, if the *Incoming* directory contains 5.XML and 6.XML, set *RCVDTFN* to 4 so the system believes the last transaction file processed was 4.XML.
7. Close the location table and process transactions as normal.

Changing the Incoming Transaction File Number at Central Location

1. Select the *Setups* module.
2. Expand the *Application>Tables* menu, and then select the *CDMLocation* table.
3. Scroll to the *RCVDTFN* (last received transaction file number) column on the row where *CDMLOCID* is the remote location ID.
4. Change the *RCVDTFN* value to 1 less than the lowest-numbered transaction file in the *Incoming\0* directory (where n is the remote location ID). For example, if the *Incoming* directory contains 5.XML and 6.XML, set *RCVDTFN* to 4 so the system believes the last transaction file processed was 4.XML.
5. Close the location table and process transactions as normal.

Changing the Outgoing Transaction File Number at Remote Location

1. At the remote location, start EDM by running *go.bat*.
2. Log in as a user with permission to edit the *CDMLocation* table.
3. Select the *Setups* module.
4. Expand the *Application>Tables* menu, and then select the *CDMLocation* table
5. Scroll to the *SENTTFN* (last sent transaction file number) column on the *Headquarters* row (*CDMLOCID* 0).
6. Change the *SENTTFN* to 1 less than the next transaction file number that should be used when sending transactions to the central location. For example, if the next transaction file to send to central should be 7.XML, then set *SENTTFN* to 6 so the system believes the last transaction file sent was 6.XML.
7. Close the location table.

Changing the Outgoing Transaction File Number at Central Location

1. Select the *Setups* module.
2. Expand the *Application>Tables* menu, and then select the *CDMLocation* table
3. Scroll to the *SENTTFN* (last sent transaction file number) column on the row where *CDMLOCID* is the remote location ID.

4. Change the *SENTTFN* value to 1 less than the next transaction file number that should be used when sending transactions to the remote location. For example, if the next transaction file to send to the remote location should be 7.XML, set *SENTTFN* to 6 so the system believes the last transaction file sent was 6.XML.
5. Close the location table.

Remote and Central Transaction File Processing

Remote agent

When the Remote agent contacts the Central location, it performs a series of operations using "Commands". These commands are Java Beans with very little logic in them at the Remote location with a counterpart at the Central location that does all of the work.

The Remote agent performs the following:

1. Instantiates the object.
2. Sets the data.
3. Sends it to Central for processing.

The Remote agent always initiates the connection to Central. Central never contacts the Remote agent directly.

Batch script

The Remote agent is usually initiated via a batch script that is executed as part of End-of-Day operations at the store.

When it is run, the Remote agent performs the following:

1. Logs into Central (**LoginCmd**).
2. Requests the location of the outgoing transaction folder (**GetTranFolderCmd**). This is a site specific folder based on the site's ID, which is a number that is unique per Remote site.
3. Requests outgoing files from that folder (**GetFileListCmd**).
4. Retrieves the name & size of each file that it needs to download (**GetFileInfoCmd**).
5. Reads each file from Central (**ReadFileBytesCmd**) reading only small blocks of data at a time.
6. Once the file is transferred to the store, it is deleted from Central (**DeleteFileCmd**).
7. The file is processed locally by the client - transactions are imported into the CDMAUDIT table and data is updated in the database.

Transaction types

The transaction files may contain multiple transactions (e.g. inserts, updates, deletes). The following are the different transaction types that the remote client knows how to process.

```
/** Status report on the success or failure of a transaction. */
STATUS_UPDATE( 0, "statusUpdate", Msg.get("Status Update") ),

/** Insert a row in a table. */
INSERT_ROW( 1, "insertRow", Msg.get("Insert Row") ),

/** Update a row in a table. */
UPDATE_ROW( 2, "updateRow", Msg.get("Update Row") ),

/** Delete a row in a table. */
DELETE_ROW( 3, "deleteRow", Msg.get("Delete Row") ),

/** Send a file. */
SEND_FILE( 4, "sendFile", Msg.get("Send File") ),

/** Delete a file. */
DELETE_FILE( 5, "deleteFile", Msg.get("Delete File") ),

/** Rename a file. */
RENAME_FILE( 6, "renameFile", Msg.get("Rename File") ),
```

```
/** Request a file. */
REQUEST_FILE( 7, "requestFile", Msg.get("Request File") ),

/** Execute a command-line command. */
EXECUTE( 8, "execute", Msg.get("Execute") ),

/** Refresh data in tables. */
REFRESH_TABLE( 9, "refreshTable", Msg.get("Refresh Table") ),

/** Request a table refresh. */
REQUEST_TABLE( 10, "requestTable", Msg.get("Request Table") ),

/** Reset a site. */
RESET( 11, "reset", Msg.get("Reset") ),

/** Report an error. */
ERROR( 12, "error", Msg.get("Error") ),

/** Request a schema file from a remote site. */
GET_SCHEMAS( 13, "getSchemas", Msg.get("Get Schemas") ),

/** Add a schema file from a remote site. */
ADD_SCHEMAS( 14, "addSchemas", Msg.get("Add Schemas") ),

/** Send polling data. */
SEND_POLL_DATA( 15, "sendPollData", Msg.get("Send Poll Data") ),

/** Evaluate an expression on the server (only uses server functions). */
EVALUATE( 16, "evaluate", Msg.get("Evaluate Expression") ),

/** Synchronize a local directory with a directory at other sites. */
SYNC_DIRECTORY( 17, "syncDirectory", Msg.get("Synchronize Directory") ),

/** Get committed transactions for a remote site. */
```

```
GET_COMMITTED( 18, "getCommitted", Msg.get("Get Committed Transactions")
);
```

Central processing

After the file is processed at Remote, status updates are sent back to the Central location (**WriteFileBytesCmd**). The processing steps that occur at the Central location include the following:

1. The status updates are written to temp files.
2. The files are renamed (**RenameFileCmd**).
3. The incoming files are processed on a schedule. Central updates the audit table (CDMAUDIT) with the various statuses that can occur on the Remote side.

Status types

```
/** Open */
OPEN( "Open", 0, Msg.get( "Open" ) ),
/** Committed */
COMMITTED( "Committed", 1, Msg.get( "Committed" ) ),
/** Sent */
SENT( "Sent", 2, Msg.get( "Sent" ) ),
/** Received */
RECEIVED( "Received", 3, Msg.get( "Received" ) ),
/** Successful */
SUCCESSFUL( "Successful", 4, Msg.get( "Successful" ) ),
/** Failed */
FAILED( "Failed", 5, Msg.get( "Failed" ) ),
/** Failed, undone */
UNDONE( "Undone", 6, Msg.get( "Undone" ) ),
/** Failed, undo failed */
UNDO_FAILED( "UndoFailed", 7, Msg.get( "Undo Failed" ) ),
/** Marked closed by user. */
CLOSED( "Closed", 8, Msg.get( "Closed" ) );
```

In some cases Central will have to take additional actions, which will result in more transaction files that Remote will pick up the next time they are processed. The scope of these transaction and status types has been kept narrow in order to keep the Remote agent from needing to be upgraded at the sites. There are sometimes thousands of remote sites, and as long as these transaction and status types do not change, the Remote agent does not need to be upgraded.

Reset Transactions

Overview

The *Reset* transaction is available to help you resolve problems with remote sites without having to manually connect to the site and manipulate the system.

Reset Transaction

When a *Reset* transaction for a remote site is committed at the central site, it has the following effects:

- The last sent and received transaction file numbers for the remote site are set to 0.
- All outgoing transaction files for the remote site are deleted.

When a *Reset* transaction is processed at a remote site, it has the following effects:

- Drop and recreate the system database tables, add the transactions in the *Reset* package that follow the *Reset* transaction back into the audit trail for processing after the *Reset*.
- Reload the remote site number.
- Delete and reload the application table structures.
- Reset the last sent and received transaction file number.
- Delete all incoming and outgoing transaction files (incoming transaction files sent after the *Reset* transaction will be processed before they are deleted).

In other words, it resets the remote system to the condition it was in just after it was installed. A *Reset* transaction received at a central site will be ignored and deleted.

Reset Package

A *Reset* package is a package that contains a *Reset* transaction. It is sent with the special name of *reset.xml*. When transactions are processed, if there is a *reset.xml* in the incoming folder, it will be processed first.

Within the *Reset* package, you can include any number of transactions of any type. Typically, there will be a *Reset* transaction to reset the remote site to its freshly installed state and a refresh table transaction to synchronize the site data with the data in the central database. The transactions in the *Reset* package can be in any order and *Reset* transactions may be included more than once if desired.

Issues Resolved by Reset

Sending a *Reset* transaction and a *Table Refresh* transaction in a *Reset* package to the remote site will resolve the following issues:

- Transaction file numbers out-of-synch with central
- Incoming transaction file can't be read preventing further processing
- Duplicate transaction number in incoming transaction causes a duplicate key error in the audit trail preventing further processing
- Transaction file is deleted or lost in transmission preventing further processing at the remote site
- Transaction in *Incoming* directory with a wrong site number

Using the *Send File* transaction to send to the remote site a new *system.properties* file and table list, followed by a *Reset* transaction in a *Reset* package will resolve the following issues:

- Remote site has an error in *system.properties*
- Remote site needs an updated table list to process additional or updated tables

Using the *Send File* transaction to send to the remote site a new *system.properties* file and table list, followed by a *Reset* transaction and a *Request Table Refresh* transaction in a *Reset* package will resolve the following issue:

- Central system needs to get data from a table that is not currently in the table list at the remote site

Reset transactions provide central users with a powerful tool for managing and updating remote sites.

Sending a Reset Transaction

1. Select the *Transactions* module.
2. Select *Reset Site* from the *Remote PC* menu.
3. Select the location to which the transaction will be sent.

When the package containing the *Reset* transaction is committed, the file *reset.xml* will be written in the *Outgoing* directory instead of the usual file name *<transaction file number>.xml*.

Import-Export Features

The system supports powerful and flexible import/export capabilities. The functions can be used to read and write ASCII files of data that can be exchanged with outside systems, such as payroll processors, vendors, etc.

Import Transactions From File and *Export Data To File* are available from the *System Data* menu of the *Setup* module, or may be called as functions. See [Client Functions](#) for more information about importing and exporting data.

Central Data Import

Importing data into the central database from outside sources is a valuable capability of the system. For example, users may want to feed price updates or new items into the system from external sources and have transactions sent to the affected sites to make the changes there. The following discusses the requirements for this functionality.

Import File Format

The import file is a comma-delimited ASCII file that consists of:

- A header row that contains the column names
- Additional rows that contain the data for the rows to be inserted, updated or deleted.

Each row in the file contains the header columns listed in the next section in addition to whatever table columns are needed to perform the transaction.

Transaction Header Columns

Column Name	Data Type	Data Length	Decimal Digits	Description
SiteID	Integer			ID of location to update
TransactionType	String	7		"INSERT", "UPDATE", or "DELETE"
PackageName	String	100		Package name
PackageDescription	String	250		Package description
EffectiveDate	DateTime	20		The date the change should be made at the site (NULL indicates 'effective immediately')
UserID	String	20		ID of user that generated the transaction
ChangeDate	DateTime	20		Date the change was requested
TableName	String	64		Name of the table being updated

Sample Table Columns: IRIS Prices (tbl_ItemPricing) in Central Database

Column Name	Data Type	Data Length	Decimal Digits	Description
ItemNum	Integer			Unique item number for price
TimePeriod	Integer			ID of price daypart (1 indicates All)
Std	Number	19	4	Standard price
Sun	Number	19	4	Sunday price (defaults to standard price if NULL)
Mon	Number	19	4	Monday price (defaults to standard price if NULL)
Tue	Number	19	4	Tuesday price (defaults to standard price if NULL)
Wed	Number	19	4	Wednesday price (defaults to standard price if NULL)
Thu	Number	19	4	Thursday price (defaults to standard price if NULL)
Fri	Number	19	4	Friday price (defaults to standard price if NULL)
Sat	Number	19	4	Saturday price (defaults to standard price if NULL)
Destination	Integer			Not currently used (default is 0)

Row Formats

The first row contains the column names surrounded by double quotes and separated by commas. All the header columns must be included. Not all of the table columns must be included, however the identifying columns are required.

The following is an example where we are updating the *Std* column to \$5.99 in the price table at site 1234. The identifying columns are the "ItemNum" 456 and "TimePeriod" 7:

```
"SiteID","TransactionType","PackageName","PackageDescription","EffectiveDate","UserID","ChangeDate","tableName""ItemNum","TimePeriod",  
"Std" 1234,"UPDATE","A package","A package description","Fred","2008-04-21 23:00:00.000","tbl_ItemPricing",456,7,5.99
```

Column Value Formats

String values are surrounded by quotes if they are not NULL. Double-quotes (") inside strings must be represented by two double-quote characters ("). An empty string is represented by "".

Numeric values are not surrounded by double-quotes.

Date values are surrounded by double-quotes if they are not NULL and follow the format: 'yyyy-MM-DD'.

DateTime values are surrounded by double-quotes if they are not NULL and follow the format: 'yyyy-MM-DD HH:mm:ss:iii' (The time is presented in military time, where iii is the number of milliseconds).

NULL values are represented by not putting any characters at all in the data.

Examples

```
1,"My package name","My package description","2008-04-21 23:00:00.000","Fred","2008-04-25 23:00:00.000",1234,1,5.99,0,0,0,0,0,0,0
```

Example with a null package description

```
1,"My package name",,"2008-04-21 23:00:00.000","Fred","2008-04-25 23:00:00.000",1234,1,5.99,0,0,0,0,0,0,0
```

Example with embedded quotes in the package description

(My "package" description):

```
1,"My package name","My ""package"" description","2008-04-21 23:00:00.000","Fred","2008-04-25 23:00:00.000",1234,1,5.99,0,0,0,0,0,0,0
```

Processing the Data

The *Import* function can be started in a browser by calling the *importTransactionsFromFile(filename)* function. In order to be imported, the filename must exist on the server and any path information in the filename must be valid for the server.

When the system reads the file, it updates the central database appropriately and creates transactions to send to the affected sites so that the same updates are made at the sites.

Updates that generate errors, such as trying to update a row that doesn't exist, are logged and ignored.

International Language Support

All text strings in the system can be loaded from a file. This file may contain multiple language versions of each text string. This file needs to be translated along with the forms whenever the system needs to be translated into a new language.

For full Unicode support including multi-byte Asian languages, you must use a Microsoft SQL Server 2000 (or later) database and set the *DBTYP E* properties in *system.properties* to *SQLSERVER_JDBC*. See [Supported Databases](#) for more information about configuring the system to support Unicode.

Data Recovery

The pages in this section describe the procedures to recover data that is lost due to a system failure or error. It is always prudent to backup your system and application data at both the central and remote locations to protect your valuable data and simplify recovery when problems occur.

Data Types

There are several types of data maintained by the system.

Central System Data	Internal data, such as location and audit records at the central location
Central Application Data	Application data from the various locations stored in the central database
Remote System Data	Internal data, such as location and audit records at a remote location
Remote Application Data	Application data at the remote locations in the application databases
Transaction Files	Files containing database update information sent between central and remote locations

See Also

[Recovering Central System Data](#)

[Recovering Central Application Data](#)

Recovering Central System Data

Recreated Database Tables

In the event of accidental data loss with no backup, there are a number of system tables maintained in the central database that may need to be recreated. The following tables will be automatically recreated if the system database and the *schema*(app version).xml_ file in the configuration files folder are deleted and the system is restarted.

- CDMAudit
- CDMLocation
- CDMLocationGroup
- CDMPackage
- CDMRemoteFiles
- CDMRemoteTables
- CDMValue

Database Recovery

Deleting the system database also deletes all the location data stored in the central database, which means that the central application data will also have to be recovered. The following describes the recovery procedures for tables that are not restored by deleting the system database and schema file.

CDMAudit	This table cannot be recovered. The system can still be used, but the transaction history will be lost.
CDMLocation	The location records must be reentered if this table is lost. Usually, the transaction file number will also need to be set. See Transaction File Numbers for more information.
CDMLocationGroup	Location groups will need to be reentered if they are lost.
CDMPackage	Packages cannot be recovered. New packages may be created once the system is running. If data in the <i>CDMAudit</i> table was not lost, then a query can be run to determine all open transactions (status=0) in a package other than the default package (package 0), and the packages can be manually entered into the <i>CDMPackage</i> table.
CDMRemoteFiles	Records in this table must be reentered if lost.
CDMRemoteTables	Records in this table may be recovered by requesting a table refresh from each location. When this table is updated by a refresh table transaction, it is assumed the location is not setup for sharing. If the locations were originally getting some of their data from share locations, the sharing will have to be setup again.
CDMValue	This table contains the current transaction number and is restored by manually entering a value higher than the highest-known transaction number used by the system. To determine the most recent transaction numbers, check the transaction numbers at remote locations where transactions were recently sent.

Recovering Central Application Data

If the entire central database is lost, the application data for each location can be recovered by requesting a table refresh from each location.

1. Select the *Sites* module.
2. Select *Edit Locations and Groups* from the *Setup* menu to add the missing location records to the central database.
3. Select the *Transactions* module.
4. Select *Request Table Refresh* from the *Processing* menu.
5. Select the snapshot that contains the tables with the desired data.
6. Select the locations from which the data is to be requested.
7. Select *0-Default Package* to generate the request transactions immediately.

When the transactions are processed at the remote locations, the requested data is sent back to the central location and saved in the central

database.

The transaction file numbers for each location may need to be adjusted since transaction file numbers from the remote location will probably be greater than 1, but the central system will be expecting transaction file 1 because the location record was just reentered. To adjust the transaction file number, so that the central system will process the incoming transactions from the remote location, see the section on [Transaction File Numbers](#).

When a location that does not exist in the central database is loaded by requesting a table refresh, the location will not participate in any data sharing. If the location was getting data from a shared location before the data recovery operation, then the sharing will need to be set up again.

Recovering Remote System Data

Recreated Database Tables

There are a number of system tables maintained in the system database at the remote location that may need to be recreated in the event of accidental data loss and no backup. The following tables are automatically recreated if the system database and the *schema*(app version).xml_ file in the configuration files folder are deleted and the system is restarted.

- CDMAudit
- CDMLocation
- CDMLocationGroup
- CDMPackage
- CDMRemoteFiles
- CDMRemoteTables
- CDMValue

Database Recovery

The following describes the recovery procedures for tables that are not restored by deleting the system database and schema file.

CDMAudit	This table cannot be recovered. The system can still be used, but the transaction history will be lost.
CDMLocationGroup	This table does not need data at a remote location.
CDMPackage	Packages cannot be recovered. New packages may be created once the system is running. If data in the <i>CDMAudit</i> table was not lost, then a query can be run to determine all open transactions (status=0) in a package other than the default package (package 0), and the packages can be manually entered into the <i>CDMPackage</i> table.
CDMRemoteFiles	Records in this table must be reentered if lost.
CDMValue	This table contains the current transaction number and is restored by manually entering a value higher than the highest-known transaction number used by the system. To determine the most recent transaction numbers, check the transaction numbers at remote locations where transactions were recently sent.

Recovering Remote Application Data

When data in an application database at a remote location is lost, it can be restored if it is maintained at the central location by sending a table refresh transaction from the central location to the remote locations.

To send a table refresh to a remote location:

1. Select the *Transactions* module.
2. Select *Send Table Refresh* from the *Processing* menu.
3. Select the snapshot that contains the tables with the desired data.
4. Select the locations where the data is to be sent.
5. Select *0-Default Package* to generate the refresh transactions immediately.

When the refresh transactions are processed at the remote locations, the data in the remote database is updated to match the data sent from the central database. If some of the data in the refresh table transaction already exists in the remote database, the duplicate data is ignored.

If the location records at the remote location were damaged or replaced, the transaction file numbers for each location may need to be adjusted since the transaction file number that is expected at the remote location may be different than the transaction file number that is sent by the central location. To adjust the transaction file number so that the central system will process the incoming transactions from the remote location,

see the section on [Transaction File Numbers](#).

Recovering Transaction Files

If generated transaction files are lost before they are received and processed, the transaction information must be sent again. In the case of *Insert*, *Update* and *Delete* transactions, a table refresh should be sent to synchronize the data in the affected tables. All other types of transactions can be resent by generating a new transaction of the same type.

Database Purge Process

This page provides information about the configurable database purge process for EDM. Purging the database prevents the database from reaching its maximum capacity and adversely affecting operations. Click [here](#) to return to the Data and Transactions page. Click [here](#) to return to the EDM User Guide home page.

Overview

The database purge process is performed via a SQL script that can be automatically run on a regular basis by EDM. You can configure EDM to automatically execute this script at a specified time on either a daily or weekly basis. You can also configure the process to retain a specified number of months of data in the database.

Configuration

The database purge process is configured in the EDM Remote configuration file [Config.xml](#).

If you are using XPRESS 4.5 (or later), [Config.xml](#) is located in the following folder: `\EDM\release\config\POSTypes\XPRESS\Versions\4.5.0\remote`.

To configure the purge process, follow these steps:

1. Locate the following text in the *Config.xml* file:

```
<schedule
    frequency = "WEEKLY"
    time = "6/2/2013 4:00 AM"
    task = "executeDirectSql( 'TNG', 'exec usp_purge_TngDB 13' )"
/>
```

2. Define the following parameters. The parameter values specified in the example above are the default values. In the example above, the database purge process begins on 6/2/2013 at 4:00 am. It will occur once a week at this time thereafter. All relevant table records older than 13 months from the date the process is run will be purged from the TNG database.

Parameter	Set the value to	Default value
frequency	"DAILY" - To purge the database daily "WEEKLY" - To purge the database weekly	WEEKLY
time	The starting date for the purge process and the time of day to execute the purge process	"6/2/2013 4:00 AM"
task	The number of months of data in the database to retain. Only edit the number that appears at the end of this parameter value. The default value is 13. <div style="border: 1px solid black; padding: 5px; margin-top: 10px;">The 'TNG' value in the example above indicates that the name of the database to purge is TNG.</div>	"executeDirectSql('TNG', 'exec usp_purge_TngDB 13')"

3. Save your changes to the *Config.xml* file.

Validating Data

Data Validation Report

This report validates the relationships between the selected tables plus all the tables that refer to the selected tables and all the tables to which the selected tables refer. For example, if validating the price table and a price row for item 123 exists but item 123 does not exist, it will generate a warning. Warning reports are displayed in a new browser window so they can be printed or emailed.

The validation process ignores uncommitted transactions and transactions with future effective dates so it is checking the data that the remote sites will actually be using once they process any committed transactions. This means that the validation report should be run after committing transactions and any problems should be fixed immediately to prevent invalid data from going to the remote sites.

The validation report can be run from the menu by selecting **Sites - Validate Data**. To add the validation report to your menu, add a new menu item and set the **Name** to **Validate Data** and set the **Action** to **validateTables("", "")**.

To automatically run the validation report at a scheduled time, add a schedule element to the config.xml file on the central server that calls either `validateTables` or `validateTableList` like one of these examples:

```
<schedule frequency = "DAILY" time = "1:00 AM" task = "validateTables( '*', '*' )" />
<schedule frequency = "DAILY" time = "1:00 AM" task = "validateTableList( 'Menu Tables', '1' )" />
<schedule frequency = "DAILY" time = "1:00 AM" task = "validateTables( 'IRIS_dbo_tbl_ItemAttachments, IRIS_dbo_tbl_ItemMaster', '1' )" />
```

The functions that perform this functionality are:

- [validateTableList](#)
- [validateTables](#)

Editing Data

The system makes it easy to add and change your data. You can add, change and delete records on forms or spreadsheets. Data is stored in tables and you can display and edit the data in several different ways.

Edit Data in Tables	Reference this page for information about editing data in database tables
Edit Data in Queries	Reference this page for information about editing records selected by a query
Editing Data on Forms	Reference this page for information about editing data using forms
Moving Between Records	Reference this page for information about navigating records on both forms and spreadsheets
Changing the Column Width on Spreadsheets	Reference this page for information about changing the column width on spreadsheets
Adding Records	Reference this page for information about adding new records to the database
Editing Records	Reference this page for information about editing records on forms and spreadsheets
Deleting Records	Reference this page for information about deleting records on a form or spreadsheet
Saving Records	Reference this page for information about saving updated records
Undoing Changes	Reference this page for information about undoing the changes you made
Finding Records	Reference this page for information about searching for records
Editing Central Data	Reference this page for information about editing data for multiple remote locations at a central location
Editing Data with Other Applications	Reference this page for information about editing data using other applications
Testing Data Changes Before Transmission	Reference this page for information about testing your changes before sending the data

Edit Data in Tables

You can edit tables directly by simply clicking the table in the table list. The table is displayed as a spreadsheet with one row for each record and

one column for each field.

When you directly edit a table at a central location that contains data for remote locations, you will notice that there is a location ID field called *CD MLOCID* as the first field in every record. You can edit data normally and all changes will be sent to the appropriate remote location.

	GROUPID	SITEID
*	-2	2185
	-2	2214
	-2	2235
	-2	2287
	-2	2402
	-1	2101
	-1	2103
	-1	2105
	-1	2106
	-1	2107
	-1	2112
	-1	2115
	-1	2118

Edit Data in Queries

You can edit the records selected by a query by clicking the query in the query list, which displays a spreadsheet with the results of the query.

	CDMLOCID	RegisterNum	DrawerNum	Description	StartAmount	PullAmount	MinAmount
*	2101	1	1	Reg 1 Drawer 1	135	435	50
	2101	2	1	Reg 2 Drawer 1	135	435	50
	2101	3	1	Reg 3 Drawer 1	135	435	50
	2101	4	1	Reg 4 Drawer 1	135	435	50
	2101	5	1	Reg 5 Drawer 1	135	435	50
	2101	6	1	Reg 6 Drawer 1	135	435	50
	2101	9	1	Reg 9 Drawer 1	135	435	50
	2103	1	1	Reg 1 Drawer 1	135	435	50
	2103	2	1	Reg 2 Drawer 1	135	435	50
	2103	3	1	Reg 3 Drawer 1	135	435	50
	2103	4	1	Reg 4 Drawer 1	135	435	50
	2103	5	1	Reg 5 Drawer 1	135	435	50
	2103	6	1	Reg 6 Drawer 1	135	435	50
	2103	9	1	Reg 9 Drawer 1	135	435	50
	2105	1	1	Reg 1 Drawer 1	135	435	50

Editing Data on Forms

You can set up forms that display one record at a time for editing. With forms, you can control the layout of the data entry screen to make it more usable. Also, you can include colors, pictures, graphs and other objects to enhance the appearance of your application and improve its usability.

The forms you can edit are located inside the *Applications* menu of the *Setup* module.

When editing a form at a central location that contains data for remote locations, you will be prompted before the form opens to name the package and define the effective date and termination date for the data changes.

If you enter an *Effective* date/time, the changes you make while using the form will not go into effect until the effective date/time is reached. If this field is left blank, the changes will take effect immediately.

If you enter a *Terminates* date/time, the data will revert back to its original state once the termination date/time arrives. For example, if the price of Coffee is changed from \$1.99 to \$1.89 with the termination date set to one month from today, then in one month the price will be changed back to \$1.99. If this field is left blank, the change is not reverted.

Select *Force processing* if transactions should be processed when only processing forced transactions.

General Ledger x

New Package:

Name:

Description:

Effective: ▼

Terminates: ▼

Force processing

After clicking *Next*, you will be prompted to select the sites for the data changes. Select the applicable sites from the *Available Sites* field, and then click the right-arrow to move the selected sites to the *Selected Sites* field.

General Ledger x

Select sites for package: Package 12 (admin 2013-08-20 17:07:21.521)

Group by... ▼ No Groups Selected Remove Group

Available Sites

- Available
- Share Groups
- Location Groups
- Locations

Selected Sites

- Selected
- Share Groups
- Location Groups
- Locations

2201 - 2201 - Greenville #1, SC

2202 - 2202 - Anderson #1, SC

2203 - 2203 - Columbia #1, SC

2204 - 2204 - Greenwood #1, SC

2205 - 2205 - Spartanburg #1, SC

2206 - 2206 - Aiken #1, GA

2207 - 2207 - Columbus #1, GA

2208 - 2208 - Thomas #1, GA

2209 - 2209 - Albany, NY

2210 - 2210 - Chatsworth, TN

2211 - 2211 - Greer, SC

2212 - 2212 - City View, TN

2213 - 2213 - Columbus #2, GA

2214 - 2214 - Columbus #2, GA

2215 - 2215 - Sumner #1, SC

2216 - 2216 - Seneca, AL

2217 - 2217 - Waseca, AL

2218 - 2218 - Wetumpka, GA

2219 - 2219 - Marietta #1, GA

2220 - 2220 - Lake Charles, LA

2221 - 2221 - Columbia #2, SC

2222 - 2222 - Columbia #2, SC

2223 - 2223 - Chester #1, SC

2224 - 2224 - Chester #1, SC

2225 - 2225 - Washington, VA

2226 - 2226 - Ashburn, VA

2227 - 2227 - Shawnee, MS

2228 - 2228 - Lake City, TN

2229 - 2229 - Lake City, TN

2230 - 2230 - Lake City, TN

Select as many share groups, location groups or individual locations as you want by double-clicking them to move them from the available list to the selected list. Double-clicking a group or location in the selected list moves it back to the available list.

After clicking *Finish*, the form is displayed.

The form looks exactly like the form that is used at the remote locations, but there are some important differences. The data on the form comes from more than one location. In our example, there are two different locations that have a general ledger account 10 in their table. The *AccountID* is considered the key field for this table since it is used to identify each record.

To see which locations have a particular value for a key field

1. Tab to the key field.
2. Press *Ctrl-Enter* twice on your keyboard.

To see or change the values of a non-key field

Since data comes from multiple locations, it is possible that the data is different in different locations.

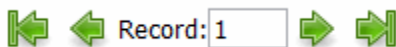
To change the values of a non-key field:

1. Tab to the field.
2. Press *Ctrl-Enter* twice on your keyboard.
3. Type a new value.

Click *Locations* to view which locations use the displayed/selected value. When you change the values, the data changes are sent to every location that has that value for that field.

Moving Between Records

On both forms and spreadsheets, you can easily move to different records by using the VCR controls on the tool bar or by using the keyboard navigation keys.



Moving Between Records on a Form Using the Keyboard

You can move around the form using the following keys. You can also move to a field by clicking the field.

Key	Action
Tab	Moves to the next field
Shift+Tab	Moves to the previous field
F6	Moves to the first control in the next section
Shift F6	Moves to the last control in the previous section
Ctrl+PageUp	Moves to the previous record
Ctrl+PageDown	Moves to the next record
Ctrl+Home	Moves to the first record
Ctrl+End	Moves to the last record
Ctrl+=	Moves to the new record
Ctrl+-	Deletes the current record

Moving Between Records on a Spreadsheet Using the Keyboard

You can move around the spreadsheet using the following keys. You can also move to a cell in the spreadsheet by clicking the cell.

Key	Action
Up	Moves to the previous row
Down	Moves to the next row
Left	Moves to the previous column, if the current field is entirely selected or the cursor is at the left edge of the field
Right	Moves to the next column, if the current field is entirely selected or the cursor is at the right edge of the field
Page up	Moves up one page of rows
Page down	Moves down one page of rows
Ctrl+Page up	Moves to the first row
Ctrl+Page down	Moves to the last row
Tab	Moves to the next field
Shift+Tab	Moves to the previous field
Enter	Moves to the next field
Ctrl+Home	Moves to the first field on the current row
Ctrl+End	Moves to the last field on the current row

Changing the Column Width on Spreadsheets

To change the width of a spreadsheet column to view its contents:

1. Position the mouse at the right-end of the column name button at the top of the column. The pointer will change to a double-headed horizontal arrow.
2. Hold down the left-mouse button and drag the right edge of the column to the desired width.

Adding Records

This page describes how to add new records to the database.

1. Move the cursor to the blank record at the bottom of the table after the last record.
2. Type the data.
3. Press *Tab* on your keyboard to move to the next blank record.

Form: CDMUserRoles x

File Edit Records Help

Record: 4 Search:

	USERID	ROLENAME
	admin	Administrator
	central	Central User
	remote	Remote User
+	Guest	

On both forms and spreadsheets, you can quickly jump to the blank record by clicking the *New Record* tool on the tool bar.

Editing Records

You can edit records on forms and spreadsheets by simply tabbing to the field you want to change and typing in the new data. When you move to a different record or close the form, the changes will automatically be saved in the database.

Record: 5 Search:

	CDMLCID	GLID	GLNum	GLDesc	GLAccountType	GLTypicalBalance	GLPostingType	_EAccountNumber	GL_EAccountDesc
	2101	1	11011101	Bakery	<NULL>	<NULL>	<NULL>	<NULL>	<NULL>
	2101	2	11011102	Paper	<NULL>	<NULL>	<NULL>	<NULL>	<NULL>
	2101	3	11011103	Dairy	<NULL>	<NULL>	<NULL>	<NULL>	<NULL>
	2101	4	11011104	Beef	<NULL>	<NULL>	<NULL>	<NULL>	<NULL>
*	2101	5	1	Primary Vendor	<NULL>	<NULL>	<NULL>	<NULL>	<NULL>
	2103	1	11011101	Bakery	<NULL>	<NULL>	<NULL>	<NULL>	<NULL>
	2103	2	11011102	Paper	<NULL>	<NULL>	<NULL>	<NULL>	<NULL>
	2103	3	11011103	Dairy	<NULL>	<NULL>	<NULL>	<NULL>	<NULL>
	2103	4	11011104	Beef	<NULL>	<NULL>	<NULL>	<NULL>	<NULL>

Editing a Field on a Form

1. Click the field you want to edit.
2. Use the arrow keys on your keyboard to move to the characters you want to change.
3. Press *Delete* to delete characters to the right of the cursor. Backspace to delete characters to the left of the cursor.
4. Type the new characters that you want.

Editing a Field on a Spreadsheet

1. Click the field you want to edit.
2. Type the new characters that you want.

To change only part of the data

1. Press *Home* or *End* on your keyboard to deselect the field.
2. Use the arrow keys on your keyboard to move to the characters that you want to change.
3. Press *Delete* to delete characters to the right of the cursor. Backspace to delete characters to the left of the cursor.
4. Type the new characters that you want.

Deleting Records

To delete the current record on a form or spreadsheet:

1. Click any field in the record you want to delete.
2. Select *Select Record* from the *Edit* menu.
3. Click the *Delete* tool



on the toolbar.

To select multiple records, drag your cursor hold down the far left column while holding down your mouse button.

	CDMLOCID	GLID	GLNum	GLDesc	GLAccountType	GLTypicalBalance	GLPostingType	_EAccountNumbe	GL_EAccountDesc
	2101	1	11011101	Bakery	<NULL>	<NULL>	<NULL>	<NULL>	<NULL>
	2101	2	11011102	Paper	<NULL>	<NULL>	<NULL>	<NULL>	<NULL>
	2101	3	11011103	Dairy	<NULL>	<NULL>	<NULL>	<NULL>	<NULL>
*	2101	4	11011104	Beef	<NULL>	<NULL>	<NULL>	<NULL>	<NULL>
	2101	5	1	Primary Vendor	<NULL>	<NULL>	<NULL>	<NULL>	<NULL>
	2103	1	11011101	Bakery	<NULL>	<NULL>	<NULL>	<NULL>	<NULL>
	2103	2	11011102	Paper	<NULL>	<NULL>	<NULL>	<NULL>	<NULL>
	2103	3	11011103	Dairy	<NULL>	<NULL>	<NULL>	<NULL>	<NULL>

Saving Records

The system automatically saves your changes when you:

- Move to a different record
- Select *Save Record* from the *Record* menu
- Close the form or spreadsheet

If you are editing data on a form or spreadsheet, you can move to the next record by clicking the *Next Record* tool



on the toolbar.

Undoing Changes

If you make a mistake when entering or changing a record, click the *Undo* tool



on the toolbar to undo your changes.

The first time you click *Undo*, the current field is changed back to its original value. If you click *Undo* a second time, all the fields in the record are changed back to their original values.

Finding Records

You can find records on spreadsheets and forms using the *Find* tool



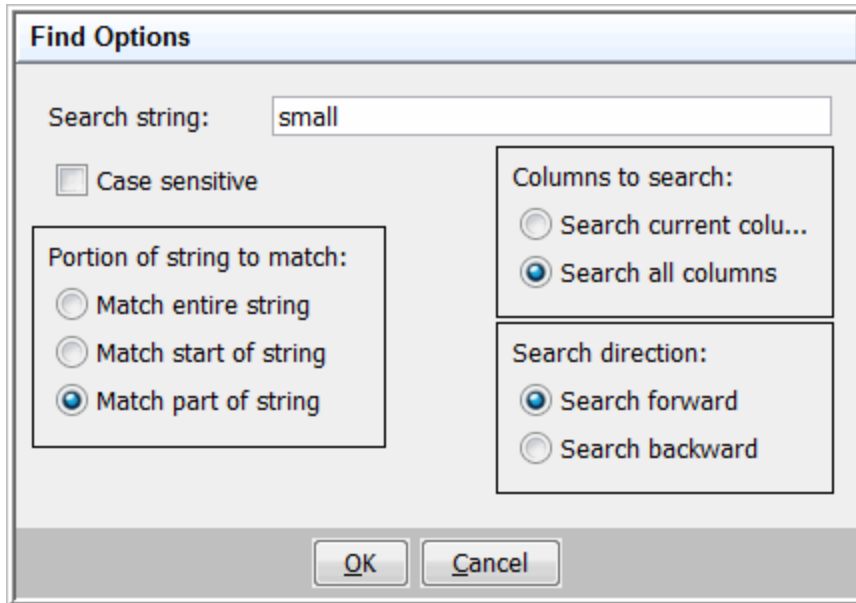
on the toolbar.

1. Press *Tab* (or *Shift+Tab*) on your keyboard until the cursor is in the *Last Name* field.
2. Click the *Find* tool



on the toolbar. The *Find* dialog box opens

3. Type the value you are looking for in the *Search string* field.



Field/Option	Action
Search string	Type the text you want to find. The text can be the entire field value, the first few characters in the value, or any group of characters in the value.
Case sensitive	Check this box if you want the search results to match the same upper and lower-case letters that you typed in the <i>Search string</i> field.
Portion of string to match	Select the type of match for the string that you typed in the <i>Search string</i> field. For example, if the string is "Smith" and <i>Match start of string</i> option is selected, then the search results will only return all records that start with "Smith", e.g. Smith, Smithson, Smithsonian, etc.
Columns to search	This option enables you to search for the string in the current column only, or in all columns in the record.
Search direction	This option tells the system to search either forward or backward through the columns. If the string is not found, you are prompted to search in the opposite direction.

Editing Central Data

When editing data for multiple remote locations at a central location, the data is summarized by grouping records by their key fields. For example, all general ledger account records are grouped by account number. When you open the general ledger account form, you will see a list of all the account numbers from all locations, but you will only see each unique account number once.

Viewing Location Information

When you select an account record to edit, you can press *Ctrl+Enter* on your keyboard to see the location information about each field.

Viewing a Location List for an Account Number

When you press *Ctrl+Enter* on a key field, such as the *Account Number* field, you will see a list of locations at which that account number exists in the database.

Viewing Field Values for Different Locations

When you press *Ctrl+Enter* on a non-key field, such as the description, you will see a list of the values that field has at the various locations. For example, if you press *Ctrl+Enter* on the *Description* field of account number 3300, you might see two different values such as *Accounts Payable* and *Payables*. This tells you that some locations have one description and some locations have another.

To see which locations have a particular value, press *Ctrl+Enter* on the value and a list of locations with that value will be displayed.

Editing Data with Other Applications

Note

The *Generate Test Database* and *Generate Snapshot* functions described here are not available on the standard EDM menu. Your menu and/or forms would need to be customized to include these function, if desired.

Some data cannot be conveniently edited using forms because the editor requires a graphical interface instead of a form interface. This presents a problem at central locations because the databases have been modified to include a location ID field for every record. Therefore, editors from other applications can no longer be used to edit the data.

To solve this problem, the system includes the ability to generate a test database that contains records from a selected location excluding the location ID field. The system registers changes made to the test database by editors from other applications by taking a snapshot of the tables before and after the editing, and then comparing the snapshots to determine what changes were made.

Generating a Test Database

Follow these steps to generate a test database and compare snapshots of the edited tables.

1. Set up a test location with a normal location database.
2. Generate a test database for the selected location using the *Generate Test Database* function.
3. Create a snapshot table list with the tables to include. See [Adding a Snapshot](#) for detailed instructions.
4. Call the *Generate Snapshot* function from an event on your custom form.
5. Edit the data in the tables using editors from other applications.
6. Call the *Generate Snapshot* function to generate a new snapshot of the edited data and compare it to the old data.

Changes made to the snapshot tables will be captured as transactions and transmitted to the other locations that maintain data for that table.

Note

You should not edit tables with both EDM forms and other applications because the snapshot comparison will detect the changes made by the EDM forms and send transactions that have already been sent.

Testing Data Changes Before Transmission

Note

The *Generate Test Database* and *Generate Snapshot* functions described here are not available on the standard EDM menu. Your menu and/or forms would need to be customized to include these function, if desired.

When changing data at a central location, it may be helpful to test the changes with the user's applications.

1. Generate a test database for the selected location using the *Generate Test Database* function.
2. When prompted, select the location for which the test database is being generated.
3. When prompted, select the ODBC dataset name of the destination database. All the records for the requested location are copied to the specified database and the *Location ID* field is removed.
4. Test the location data by running the user's applications on the test database.

Server Text File Editor

Overview

The *Server Text File Editor* is used to create or edit text files on the EDM Server.

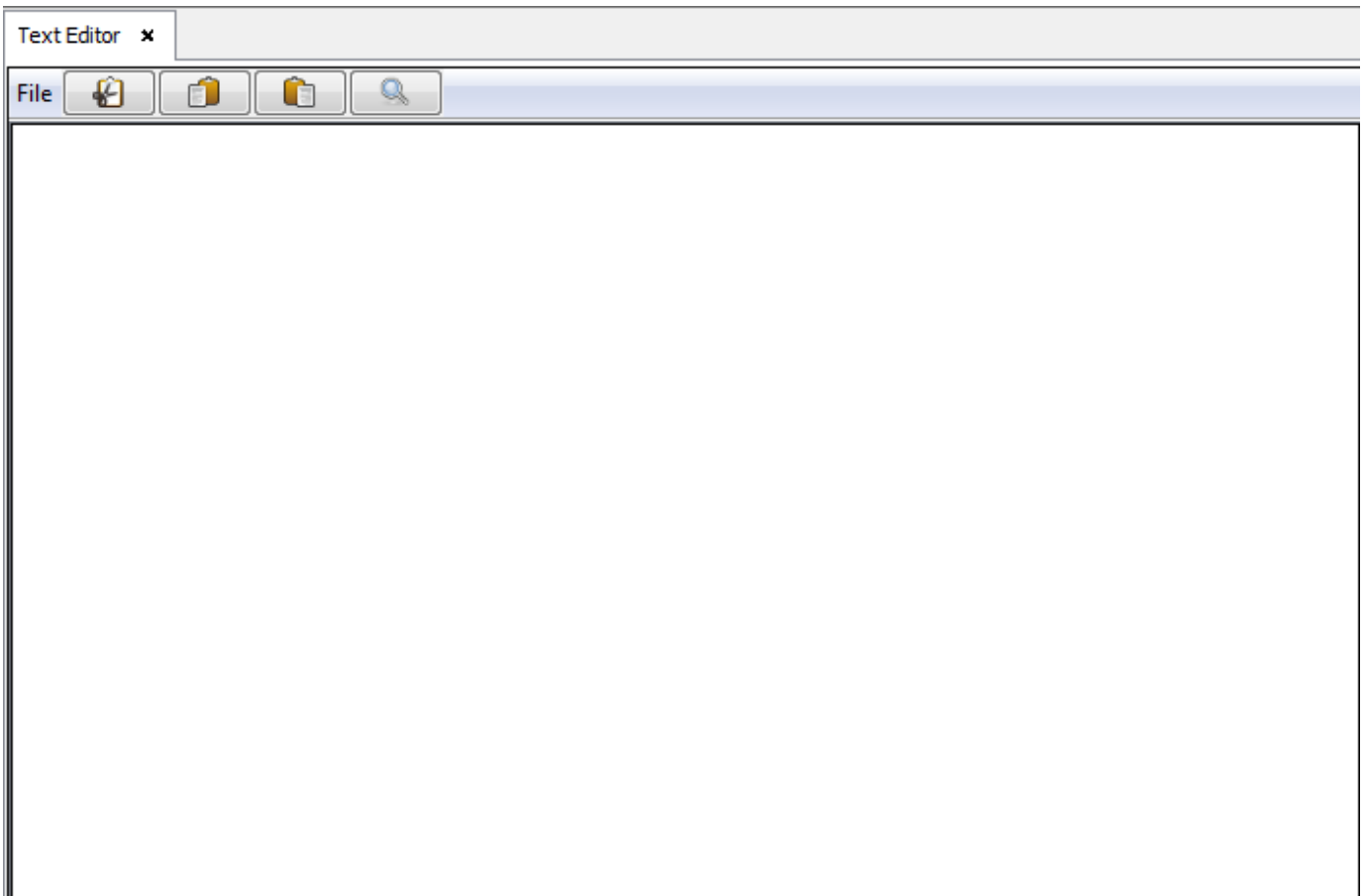
Using the Server Text File Editor

The following sections describe how to use the *Server Text File Editor*.

Opening the Editor

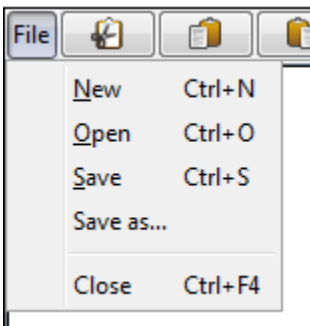
Select *Menus>Edit Server Text File* to open the editor. A new tab is created with an empty edit panel. There is no association with a file name.

If *Edit Server Text File* menu option is not available, you can add it by creating a menu item and setting the action to 'editServerTextFile("")'.

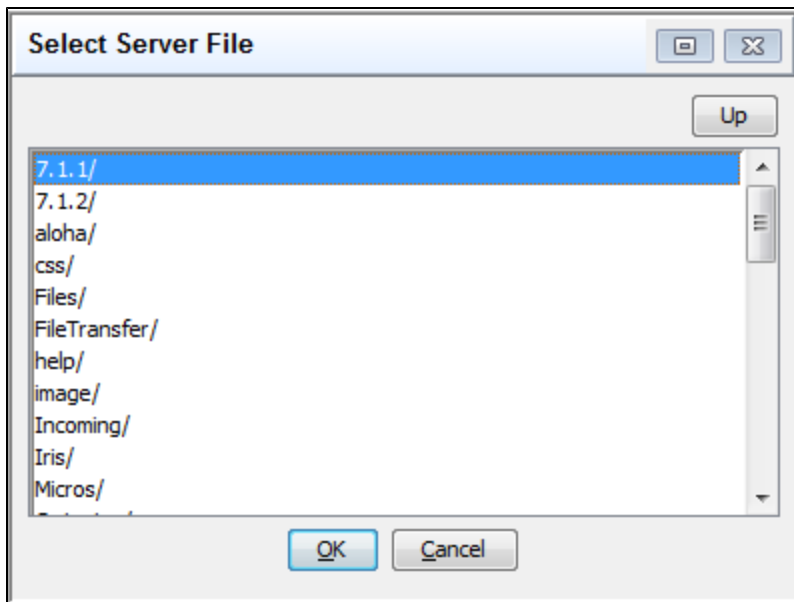


Opening a File

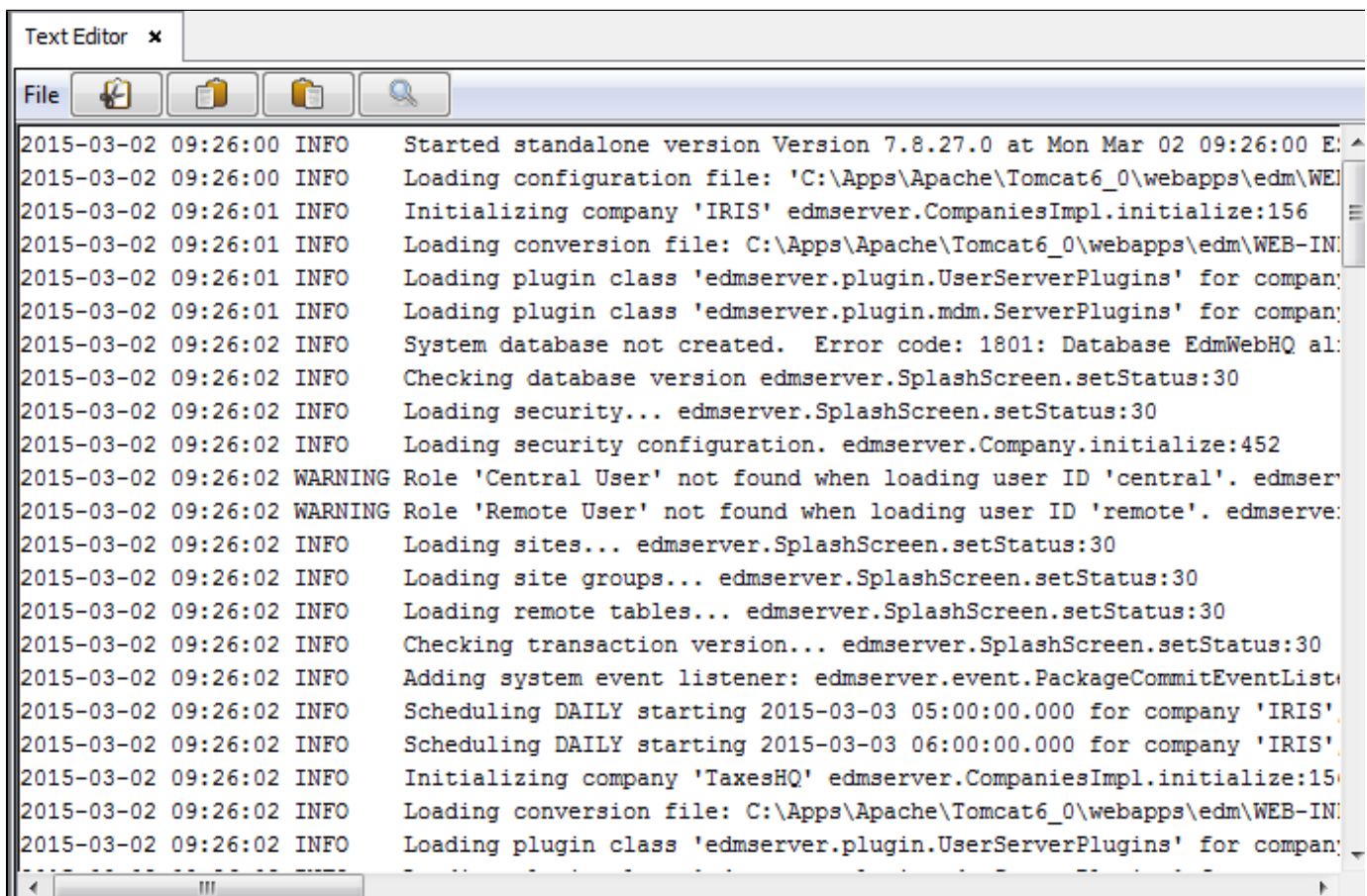
To open a file, select *Open* from the *File* menu.



The *Select Server File* panel opens.



Initially, the contents of the top-level folder of the server are displayed. Double-clicking on a listed folder (or selecting *OK* while that folder is highlighted) will open that folder. The *Up* button will go up one folder level. Double-clicking on a listed file (or selecting *OK* while that file is highlighted) will open that file in the text viewer. In the following example, the *Log.txt* file has been opened.



Editing a File

You can edit the text of a file by typing directly in the panel, or by using the *Cut/Copy/Paste* buttons.

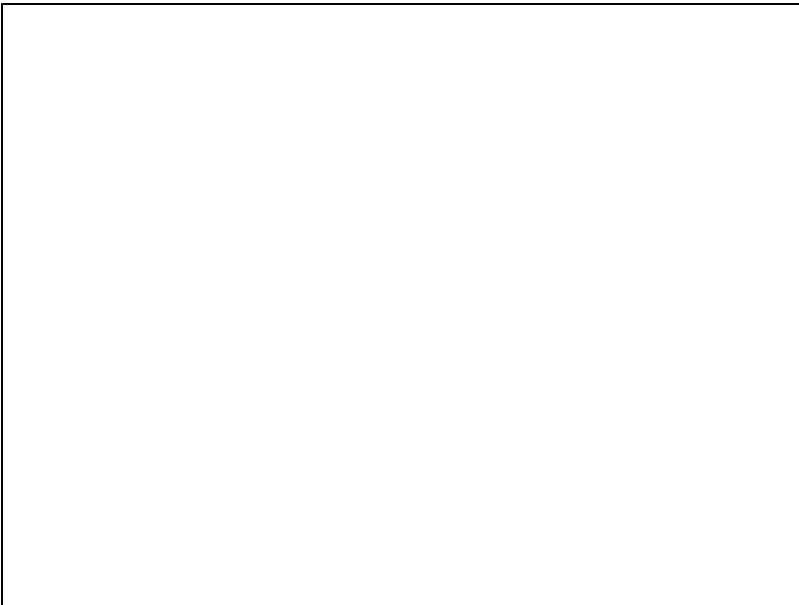


Selecting the *Find* option opens the Find and Replace panel. Text may be found and replaced as required. A text search will be performed from the cursor position to the end of the file, or may optionally wrap around to include the whole document.



Saving a File

Select or from the menu. If you are working with a new file or renaming an existing file, you are prompted to enter the file



Introducing Queries

A query is a request to read or write data in the database. Queries are used to:

- Answer questions about the data like "What are the total sales by month?" or "Which category of products is selling best?"
- Summarize data, e.g. total sales for a state
- Combine data from more than one table into meaningful information, e.g. a list of product sales by category

Why Use Queries?

Queries help you see your data in the way that best suits the task at hand. The following describes the various functions of queries:

Choose fields	Filter the fields in a table to view only the fields that you want to view
Sort records	Sort your data in either ascending or descending order on any number of fields, e.g. list all departments in alphabetical order
Calculate totals	Calculate totals and perform other mathematical calculations on your data
Change data	Add, change and delete records in tables (or even create new tables), e.g. use an Update query to give all employees in the 'Sales' department a 10% raise
Combine data from multiple tables	For example, combine the category name, product name and total sales for a product on one spreadsheet
Use as the source for forms	For example, show the category name of each product you edit or a graph that combines data from several different tables

See Also

[Creating Queries](#)
[Query Columns](#)
[Sorting Results](#)
[Viewing and Editing Query Results](#)
[Multi-Table Queries](#)
[Editing the SQL Statement for a Query](#)
[Query Parameters](#)
[Calculating Totals](#)
[Action Queries](#)
[Changing a Query Design](#)
[Copying Queries](#)
[Getting Query Parameters from a Form](#)

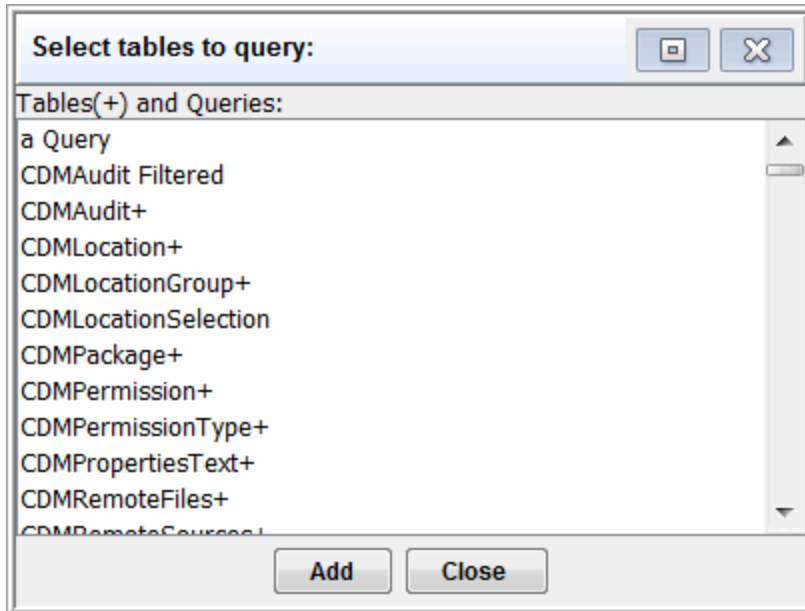
Creating Queries

Selecting the Data Sources to Use in the Query

The first step in creating a query is to add data sources. Data sources can be tables or other queries. A query can have as many data sources as necessary. Data sources can be connected together with joins between the columns in the data sources.

For example, to create a query that shows customer orders, you would add the *Customers* and *Orders* tables to the query as data sources, and then join them using the *Customer ID* column from each table.

1. Select the *Setup* module.
2. Expand the *Application* menu.
3. Right-click *Queries*, and select *New* from the shortcut menu that appears. The *Select tables to query* dialog opens, which lists the tables and queries that you can use as data sources. Tables are marked with a + sign.



4. Select the tables and queries to use as data sources.
5. Click *Add*. A window will be added to the query designer showing the data source name in the title bar and the fields in the table or query in the main part of the window.
6. Click *Close*.

Query Designer

After you have selected the tables, the *Query* designer opens where you define:

- The data sources
- The columns
- The condition to use for selecting records

The *Query* designer has two sections:

- The top section is the diagram section where the data sources are displayed
- The bottom section is the column section where the columns you have defined for the query are displayed

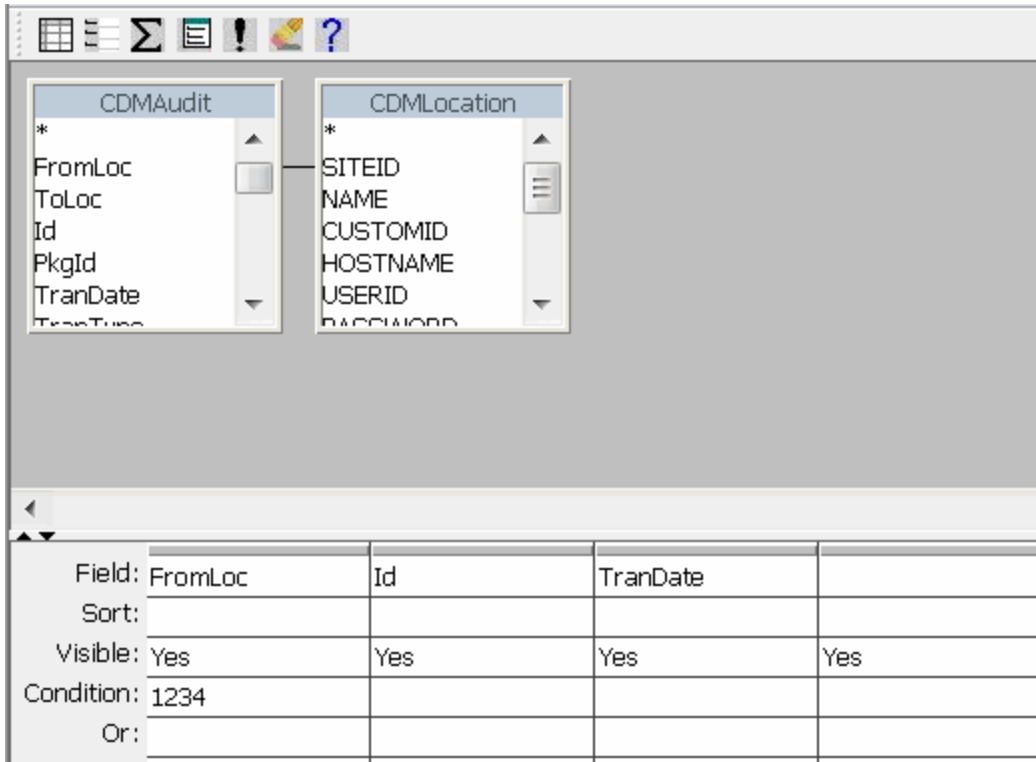


Diagram Section

The diagram section contains small windows for each data source. These source windows can be moved by dragging their title bar, and sized by dragging their sizing border. To delete a source, click anywhere in the source window and press *Delete*.

Column Section

The column section is like a spreadsheet except only the condition rows can be scrolled up and down. All the rows above the condition rows can only be scrolled left and right.

You can move around the column section using the following keys.

Key	Action
Up	Moves to the previous row
Down	Moves to the next row
Left	Moves to the previous column if the current column is entirely selected or the cursor is at the left edge of the column
Right	Moves to the next column if the current column is entirely selected or the cursor is at the right edge of the column
Page Up	Moves up one page of rows
Page Down	Moves down one page of rows
Tab	Moves to the next column
Shift-Tab	Moves to the previous column
Enter	Moves to the next column
Ctrl+Home	Moves to the first column on the current row
Ctrl+End	Moves to the last column on the current row
Alt+Down	Displays or hides the list if cursor is on a combo box

Query Columns

Query columns are like columns in a table. They contain one piece of data that can come from a table column, a calculation, or input from the user. When you create a query, you set up the columns that you want in the result set. This allows you to see just the data you want.

See Also

[Adding a Column from a Data Source](#)

[Renaming Columns](#)

[Creating Calculated Columns](#)

[Adding Multiple Fields as a Group](#)

[Changing Field Values](#)

[Rearranging Columns](#)

[Changing Column Widths](#)

[Inserting Columns](#)

[Deleting Columns](#)

[Viewing Table Names](#)

[Hiding Columns](#)

[Specifying Column Conditions](#)

Adding a Column from a Data Source

Query columns are like columns in a table. They contain one piece of data that can come from a table column, a calculation or input from the user. When you create a query, you set up the columns that you want in the result set.

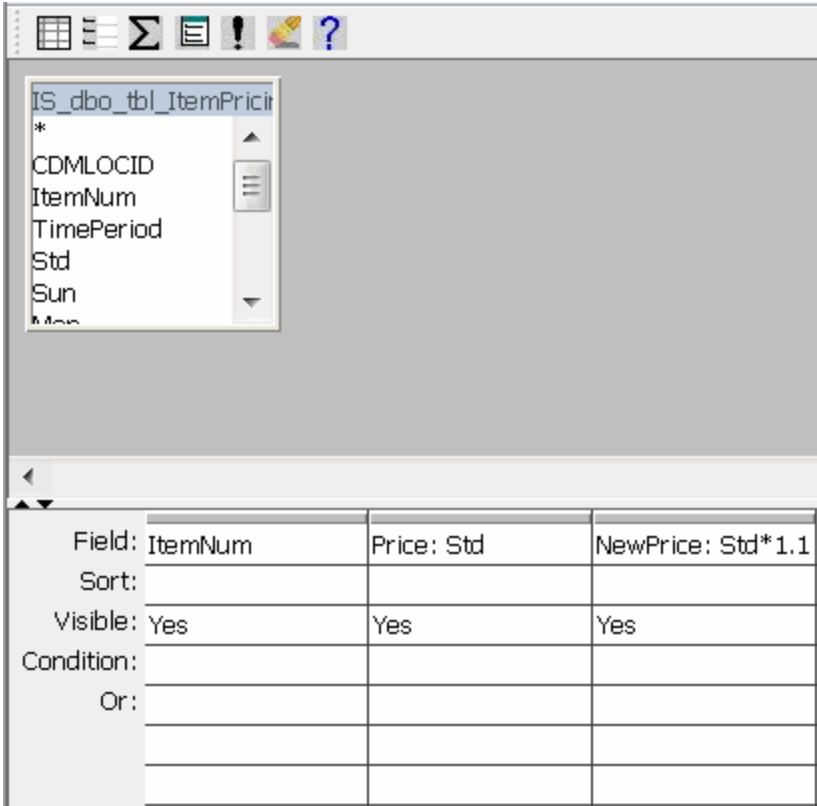
The column window of the query designer is in the form of a spreadsheet where each column represents one column in the query and each cell in the column represents an attribute of the column.

How to Add a Column

The following are different methods for adding a column:

- Double-click the column name in the data source window. The column is automatically added as the last column in the query.
- Select one or more columns in the data source window, and then drag them to the desired position in the column window.
- Type the field names or expressions directly into the *Field* row. The system will automatically find the related data source. If the column name exists in more than one data source, the system will insert the name of the first data source and a period before the column name. If the column name does not exist in any data source, the system will prompt you for the value of the column when you run the query.

The field expression is used to get the field value. This can be a field name from one of the data sources or a full expression like *Std * 1.1*.



If the field expression is a simple field name, then that name is used. If the column is an expression and you do not supply a name, the system will calculate a unique name for the field like *Expr1* or *Expr2*.

See [Expressions](#) for more detailed information about expressions.

Renaming Columns

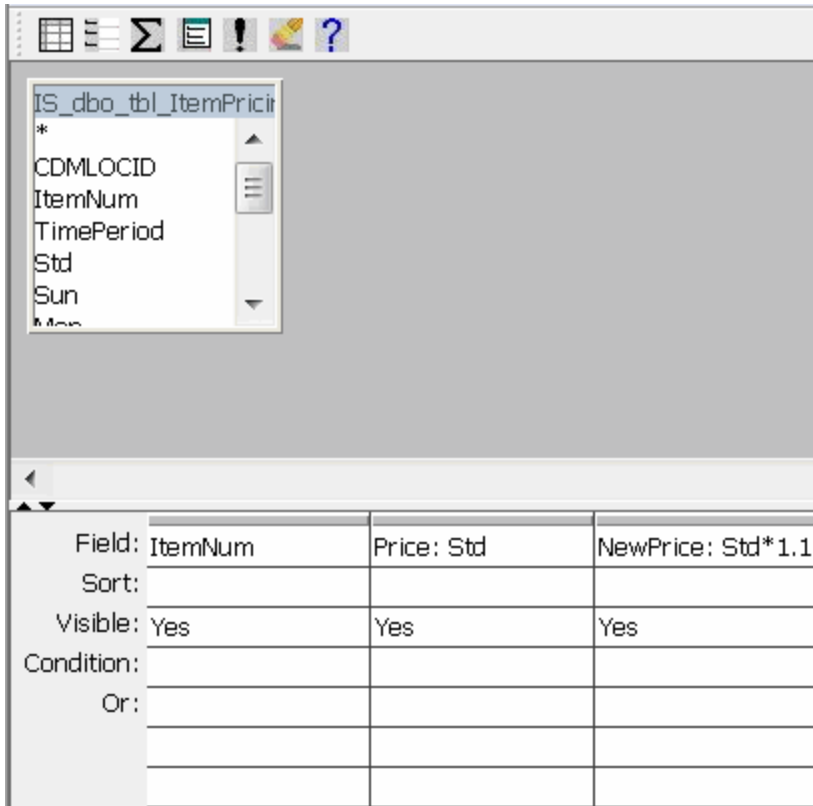
If you want to supply a name for a field, whether it is a simple column or an expression, enter the name at the beginning of the field expression followed by a colon and the column expression.

For example: *NewPrice: Std*1.1*

This creates a column named *NewPrice* with the value *Std*1.1*.

You can also rename a field from a table to make the name more useful.

For example, you could rename the *Std* field to *Price* by entering *Price: Std*



Creating Calculated Columns

To create a calculated field in a query: Enter the name of the field followed by a colon and the expression that you want to calculate.

For example, *NewPrice: Std*1.1* creates a calculated field called *NewPrice* that displays the *Std* field multiplied by 1.1

Adding Multiple Fields as a Group

Adding All Fields from a Data Source

Double-click the asterisk * field.

Adding Selected Fields from a Data Source

1. Click the first field in the list of fields to add.
2. Hold down the *Shift* key on your keyboard, and then select the last field in the list. All the fields between the first and the last field are now selected.
3. Drag the selected fields to the field window.

Changing Field Values

To change a field value in the query design:

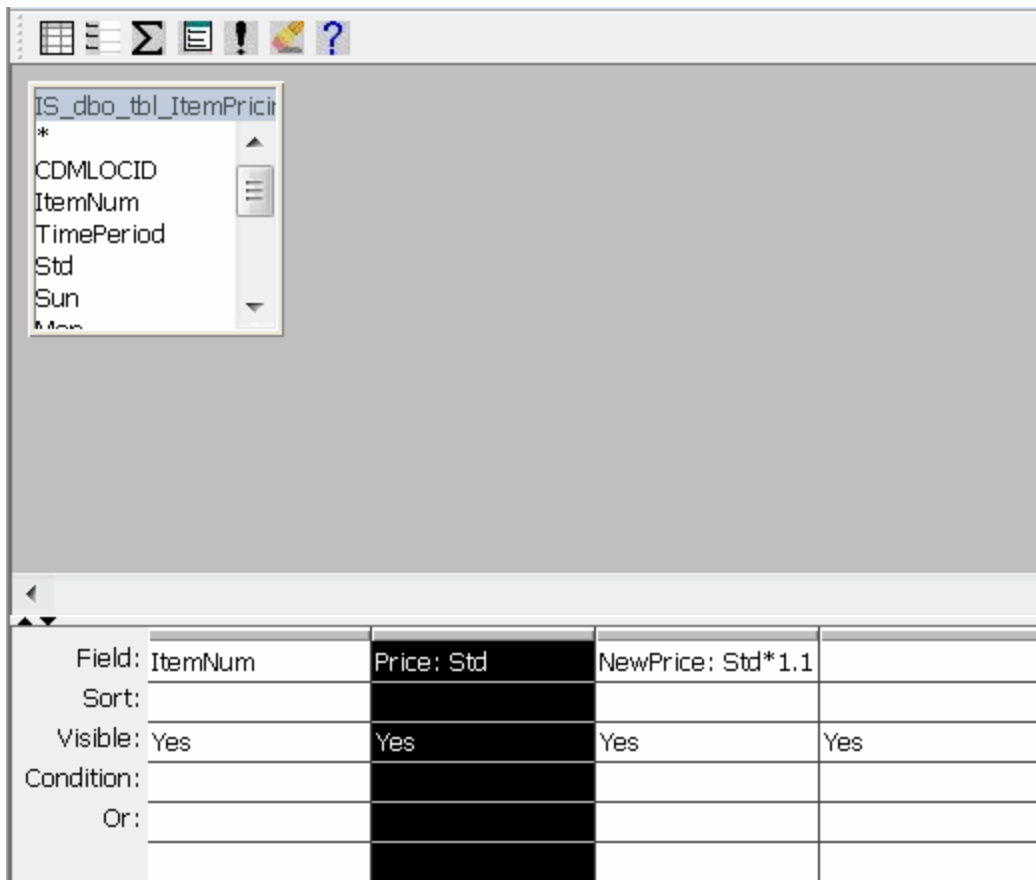
1. Use the arrow keys (or the *Tab/Shift-Tab* keys) on your keyboard to move to the field you want to change.
2. Type the new value.

Rearranging Columns

Query columns can be moved to a different position in the field window either one at a time or in groups. When the query results are displayed, the columns will be in the order that you want.

Moving a Single Column

1. Click the gray selector button at the top of the column in the field window.
2. Drag the selected column to the desired position.



Moving Multiple Column

1. Select the first column you want to move.
2. While holding down the left-mouse button, drag your mouse pointer to the last column you want to move.
3. Release the left mouse button. All the columns between the first and the last column are now selected.
4. Hold down the left-mouse button over the button at the top of any of the selected columns.
5. Drag the columns to the desired position.

Changing Column Widths

The following describes how to change the width of a column on a spreadsheet.

1. Hover your mouse pointer over the right edge of the selector button at the top of the column. The mouse pointer changes to a double-headed arrow.
2. Drag the right edge of the column to the desired width.

Inserting Columns

To insert a column from a data source between two existing columns on a spreadsheet: Drag the column (using the right mouse button) from the list of columns in the data source and drop it on an existing column.

The new column is inserted to the left of the column that you dropped it on.

Deleting Columns

Deleting a Single Column

1. Click the gray selector button at the top of the field to select the column.
2. Press *Delete*.

Deleting Multiple Columns

1. Press and hold the left mouse button on the first column to delete.
2. Drag the mouse pointer to the last column to delete.

3. Release the left mouse button. All the columns between the first and the last column are now selected.
4. Press *Delete*.

Viewing Table Names

Each field that comes from a query data source has a table name associated with it. The table name is displayed on the *Table* row of the field window. This row is not shown when you first open the query designer.

To view table names: Select *Table Names* from the *View* menu.

Hiding Columns

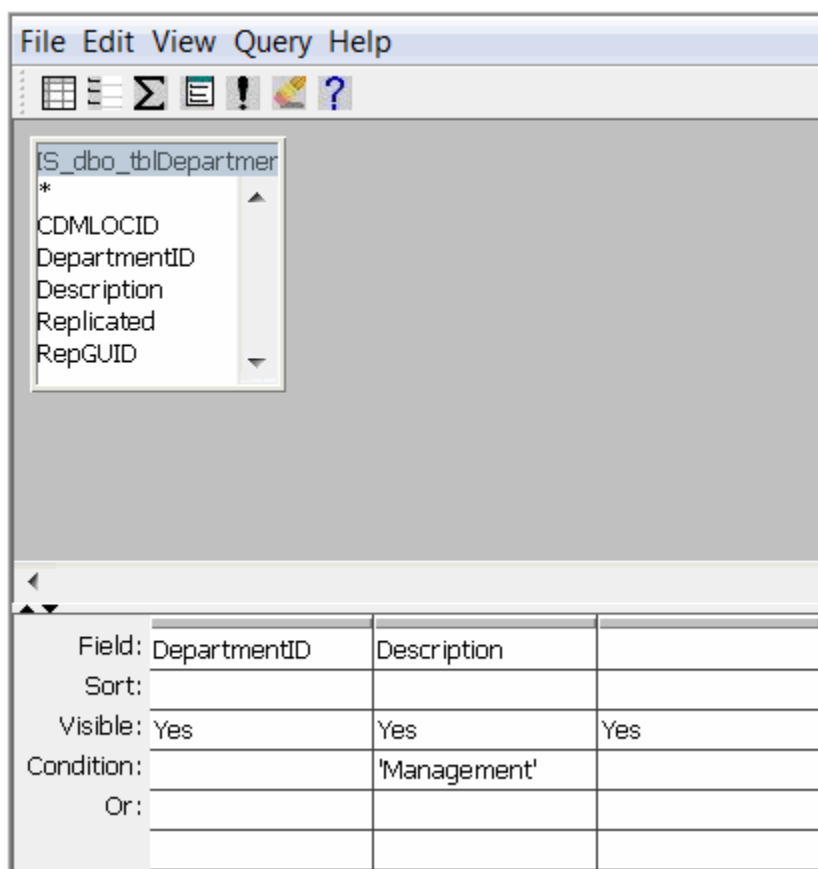
Sometimes you want to set a condition on a column or sort a column, but you do not want the column to show up in the result set. To set the visibility of a column, select *Yes* from the *Visible* row of the column.

Specifying Column Conditions

Setting Conditions on Columns

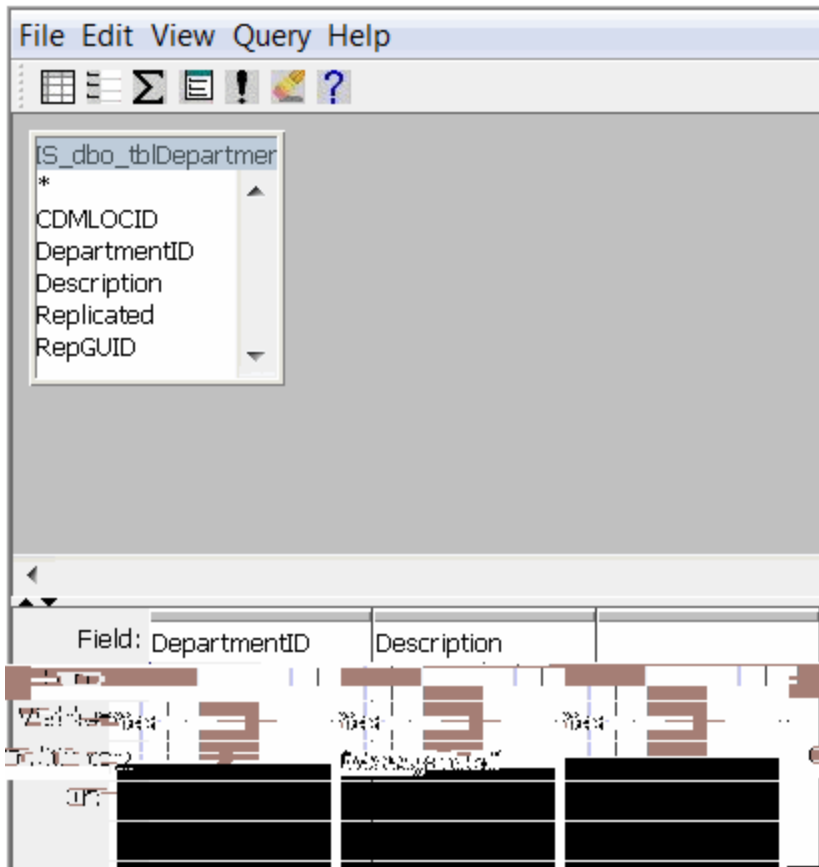
1. Navigate to the *Condition* row in the column section.
2. Type the condition expression.

For example, if you want to query all departments named 'Management', add the *Description* column from the *Departments* table to the query. Navigate to the *Condition* row, and then type 'Management'. See [Expressions](#) for more information on condition expressions.



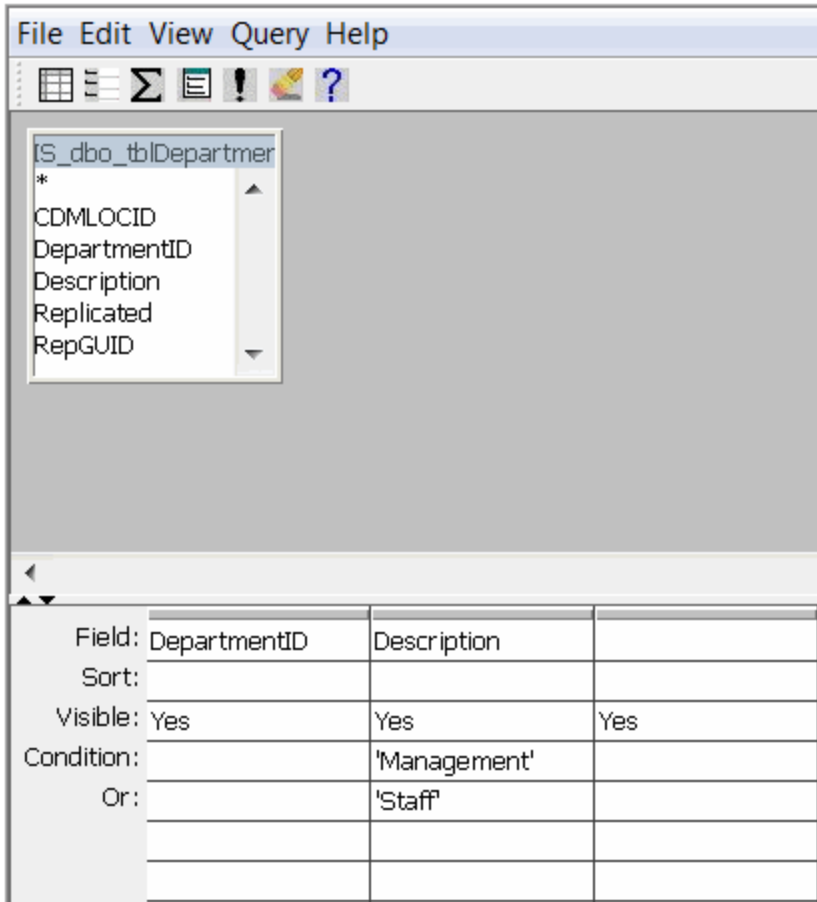
If you add condition expressions for several columns on the same *Condition* row, then the conditions must all be true for the row in order to be included in the result set.

For example, if you put 'Management' on the first *Condition* row of the *Description* column, and 2 on the first *Condition* row of the *DepartmentID* column, then the only rows that have *DepartmentID*=2 and *Description*='Management' will be included in the result set.



If you add condition expressions for more than one row, then if any single row of conditions is true, that row will be included in the query result set. If all rows of conditions are false, then the row will not be included.

For example, if the first row of conditions checks for *Description*='Management', and the second row of conditions checks for *Description*='Staff', then any rows where the *Description*='Management' OR 'Staff' will be included.



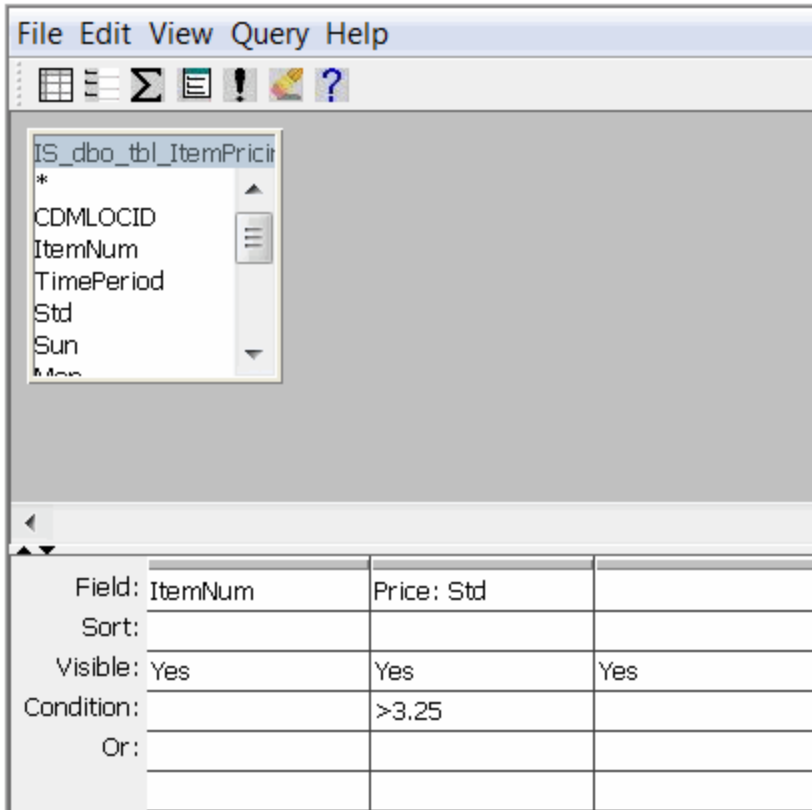
See Also

- [Condition Expressions](#)
- [Using OR Conditions](#)
- [Selecting Rows by Date](#)

Condition Expressions

Condition expressions are similar to normal expressions except that they do not include the first operand. Instead, the column value is assumed to be the first operand in the expression.

For example, the condition expression `>3.25` on the *Std* column translates to the expression `Std>3.25`.



If the condition expression is checking for equality, you may omit the = operator.

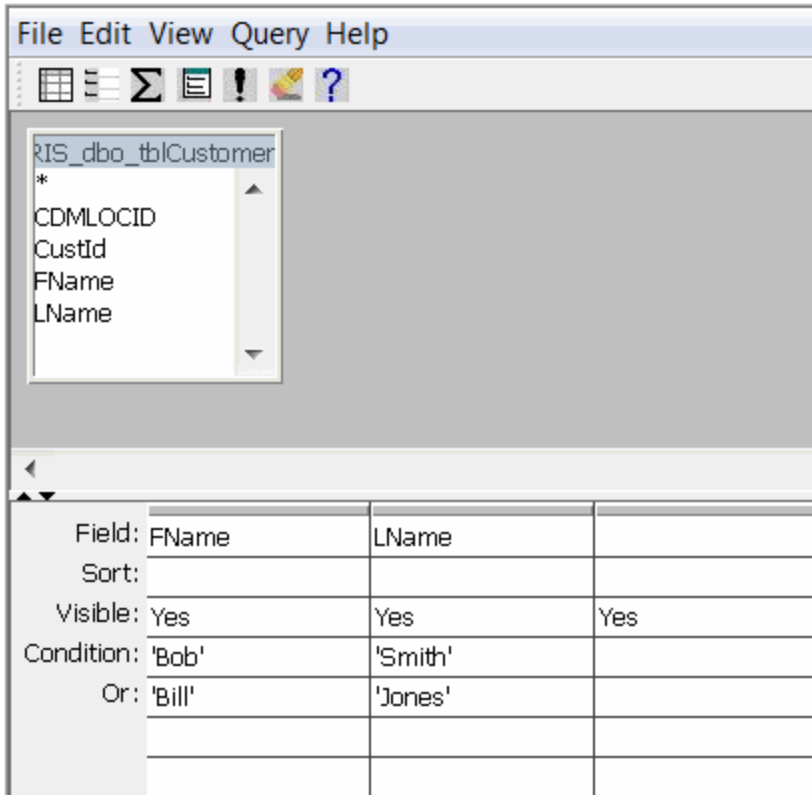
For example, the condition expression "Smith" OR "Jones" on the *LASTNAME* column translates to the expression *LASTNAME*="Smith" OR *LASTNAME*="Jones". For each potential record in the query result set, the condition expressions are checked and if they are true, then the row is included in the result set. Otherwise, the row is not included.

Condition Expression Examples

Condition expression	Selects records where
"Smith"	Value is 'Smith'
="Smith"	Value is 'Smith'
>1000	Value is > 1000
< CharToDate("2/5/95")	Date value is before 2/5/95
Between 300 and 500	Value is between 300 and 500 (inclusive)
<Forms:Employee:ID	Value is less than the <i>ID</i> field on the <i>Employee</i> form
> 10 and <= 20	Value is greater than 10 and less than or equal to 20
LIKE "Sm%"	Values that begin with "Sm" like Smith and Smart
LIKE "A_C"	Value with A_any letter_then C e.g. ABC or ARC

Using OR Conditions

When entering conditions, you may have several conditions for which you want to select records. For example, to list all customers with the name 'Bob Smith' or 'Bill Jones', enter each set of conditions on a separate row.



To read this type of query, say "and" between each condition on the same row and "or" between rows.

For example, you would read this query as selecting customers whose first name is 'Bob' and last name is 'Smith', or whose first name is 'Bill' and last name is 'Jones'. This query will only select customers whose name is 'Bob Smith' or 'Bill Jones'.

Selecting Rows by Date

Current Date

To select records with the current date:

Type `Date("")` on the *Condition* row of the date column to check.

The `Date` function parameter is the format string that you want the date to use. An empty format string `Date("")` returns the current date in the default format.

Specific Date

To select records for a specific date:

Type `CharToDate('m/d/y', "")` where `m/d/y` is the desired date, e.g. `CharToDate('5/17/13', "")` on the *Condition* row of the date column to check.

The `CharToDate` function converts a date string to a date value that can be compared to the values in a date field. The `CharToDate` function has two parameters:

- The date string to convert
- The format string that describes the format of the date string; this string can be "" if the date string is in the default format

Date Range

To select records where the date value is more than 30 days old:

Type `<Date("")-30` on the *Condition* row of the date column to check.

Sorting Results

Applying Sorting

1. Move the cursor to the *Sort* row of the fields to use for sorting.
2. Click the combo box button.
3. Select *Ascending* or *Descending*.

The columns of the query will be sorted in the order they appear in the column window, so if you want to sort on *Last Name* before *First Name*, make sure that *Last Name* appears in an earlier column than *First Name*.

Removing Sorting

1. Move the cursor to the *Sort* row of the column.
2. Click the combo box button.
3. Select *Not sorted*.

Viewing and Editing Query Results

Viewing Query Results

Click



to run the query and view the results.

File Edit Records Help			
Record: <input type="text" value="1"/> Search: <input type="text"/>			
	ItemNum	Std	New Price
*	<input type="text" value="0"/>		0 0
	1000	4.99	5.489
	1002	6.99	7.689
	1004	9.99	10.989
	1062	0 0	
	1064	0 0	
	1066	1.29	1.419
	1068	1.29	1.419
	1072	1.29	1.419
	1074	1.29	1.419

Editing Query Results

To open a query result set for editing in a spreadsheet from the application window:

1. Select the Queries page.
2. Double-click the query to edit.

If there is more than one table in the query then only the fields from the "many" side of a one-to-many join will be editable. Also, Totals queries and Action queries are not editable.

For more information on using a spreadsheet to edit data, see [Editing Records](#).

Multi-Table Queries

To show related information, you can add more than one table or query to your query design. For example, you could design a query to show the relation between product category names and product names even though the product category names and the product names are in different tables.

See Also

- Joining Tables
- Types of Joins Between Data Sources in a Query
- Deleting a Join
- Saving Joins

Joining Tables

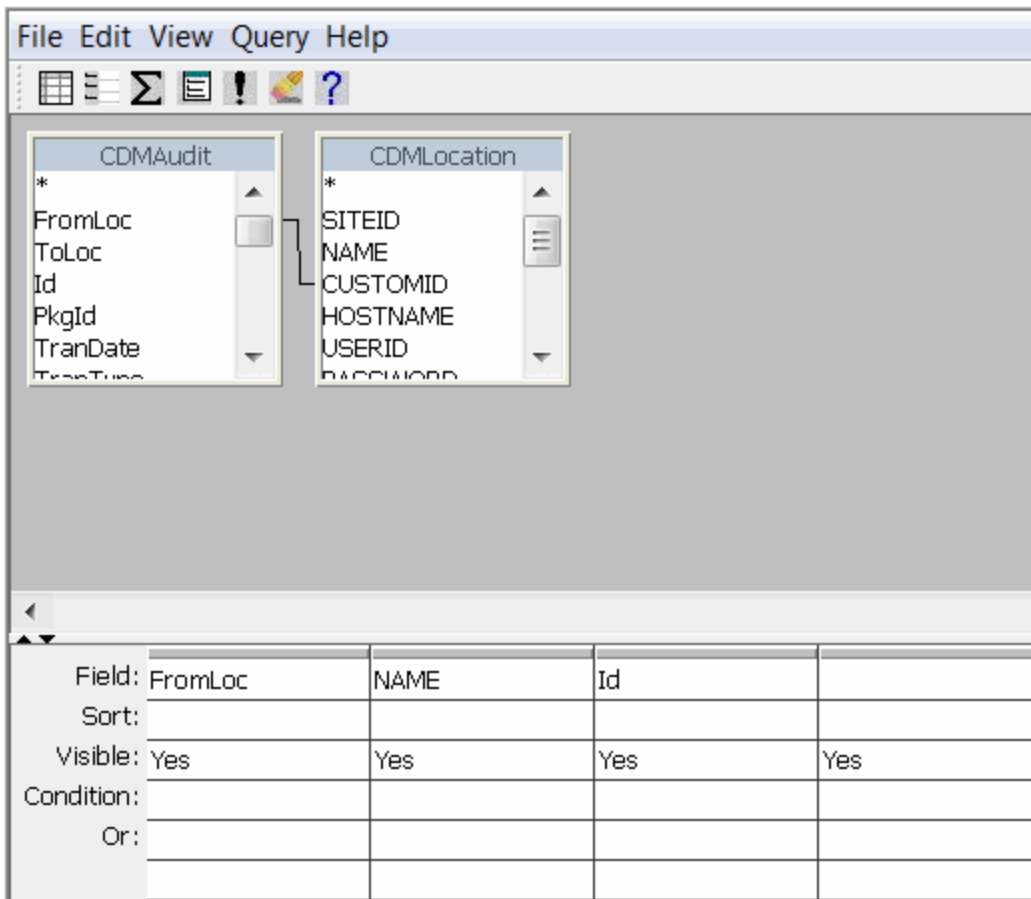
You can join common fields from different tables in a query to combine information. In the following example, we will join fields in order to display a transaction ID along with the ID and name of the location associated with the transaction.

1. Select the *Setup* module.
2. Expand the *Application* menu.
3. Right-click *Queries*, and select *New* from the shortcut menu that appears. The *Select tables to query* window opens.
4. Double-click the *CDMAudit* table to add it to the query.
5. Double-click the *CDMLocation* table to add it to the query.
6. Close the *Select tables to query* window.
7. Press and hold down the right mouse button on the *FromLoc* column in the *CDMAudit* table.
8. Move the mouse pointer to the *CDMLOCID* column in the *CDMLocation* table and release the mouse button.

To determine which location name goes with each transaction, the query goes through the transaction records in the *CDMAudit* table and references the *FromLoc* column.

If more than one field must match between the tables

You can drag other columns from one table to another to create joins.



By creating a join between the two tables, you can include fields from both tables in the query. For example, you might want to see the transaction ID, location ID and name in the results. Since the *NAME* column only exists in the *CDMLocation* table, and the *FromLoc* and *ID* columns only exist in the *CDMAudit* table, you have to join the tables to get the information you need.

Types of Joins Between Data Sources in a Query

Join type	Description
-----------	-------------

Union	No fields from either data source are joined together; all records in the first table are matched with all records in the second table, e.g. if there are 10 records in the first table and 20 records in the second table, there will be 200 records in the result set.
One-to-One	A join exists between all the fields of a unique key in the first table to all the fields of a unique key in the second table. There will be either 0 or 1 matching record in the first table for every record in the second table.
One-to-Many	The fields on one side of a join include all the fields of a unique key and the fields on the other side of the join do not. There may be many records found to match each record from the table on the unique side of the join.
Many-to-Many	Neither side of a join includes all the fields of a unique key

When you join multiple tables, it can render some of the columns Read-Only. In the example provided [here](#), the *NAME* column is Read-Only since the same customer record can be joined to many different order records, so the record appears on many different rows in the query results. When this situation occurs, the columns are rendered Read-Only to prevent the user from editing a record on one row that would affect many other rows.

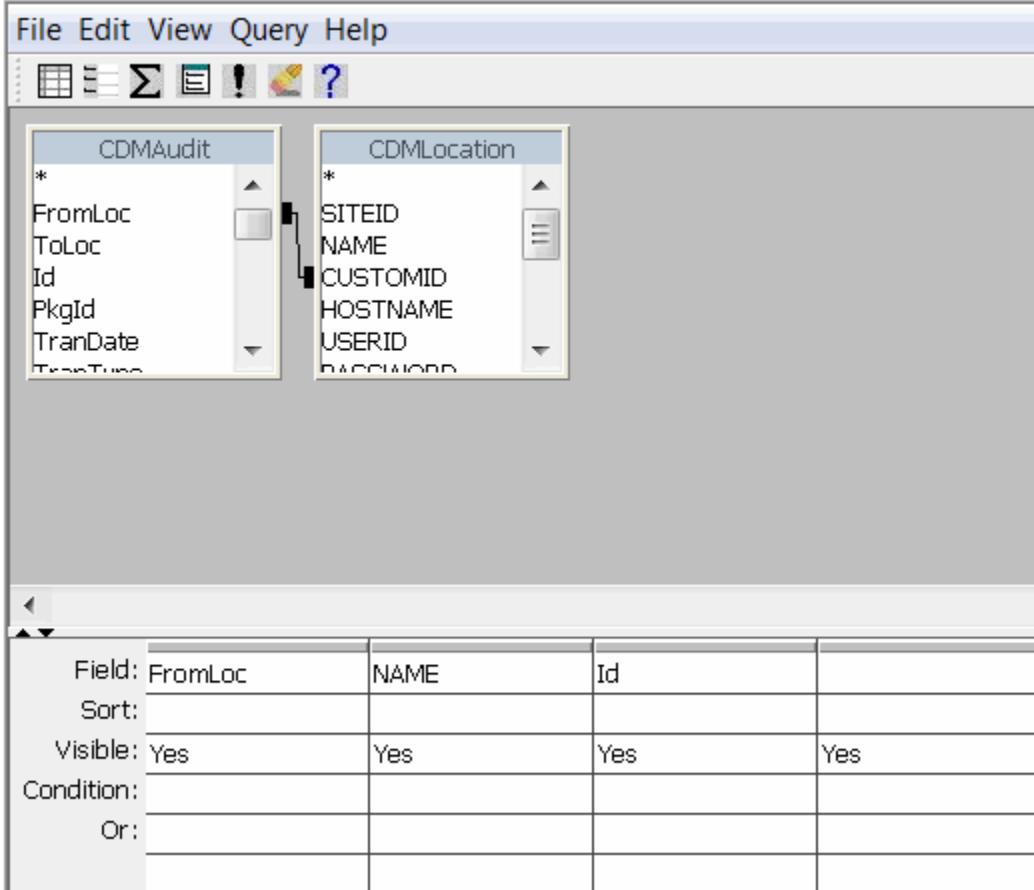
Columns in underlying tables of a query are editable when:

- The query does not have the *Unique Values* flag set
- It is not a Total query and one of the following conditions is true:
 - The table is the only table in the query -OR-
 - All tables are joined with One-to-One joins (joined by unique keys) -OR-
 - The table is on the many side of One-to-Many joins (unique key on one side) -OR-
 - The table is on the one side of a One-to-Many join and no columns from the many side are used

Deleting a Join

The following describes how to delete a Join between two fields in a query.

1. Click the line that is displayed between the two joined fields in the Query design window. When the join is selected, a select block appears at each end.
2. Press the *Delete* key on your keyboard.



Saving Joins

When a join between tables is saved, it is called a *Table Relation*. Saved join lines are displayed in the Query designer as blue lines. Joins that are NOT saved are displayed as black lines.

For example, you can save the join between the *Invoice* table and the *Item* table using the *InvoiceNumber* field in each table. Whenever tables with a saved join are added to a query, the tables are automatically joined on the columns in the saved join.

Table joins can only be saved when the destination table has a primary key and all the fields in the primary key are included in the join.

For example, you can save the join between the *Customer ID* field in the *Orders* table and the *ID* field in the *Customers* table since the destination field (*ID*) is the primary key. You CANNOT save the join between the *ID* field and the *Customer ID* field because *Customer ID* is not the primary key in the *Orders* table.

To Save a Join

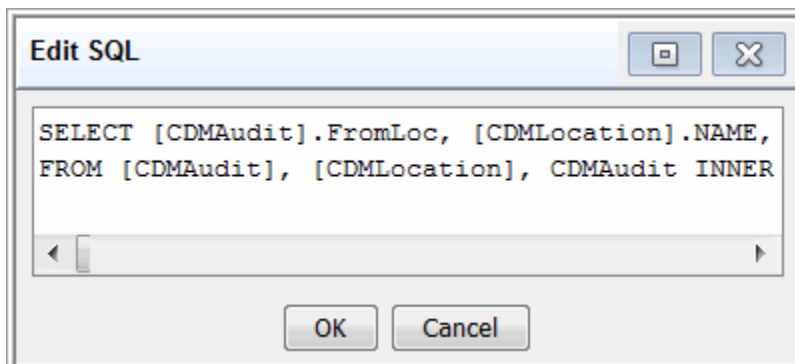
1. Select the *Setup* module.
2. Expand the *Application* menu.
3. Right-click *Queries*, and select *New* from the shortcut menu that appears. The *Select tables to query* window opens.
4. Add the desired tables.
5. Drag the fields from the source table to the primary key fields of the destination table.
6. Select one of the join lines.
7. Select *Save Relation* from the *Edit* menu.

To Delete a Saved Join

1. Select the *Setup* module.
2. Expand the *Application* menu.
3. Right-click *Queries*, and select *New* from the shortcut menu that appears. The *Select tables to query* window opens.
4. Add the tables with the saved Join.
5. Select one of the join lines.
6. Select *Delete Relation* from the *Edit* menu.

Editing the SQL Statement for a Query

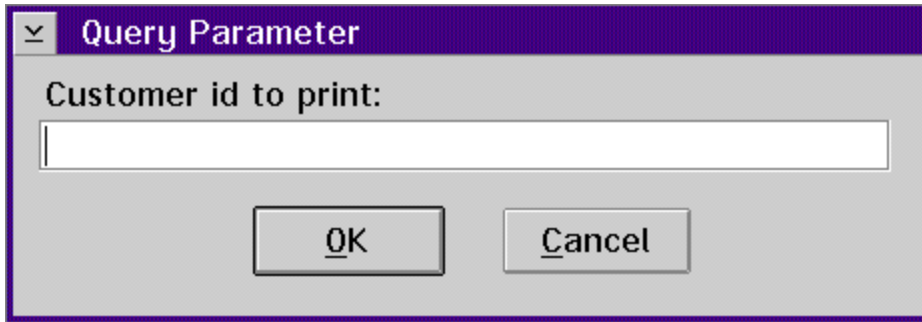
1. Select *View SQL* from the *View* menu of the Query designer. A dialog box opens displaying the SQL statement.



2. Make the necessary changes to the SQL statement.
3. Click *OK*. If the SQL statement is valid, the diagram and column window are updated accordingly.

Query Parameters

You can setup a query that prompts the user to provide parameters. You can include as many parameters in a query as you need. For example, to setup a work order form where the user enters a Customer ID, use a condition expression like `= [Customer id to print]` on the *Condition* row of the *Customer ID* column of the query. The text inside the square brackets appears on the form when the query is run.



After the user enters a value and clicks *OK*, the query continues and the appropriate rows are selected.

Note

Ensure that the parameter prompt text does not match a column name.

Calculating Totals

Total Query

A *Total Query* combines all the records of its result set using the functions: *SUM*, *AVG*, *MINIMUM*, *MAXIMUM* and *COUNT*

For example, to add up all the costs of inventory items based on their recipe ID, you would create a query that calculated the sum of the *StdCost* column. The results of the query would be one row for each recipe ID with a *StdCost* column containing the sum of all the costs for that recipe.

To change a query to a *Total Query*, select *Totals* from the *View* menu -OR- click



from the toolbar.

Total Functions

Total function	Description	Example
Group By	Totals all rows with the same value into a single row. When you group by a column, the result set will have as many rows as there are unique values in the columns.	If you <i>Group By</i> the <i>Department</i> column and <i>Sum</i> the <i>Salary</i> column, the query result will contain one row for each department with the <i>Salary</i> column containing the sum of all the salaries of employees in that department.
Where	Limits the records included in the query by setting conditions on the column	If you set the <i>Total</i> function of the <i>ID</i> column to <i>Where</i> and the condition to <1000, then only records with ID's less than 1000 are included.
Sum	Places the sum of the column values in all the included rows in the column	If you query the <i>Salary</i> column in the <i>Employee</i> table using the <i>Sum</i> function, then the result will contain one row with the <i>Salary</i> column containing the sum of all the salaries of all the employees.
Avg	Places the average of the column values in all the included rows in the column	If you query the <i>Salary</i> column in the <i>Employee</i> table using the <i>Avg</i> function, then the result will contain one row with the <i>Salary</i> column containing the average of all the salaries of all the employees.
Minimum	Places the smallest value of all the included rows in the column	If you query the <i>Salary</i> column in the <i>Employee</i> table using the <i>Minimum</i> function, then the result will contain one row with the <i>Salary</i> column containing the smallest salary of all the employees.
Maximum	Places the largest value of all the included rows in the column	If you query the <i>Salary</i> column in the <i>Employee</i> table using the <i>Maximum</i> function, then the result will contain one row with the <i>Salary</i> column containing the largest salary of all the employees.
Count	Places the total number of included rows in the column	If you query the <i>Salary</i> column in the <i>Employee</i> table using the <i>Count</i> function, then the result will contain one row with the <i>Salary</i> column containing the number of employees.

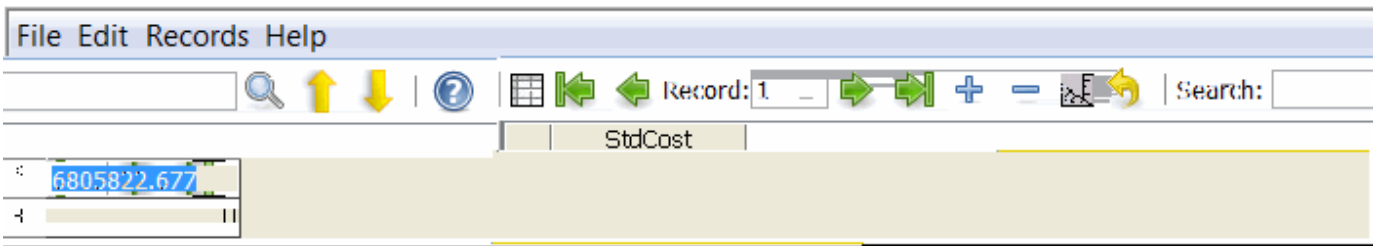
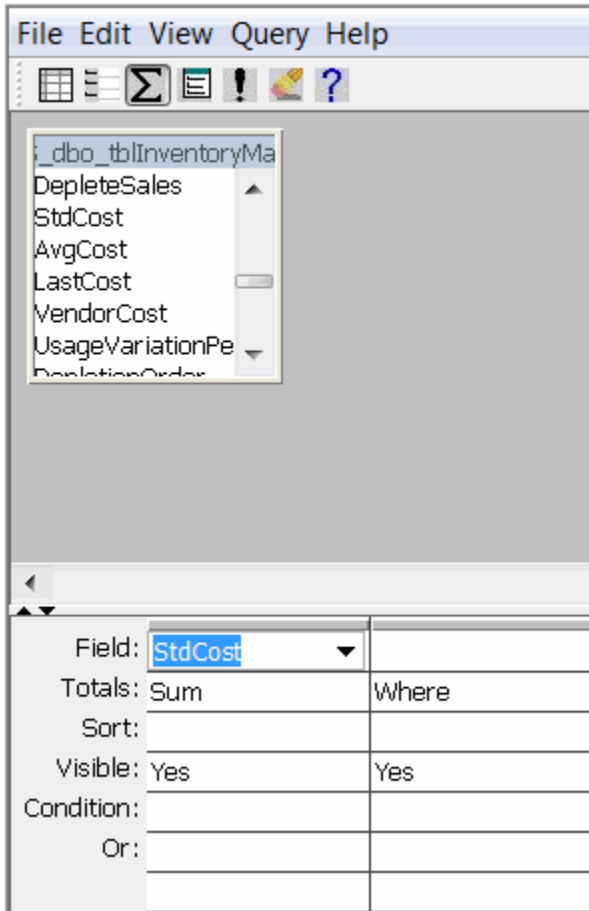
See Also

[Calculating Totals for All Records](#)
[Calculating Totals for Groups of Records](#)
[Specifying Conditions for Groups](#)
[Specifying Conditions for Totals](#)
[Specifying Conditions Before Calculating Totals](#)

Calculating Totals for All Records

If you want to calculate a single set of totals for all the records selected by the query, then you should make sure that none of the fields are set to *Group By*.


For example, if you want to see the total cost for all inventory items, then set the total function of the *StdCost* field from the *Inventory* table to *Sum*



Calculating Totals for All Records

1. Create a query and drag the desired fields to the field window.
2. Click

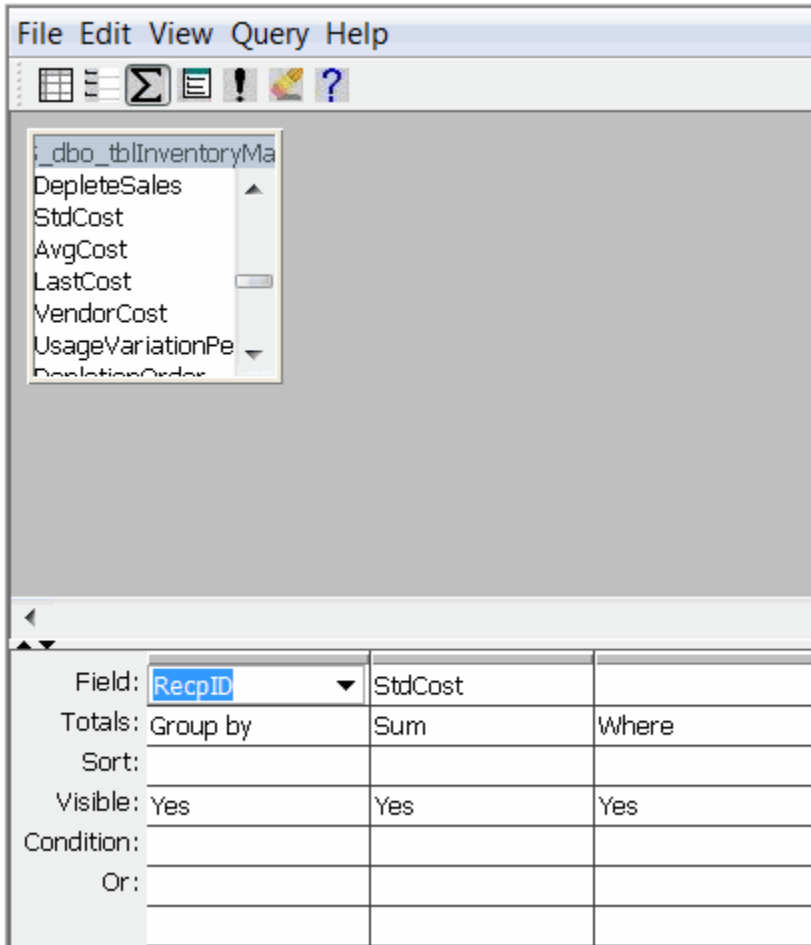


- from the toolbar.
- On the *Total* row, select a total function for each field e.g. *Sum*, *Avg*, *Minimum*. Since you want totals for all records, none of the fields should use the total function *Group By*.
 - To specify conditions for certain fields to limit which records are totaled, select the total function *Where* and fill in the *Condition* row for those fields.
 - Click  from the toolbar.

Calculating Totals for Groups of Records

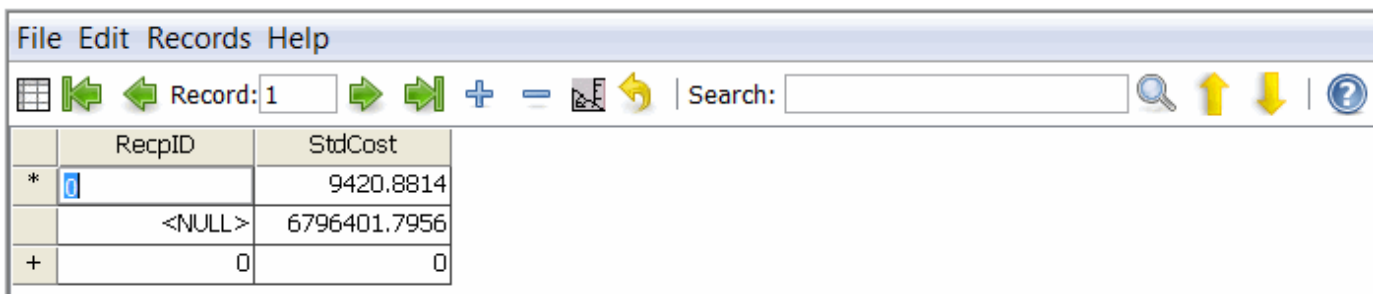
To calculate a set of totals for groups of records, use the *Group By* total function to mark the applicable fields.

For example, to display the total cost for all inventory items with the same recipe ID, set the *Group By* total function on the *RecpID* field. The result contains a row for each recipe ID containing the total cost for all inventory items that use that recipe.



The screenshot shows a software window with a menu bar (File, Edit, View, Query, Help) and a toolbar. A field list is open, showing fields from the table `_dbo_tblInventoryMa`: DepleteSales, StdCost, AvgCost, LastCost, VendorCost, UsageVariationPe, and ReplicationOrder. Below the field list is a table for configuring totals:

Field:	RecpID	StdCost	
Totals:	Group by	Sum	Where
Sort:			
Visible:	Yes	Yes	Yes
Condition:			
Or:			





The screenshot shows a software window with a menu bar (File, Edit, Records, Help) and a toolbar. A records table is displayed with the following data:

	RecpID	StdCost
*	0	9420.8814
	<NULL>	6796401.7956
+	0	0

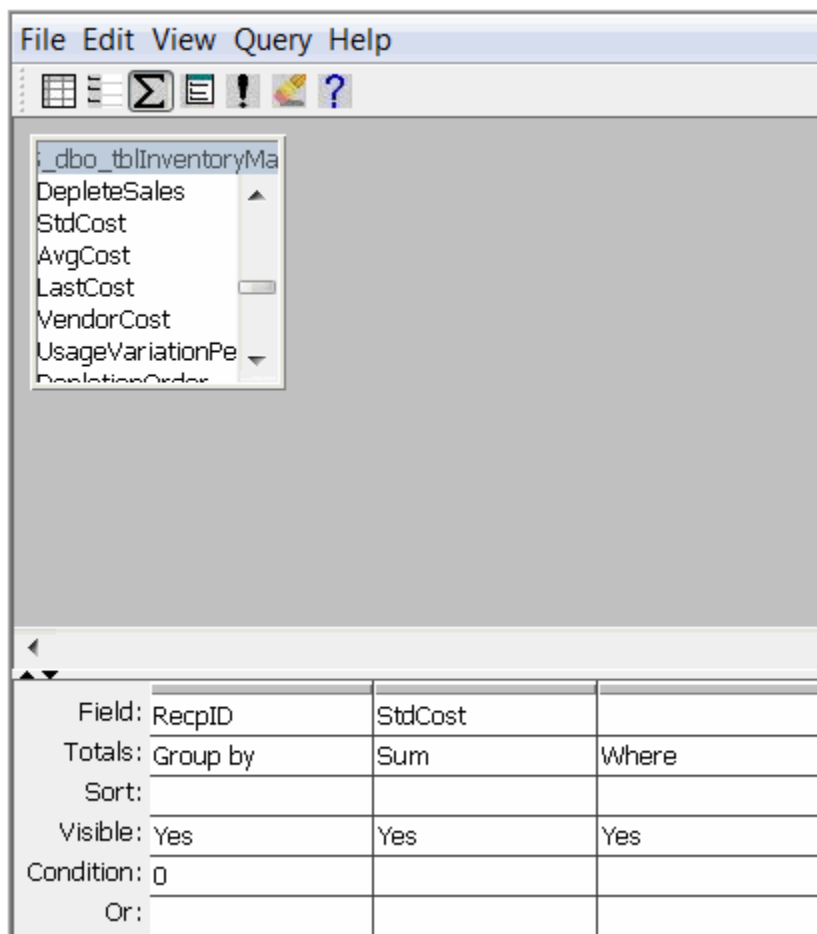
Calculating Totals for Groups of Records

- Create a query and drag the desired fields to the field window.

2. Click  from the toolbar.
3. From the *Totals* row, apply the *Group By* total function to the fields you want to group by.
4. Apply the desired total function to the remaining fields.
5. To specify conditions for fields that are totaled, select the total function *Where* and fill in the *Condition* row.
6. Click  from the toolbar.

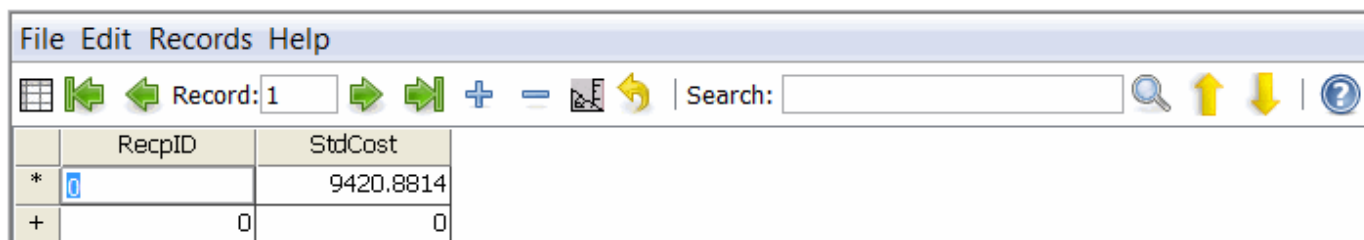
Specifying Conditions for Groups

You can specify conditions on the *Condition* row of a *Group By* field to limit the groups that are selected. For example, to show the total costs for all inventory items with recipe ID 0, enter 0 in the *Condition* row of the *RecpID* field.



The screenshot shows a software window with a menu bar (File, Edit, View, Query, Help) and a toolbar. A list of fields is displayed, including *RecpID*, *StdCost*, *AvgCost*, *LastCost*, *VendorCost*, *UsageVariationPe*, and *RelationOrder*. Below the list is a table for specifying conditions for groups.

Field:	RecpID	StdCost	
Totals:	Group by	Sum	Where
Sort:			
Visible:	Yes	Yes	Yes
Condition:	0		
Or:			



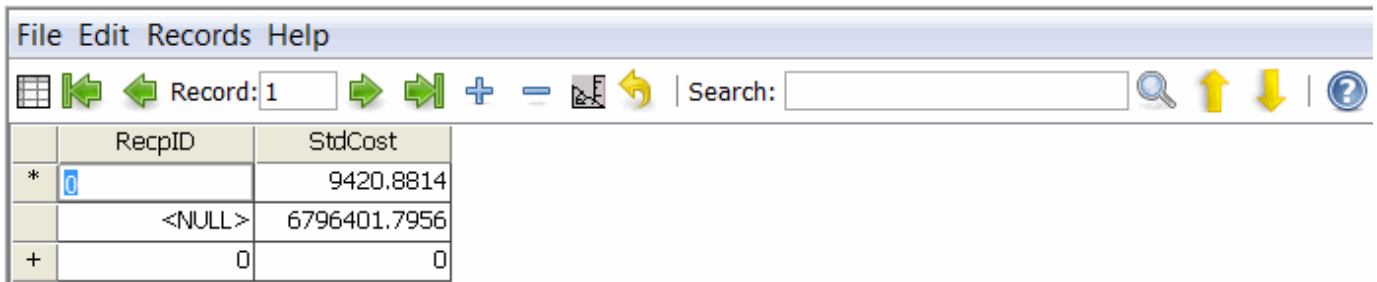
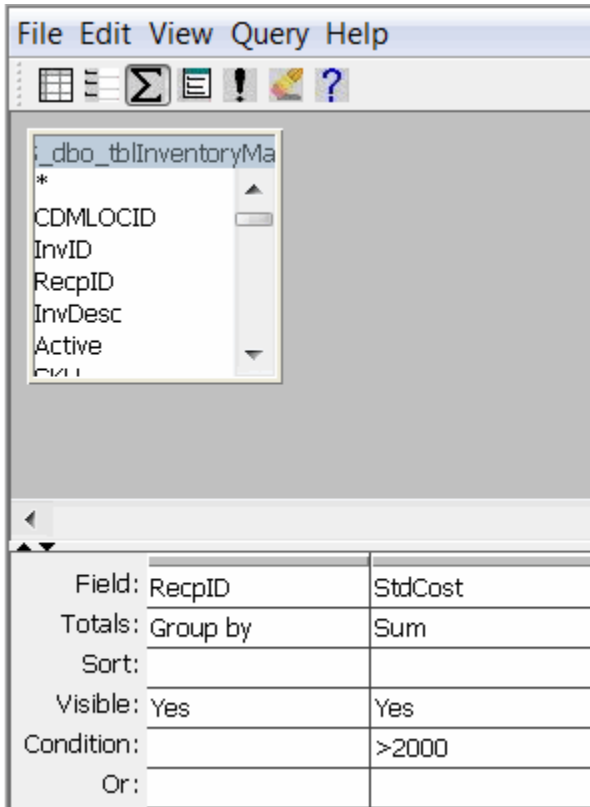
The screenshot shows a software window with a menu bar (File, Edit, Records, Help) and a toolbar. A search bar is visible with the text "Record: 1". Below the search bar is a table of data.

	RecpID	StdCost
*	0	9420.8814
+	0	0

Specifying Conditions for Totals

You can specify conditions on the *Condition* row of a *Sum*, *Avg*, *Minimum*, *Maximum*, or *Count* field to limit which totals are included. For example, to show the total costs for all inventory items with the same recipe with a total greater than \$2,000, enter >2000 on the *Condition* row.

w of the *StdCost* field.



Specifying Conditions Before Calculating Totals

You can specify conditions on fields to limit the records before the totals are calculated by using the *Where* total function.

For example, to show the total costs for each recipe, but only include costs greater than \$2,000:

1. Add the *StdCost* field to the query again.
2. Set the total function to *Where*.
3. Set the *Visible* attribute to *No*.
4. Set the condition to *>2000*.

Action Queries

Action queries are queries that change data in the database by inserting/updating/deleting records or by creating a new table.

See Also

[Make Table Query](#)
[Insert Query](#)
[Update Query](#)

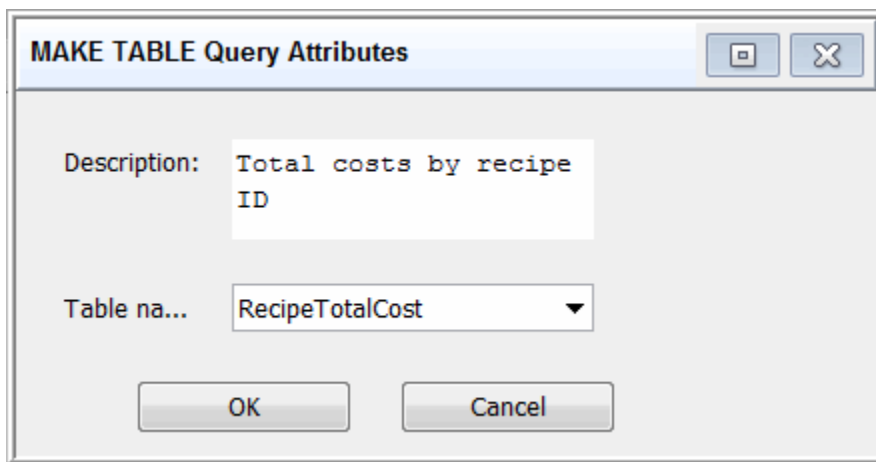
Deleting Rows

Make Table Query

A *Make Table* query creates a table with selected records with the same columns as the result set. *Make Table* queries can be used to take snapshots of data for use in other queries or for partial backups.

Changing a Query to a Make Table query

1. Select the *Setup* module.
2. Expand the *Application* menu.
3. Right-click *Queries*, and select *New* from the shortcut menu that appears. The *Select tables to query* dialog opens, which lists the tables and queries that you can use as data sources.
4. Select the tables and queries to use as data sources.
5. Select the desired columns from the selected data sources to make the table.
6. Select *Make Table* from the *Query* menu. The *MAKE TABLE query attributes* window opens.
7. Type a description for the table to create in the *Description* field.
8. Type a name for the table to create in the *Table name* field.
9. Click *OK*.



Running a Make Table Query

When you run a *Make Table* query, you are prompted for the number of rows to add to the new table. The rows of the result set are added to the new table.

To disable the warning message when running action queries in expressions: Call *ShowWarnings(false)* before running the query to turn warnings off, and *ShowWarnings(true)* after running the query to turn warnings back on.

For example: `=ShowWarnings(false) + Open('Query','UpdateInventory') + ShowWarnings(true)`

Make Table queries cannot be used as the source of a form or as a data source for another query.

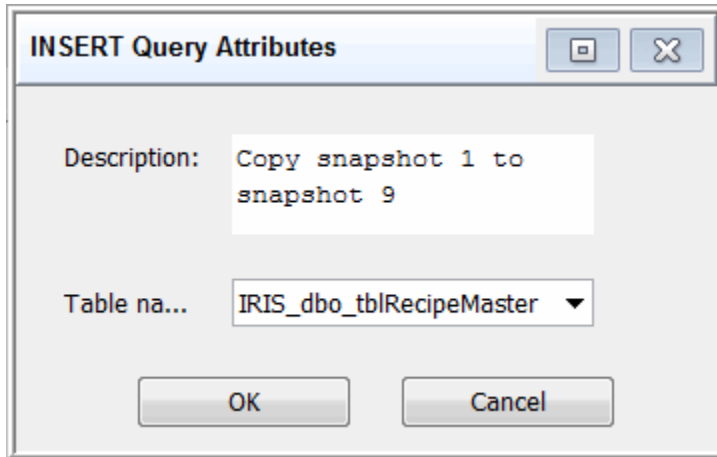
Insert Query

An *Insert* query creates a result set like a *Select* query and inserts those rows in an existing table.

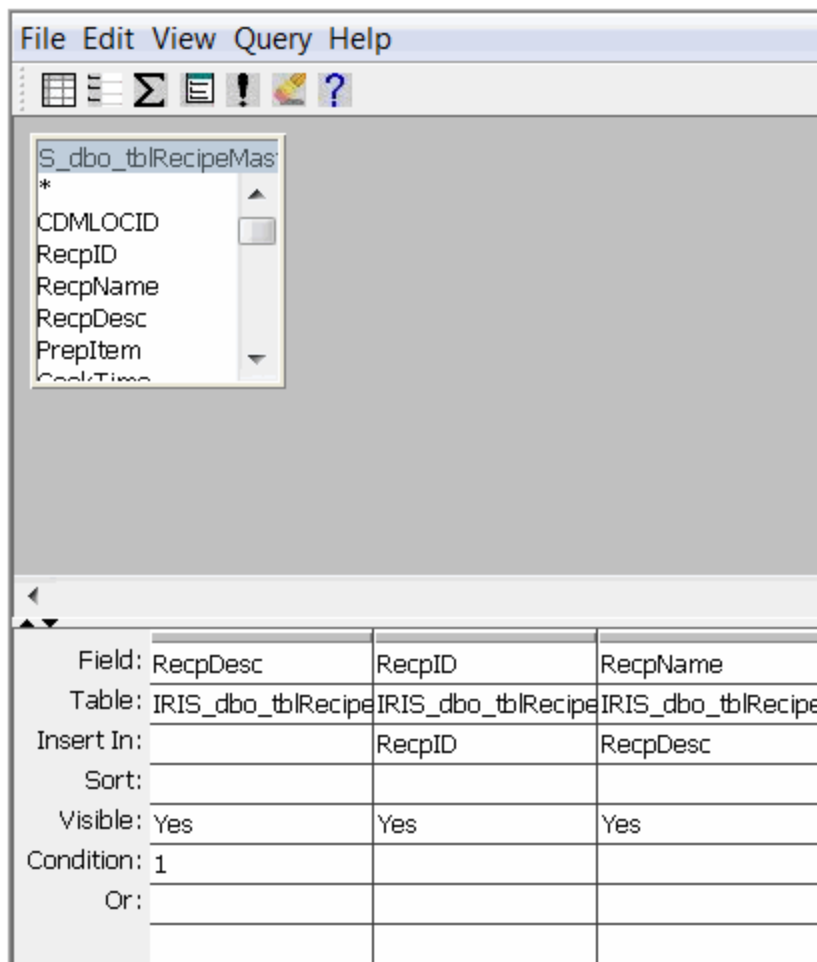
Setting up an Insert Query

1. Select the *Setup* module.
2. Expand the *Application* menu.
3. Right-click *Queries*, and select *New* from the shortcut menu that appears. The *Select tables to query* dialog opens, which lists the tables and queries that you can use as data sources.
4. Select the tables and queries to use as data sources.
5. Select the desired columns from the selected data sources to insert in the table.
6. Select *Insert* from the *Query* menu. The *INSERT Query attributes* window opens.
7. Type a description of the query in the *Description* field.
8. Select the table where you want the rows inserted from the *Table name* drop-down.

9. Click *OK*.



10. For each column of the query, select from the *Insert In* row the column in the destination table where you want to insert the query column data. The columns where you do not insert data will be NULL after the rows are inserted.



Running an Insert Query

When you run an *Insert* query, you are prompted to confirm the number of rows to add to the destination table. Upon confirmation, the rows of the result set are added to the destination table.

To disable the warning message when running action queries in expressions: Call *ShowWarnings(false)* before running the query to turn warnings off, and *ShowWarnings(true)* after running the query to turn warnings back on.

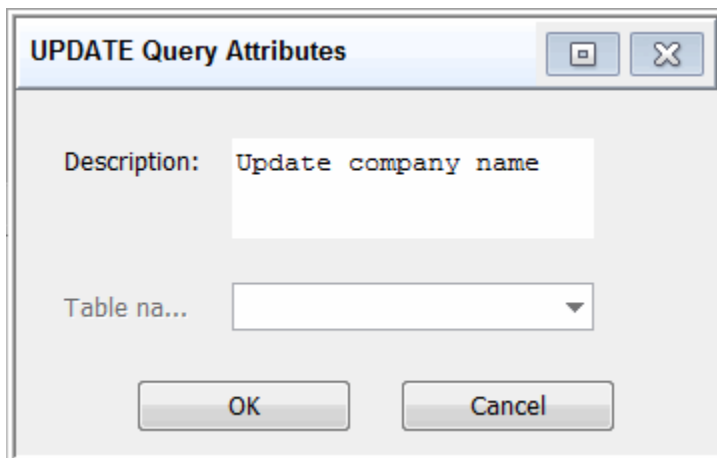
Insert queries cannot be used as the source of a form or as a data source for another query.

Update Query

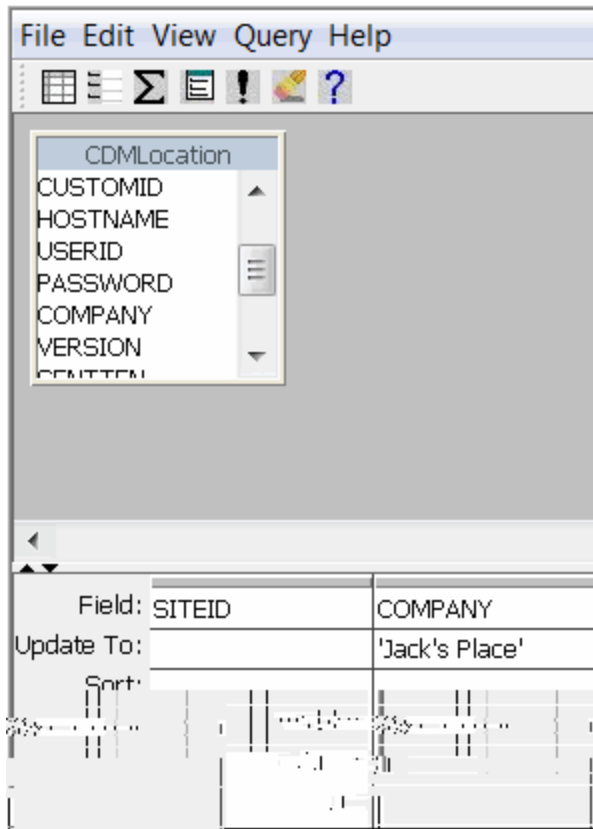
An *Update* query creates a result set like a *Select* query and updates each row in the result set using an expression that you provide.

Setting up an Update Query

1. Select the *Setup* module.
2. Expand the *Application* menu.
3. Right-click *Queries*, and select *New* from the shortcut menu that appears. The *Select tables to query* dialog opens, which lists the tables and queries that you can use as data sources.
4. Select the tables and queries to use as data sources.
5. Select the desired columns from the selected data sources.
6. Select *Update* from the *Query* menu. The *UPDATE Query Attributes* window opens.
7. Type a description of the query.
8. Click *OK*.



9. For each column of the query to update, type an expression in the *Update To* row that will evaluate to the new column value. Column values without *Update To* expressions will not be changed.



Running an Update Query

When you run an *Update* query, you are prompted to confirm the number of rows to update. Upon confirmation, the rows of the result set are updated.

To disable the warning message when running action queries in expressions: Call *ShowWarnings(false)* before running the query to turn warnings off, and *ShowWarnings(true)* after running the query to turn warnings back on.

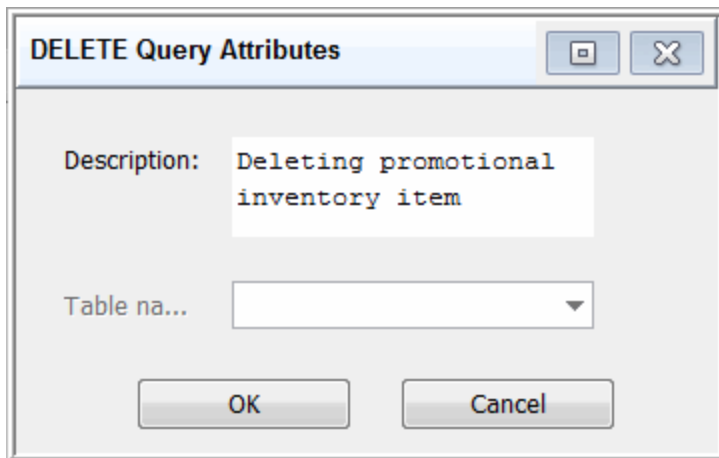
Update queries cannot be used as the source of a form or as a data source for another query.

Deleting Rows

A *Delete* query creates a result set like a *Select* query and deletes each row in the result set from the underlying tables.

Setting up a Delete Query

1. Select the *Setup* module.
2. Expand the *Application* menu.
3. Right-click *Queries*, and select *New* from the shortcut menu that appears. The *Select tables to query* dialog opens, which lists the tables and queries that you can use as data sources.
4. Select the tables and queries to use as data sources.
5. Select the desired columns from the selected data sources to make the table.
6. Select the desired columns from the selected data sources to delete.
7. Select *Delete* from the *Query* menu. The *DELETE Query Attributes* window opens.
8. Type a description of the delete action in the , and then click *OK*.



Running a Delete query

When you run a *Delete* query, you are prompted to confirm the number of rows to delete. Upon confirmation, the rows of the result set are deleted from the underlying tables.

To disable the warning message when running action queries in expressions: Call *ShowWarnings(false)* before running the query to turn warnings off, and *ShowWarnings(true)* after running the query to turn warnings back on.

Delete queries cannot be used as the source of a form or as a data source for another query.

Changing a Query Design

The following describes how to open a the Query design window where you can change the design of a query.

1. Select the *Setup* module.
2. Expand the *Application* menu.
3. Expand the *Queries* menu.
4. Right-click the query to change, and then select *Design* from the menu that appears. The Query designer window opens enabling you to add, change or delete columns and data sources.

Copying Queries

The following describes the methods for copying existing queries to create new queries.

Copying a Modified Query to Create a New Query

Follow these steps to create a new query based on the modified design of an existing query.

1. Select the *Setup* module.
2. Expand the *Application* menu.
3. Expand the *Queries* menu.
4. Right-click the query to copy, and then select *Design* from the menu that appears. The Query designer window opens enabling you to add, change or delete columns and data sources.
5. Select *Save As* from the *File* menu.
6. When prompted, type a unique name for the query.

Copying a Query from the Query List

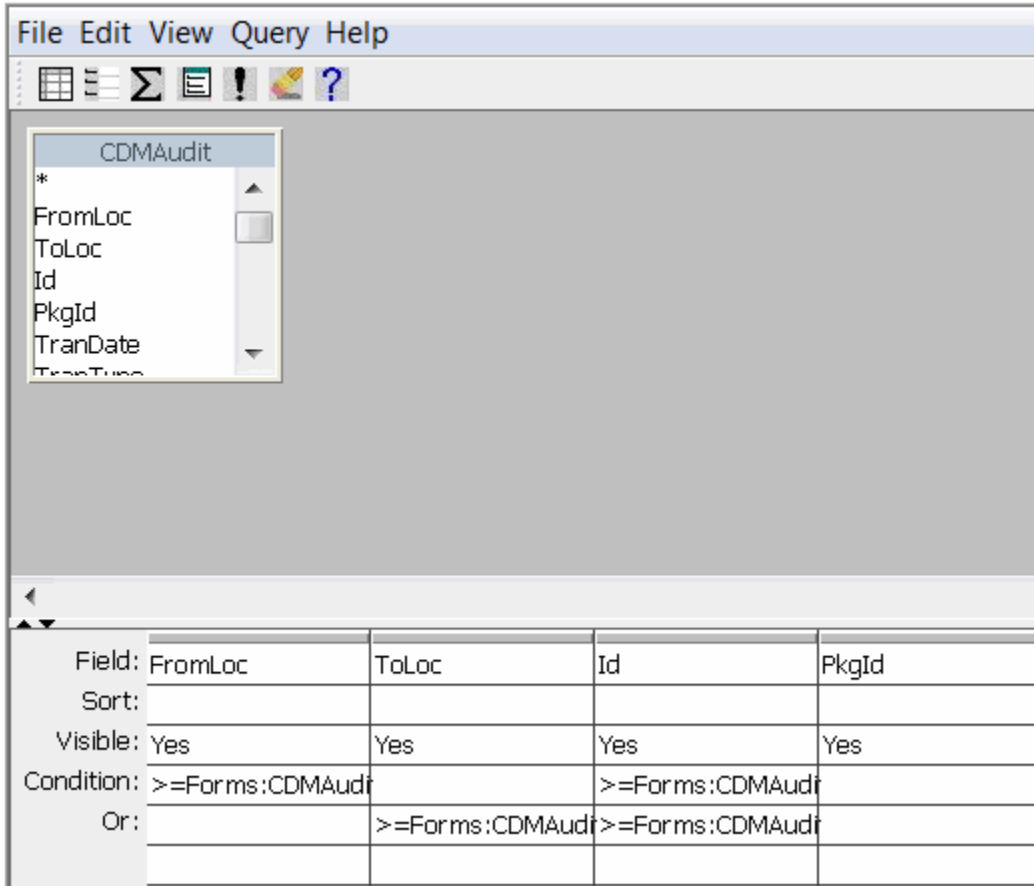
Follow these steps to copy a query from the query list.

1. Select the *Setup* module.
2. Expand the *Application* menu.
3. Expand the *Queries* menu.
4. Right-click the query to copy, and then select *Copy* from the menu that appears.
5. Right-click any query, and then select *Paste* from the menu that appears.
6. When prompted, type a unique name for the query.

Getting Query Parameters from a Form

To display a form where another user enters query parameters, you need to setup the *Condition* value of the query fields, so that the query field values are retrieved from the form.
 For example, an Audit Trail form that lets the user set the parameters that limit the transactions that are included in the query.

The following sample screen shows the *Condition* for the *FromLoc* column is set to: `>=Forms:CdmAuditFilter:FromLoc` and `<=Forms:CdmAuditFilter:ToLoc`



Forms

Forms are used to enter and edit data in tables. When creating forms, you can control the layout of the fields, the fonts and colors, and the types of controls. Usually, one form is created for each table that contains data to be edited. Whenever you need to add or edit a record in the table, you simply open the related form.

The following sample screen shows a form for editing locations and groups.

Record: 1 | Search: |

Select site by name: Price Tier 2 - \$7.29 Lunch, -2

Select site by number: -2, Price Tier 2 - \$7.29 Lunch

Location ID: -2
Name: Price Tier 2 - \$7.29 Lunch
Transaction version: 7
Custom ID: <NULL>
Transfer host: <NULL>
Transfer user id: <NULL>
Transfer password: <NULL>
Transfer company: <NULL>

View Transaction Status

Location Notes
<NULL>

Locations (or groups) in this group:

	Location
*	2185, 2185 - Lake Charles, LA
	2214, 2214 - Alexandria, LA
	2235, 2235 - Des Moines, IA
	2287, 2287 - Champaign, IL
	2402, 2402 - Pineville, LA
+	<NULL>

Add Locations To This Group

Form Sections

Forms are divided into the following sections.

Header	The header section is displayed at the top of the form window.
---------------	--

Detail	The detail section is displayed between the header and footer sections. Most of the form controls are located in this section. Controls with a defined field source should only be added to this section. Controls can be configured as not visible, if necessary.
Footer	The footer section is displayed at the bottom of the form window.

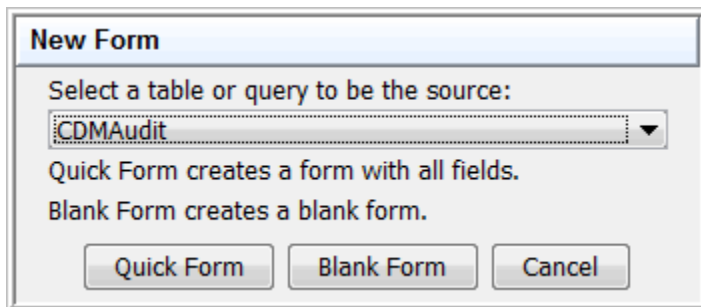
See Also

[Creating a Form Designer Windows Controls](#)
[Form Sections](#)
[Form Attributes](#)
[Events](#)
[Opening Forms](#)
[Navigating Forms](#)
[Record Selectors](#)

Creating a Form

Follow these steps to create a new form.

1. Select the *Setup* module.
2. Expand the *Application* menu.
3. Right-click *Forms*, and then click *New* from the menu that appears. The *New Form* window opens prompting you to select a *data source* for your form. The data source is the table or query that contains the data the form reads. If you are creating a form that prompts for information from a user that does not come from a table or query, leave the source field blank.



4. Select the type of form to create.

Quick Form	The system creates the form and all fields from the selected data source are added to it.
Blank Form	The system creates the form, but no fields from the selected data source are added to it. If you did NOT select a data source and you plan to edit data from a table or query using the form, then you will need to set the <i>Source</i> attribute of the form to the table or query to edit.

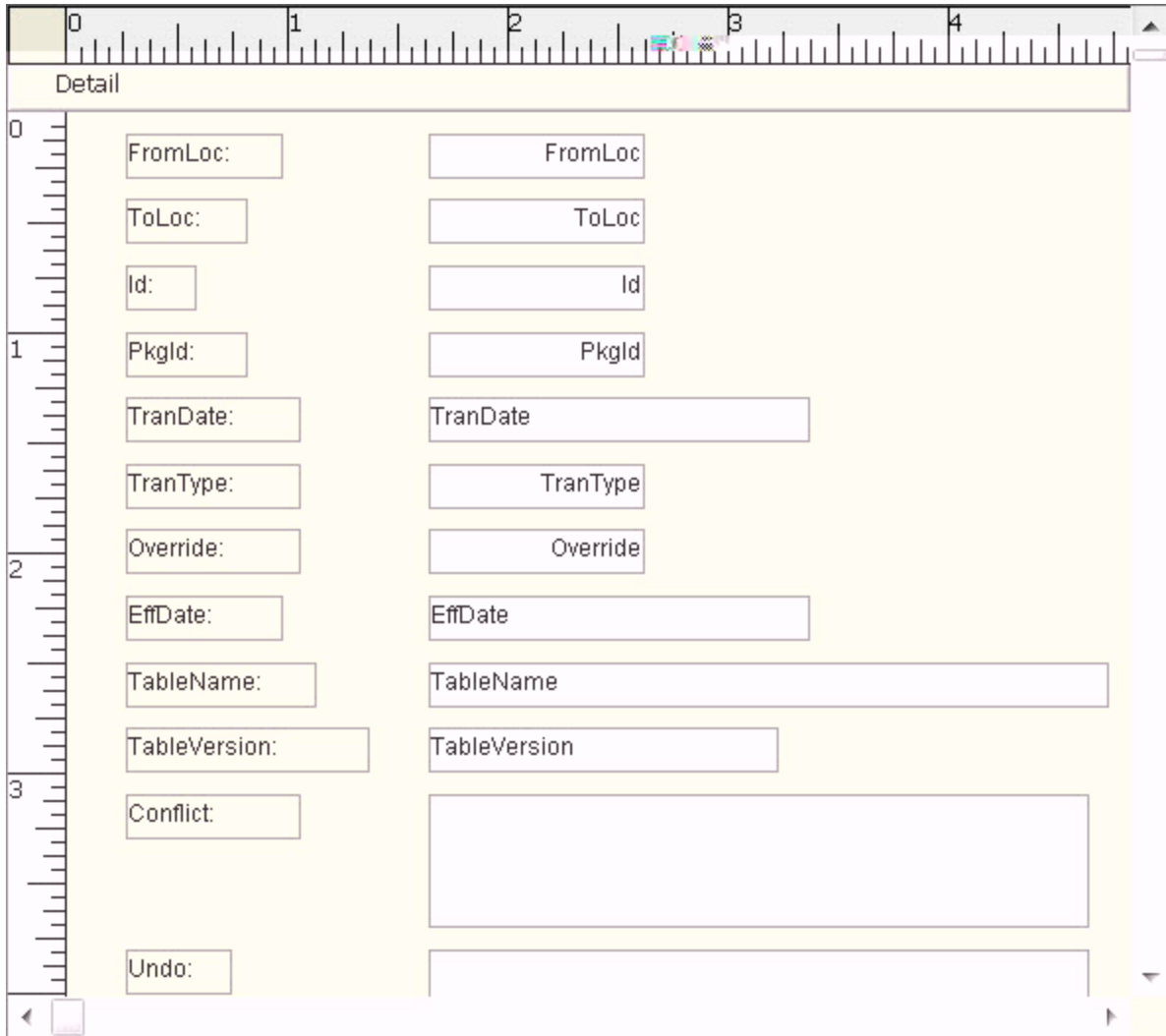
Designer Windows

The pages in this section contain information about designing EDM forms.

Topics

[Form Design Window](#)
[Toolbar Tools](#)
[Attribute Window](#)

Form Design Window



The form design window is divided into several parts. At the top of the window is a ruler that shows the width and horizontal position of form controls. The main detail body of the window is divided into the form sections. Each form section has a vertical ruler along the left side of the window that shows the height and vertical position of form controls. There is a section header window at the top of each section with the section name.

When designing a new form, only the main detail section is viewable. The right boundary of the sections is a black vertical line. The bottom boundary of each section is a black horizontal line, which can be dragged up and down to change the section's height. Controls selected from the Control toolbox can be added to each section.

Toolbar Tools

The form designer tools are displayed on the toolbar beneath the menu.

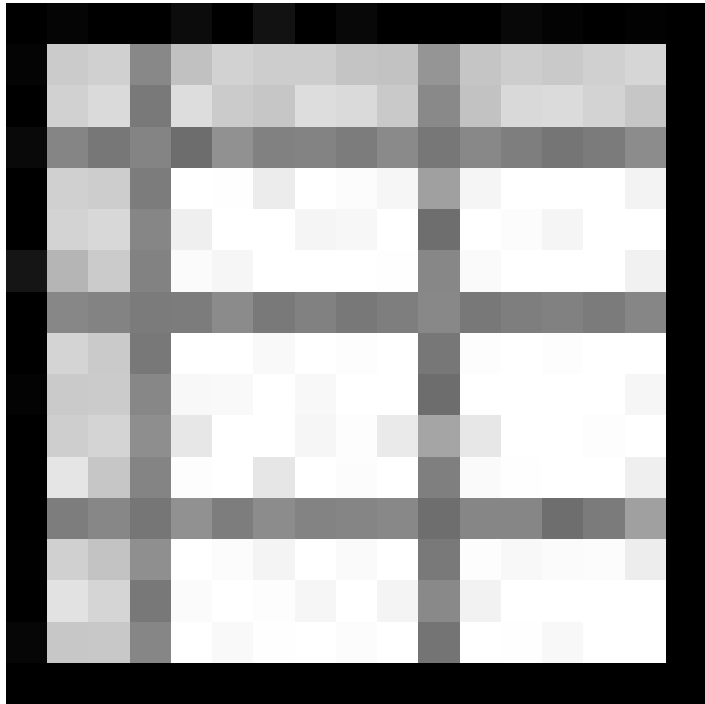


Tool name	What it does
-----------	--------------



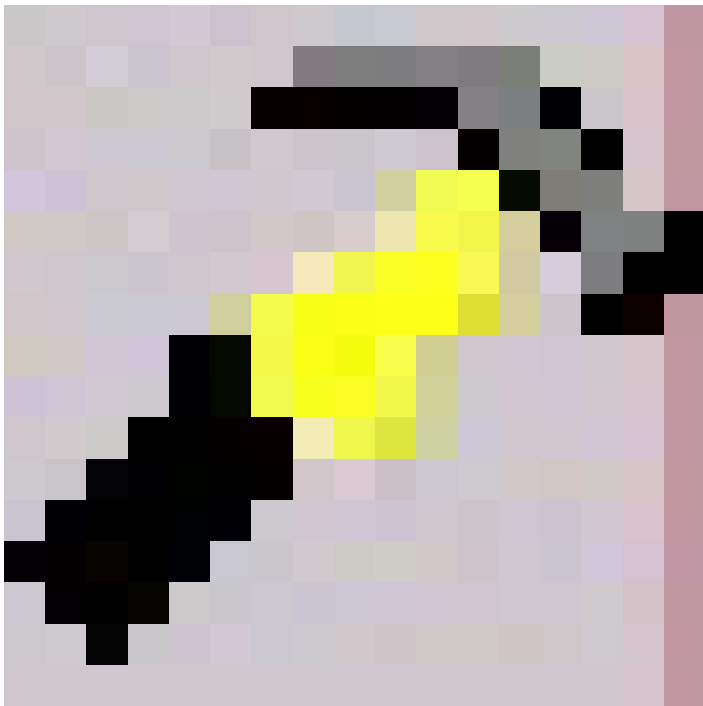
Form

Opens the form in normal view for editing data



Spreadsheet

Opens the form in spreadsheet view for editing data



Tool box

Shows/hides the control toolbox, which is used for creating controls



Field List

Displays the list of fields from the data source tables that are used for the form



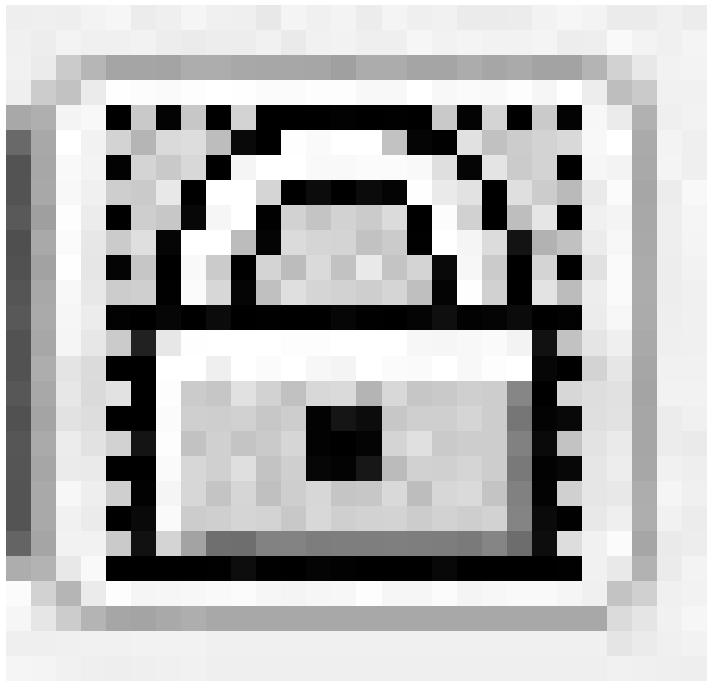
Palette

Changes the color of controls and borders

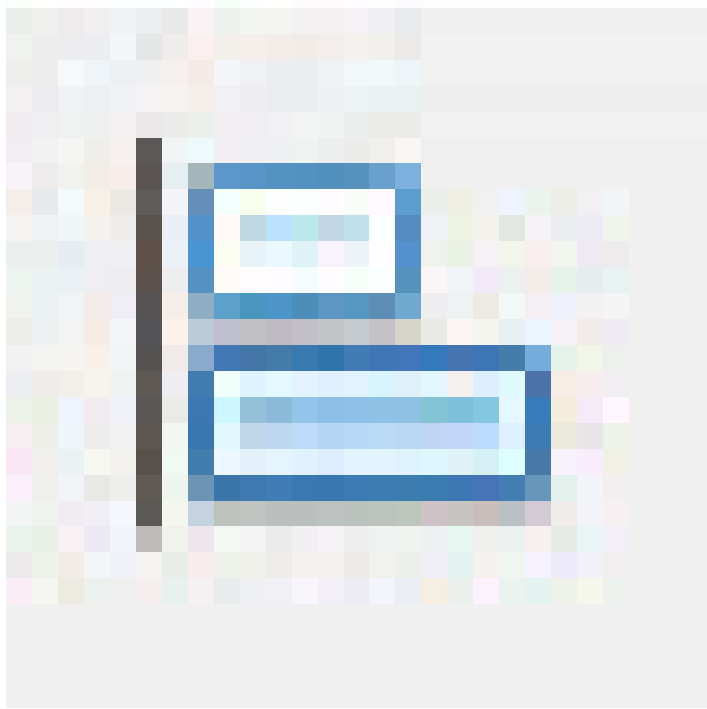


Attributes

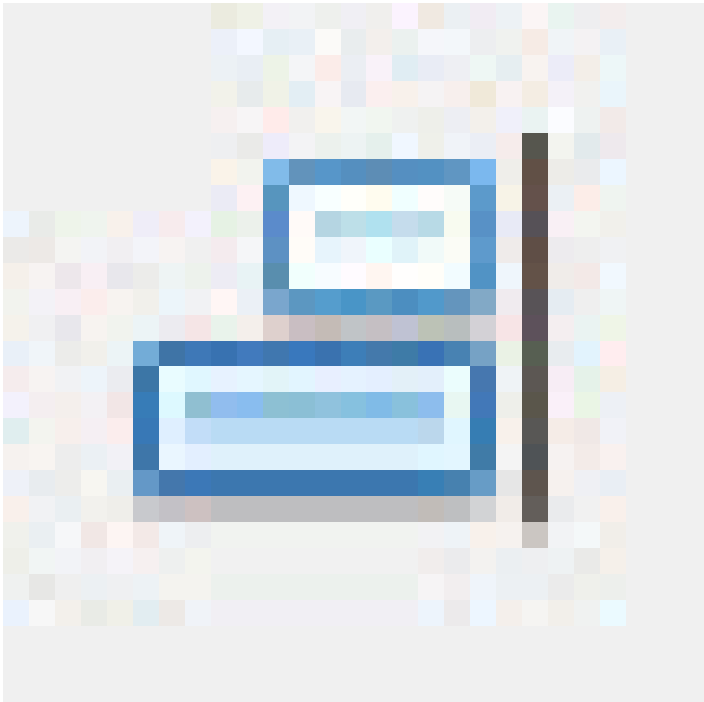
Shows or hide the attribute window



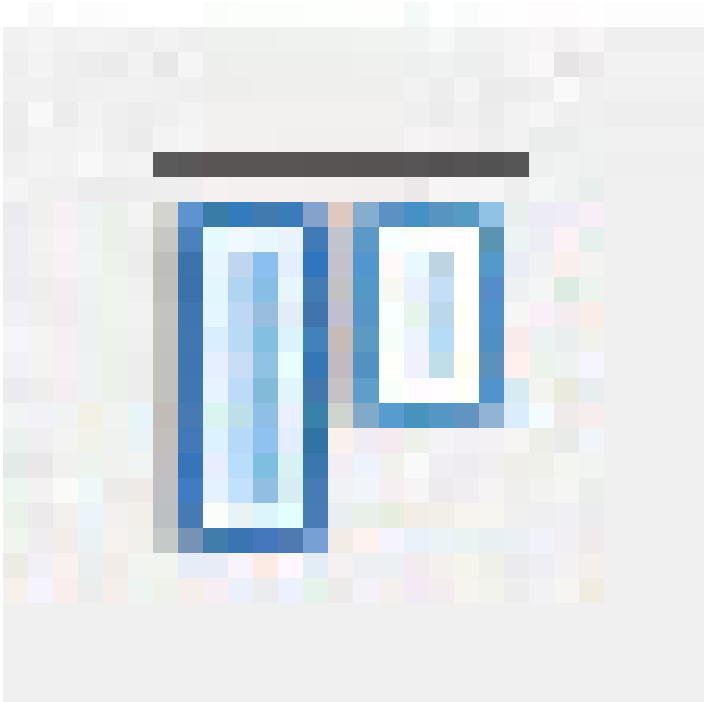
When selected, controls cannot be repositioned or resized



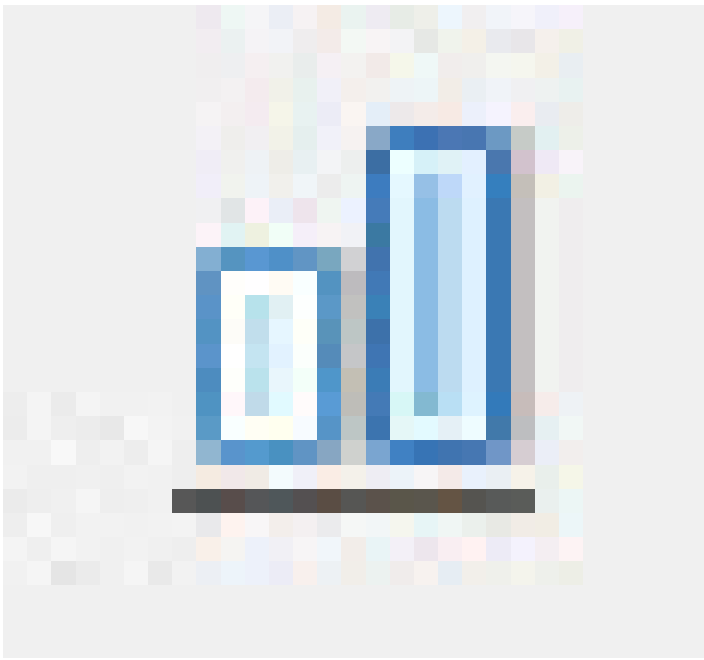
Left-aligns selected controls



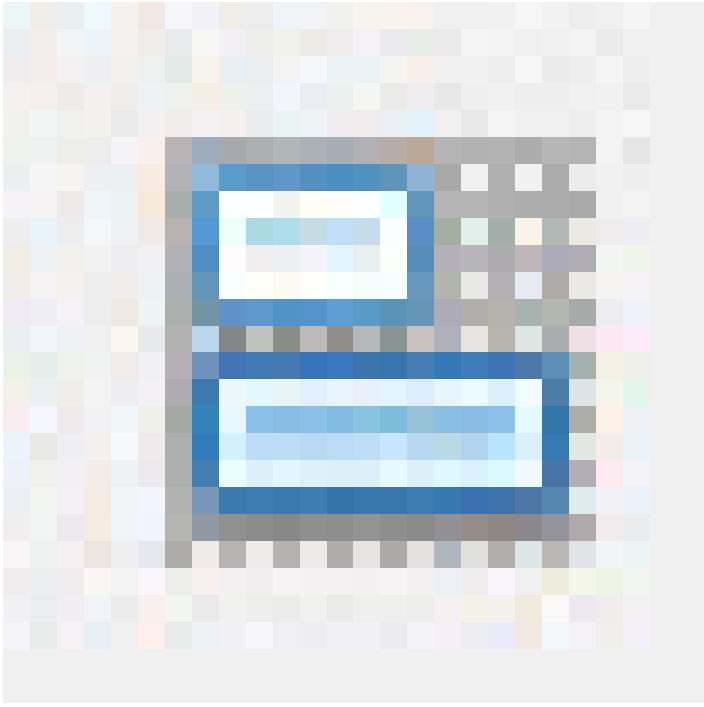
Right-aligns selected controls



Top-aligns selected controls



Bottom-aligns selected controls



Aligns selected controls to the grid


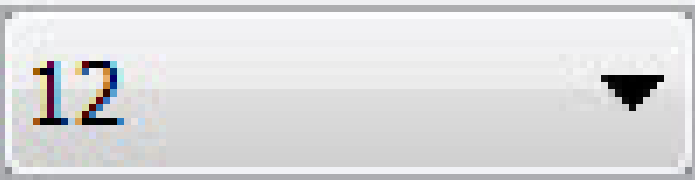


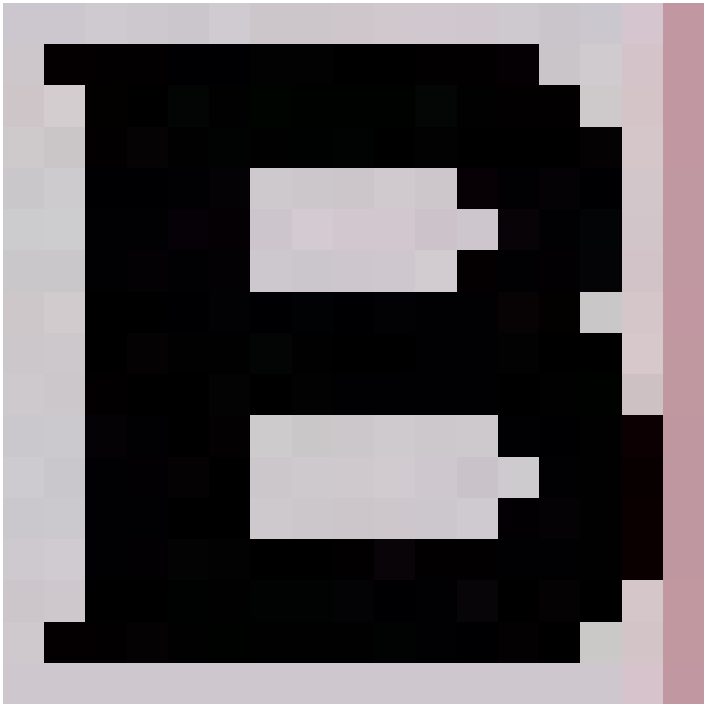
Help

Displays help information

Form Tools Available when Controls are Selected

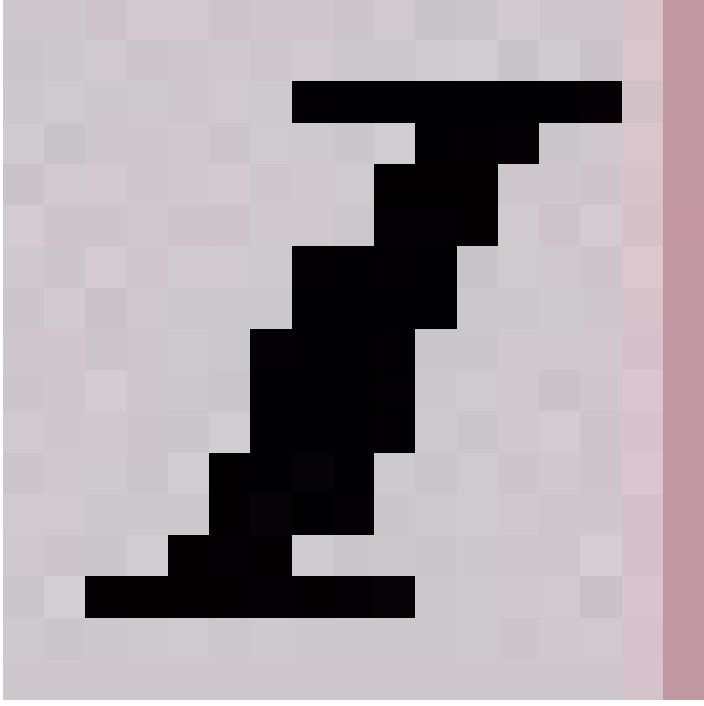
The following tools are available for controls when the controls are selected/highlighted.

Tool name	What it does
 <p>Font</p>	Sets the font for the selected fields
 <p>Font Size</p>	Sets the font size for the selected fields



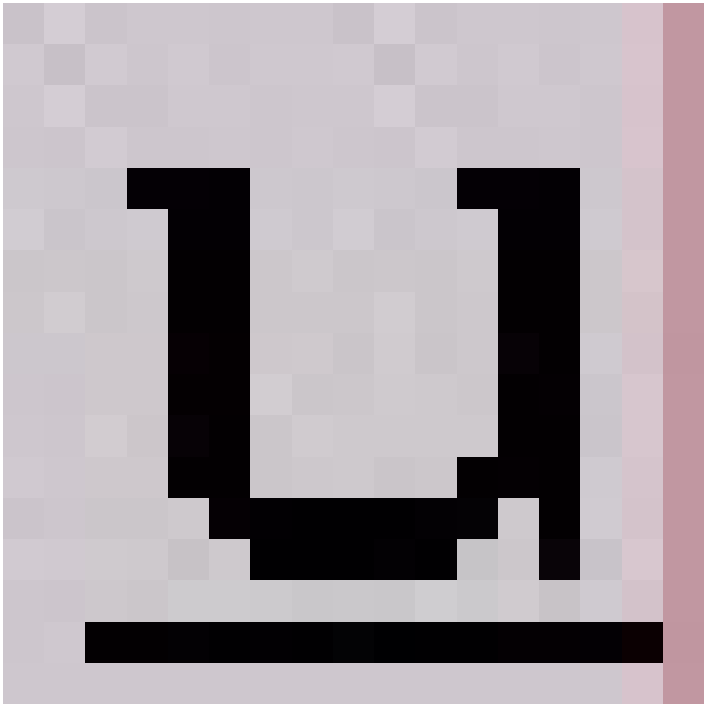
Bold

Sets the text for the selected fields to bold



Italic

Sets the text for the selected fields to italic



Underline

Underlines the text for the selected fields

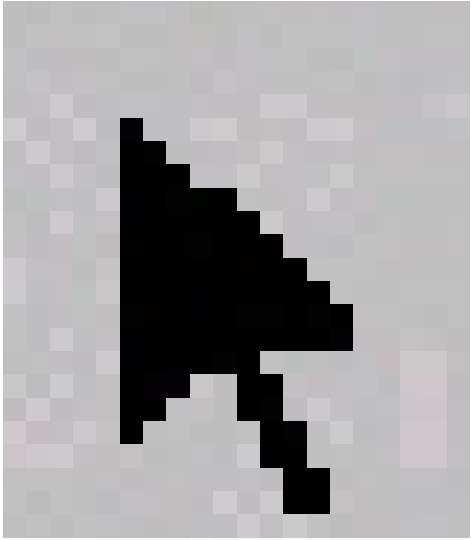
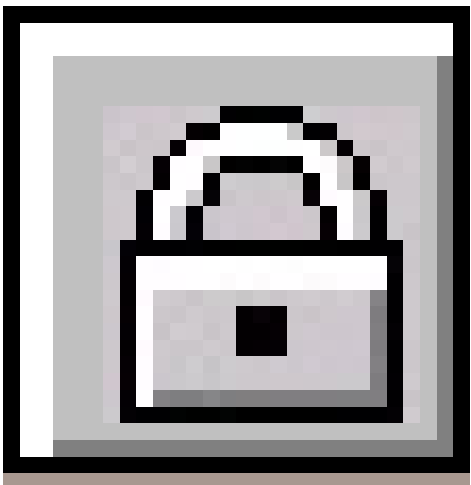


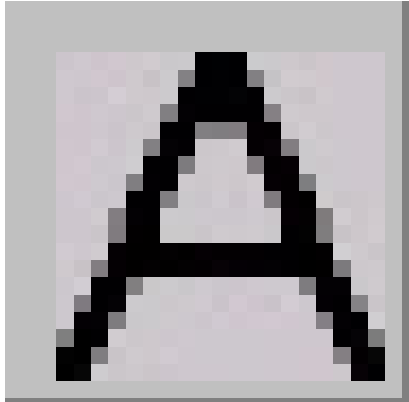
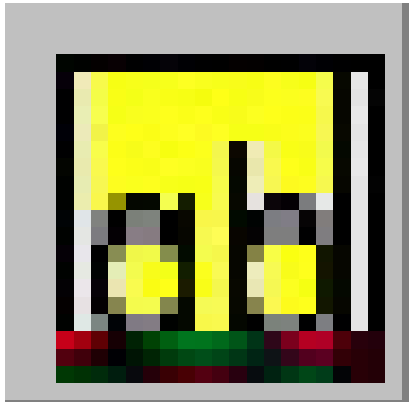
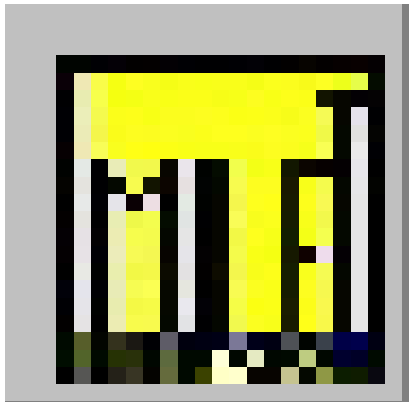
Left Justify

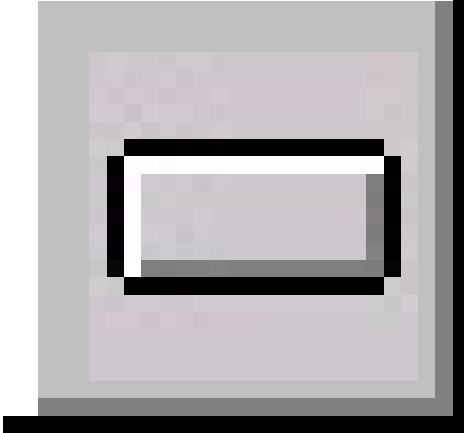
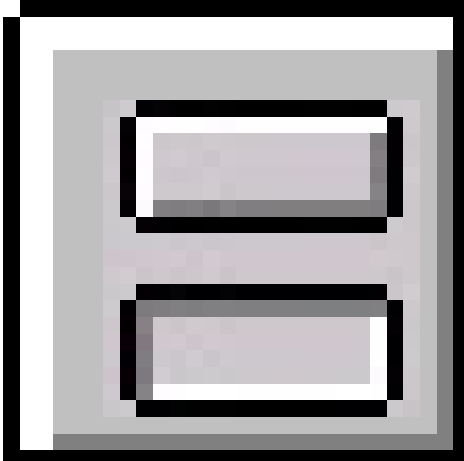
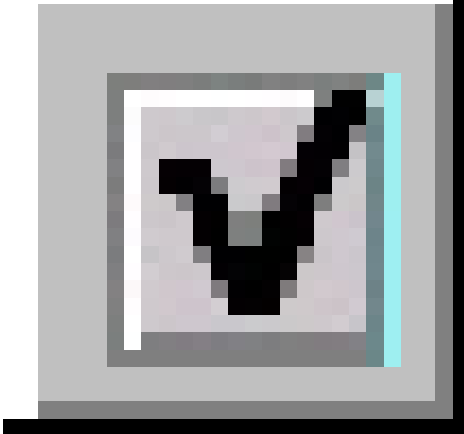
Left-justifies the text for the selected fields

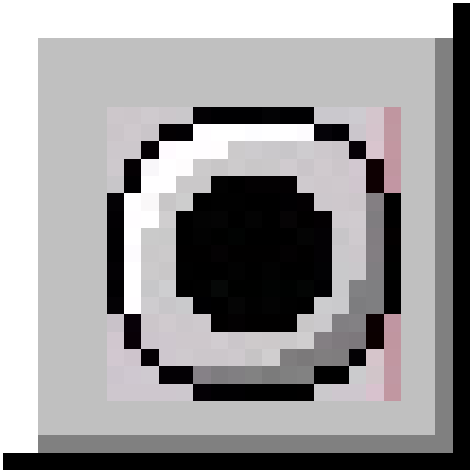
Center	



Control tool	What it does
	<p>Pointer</p> <p>When depressed, you can select\move\size controls and select\size sections</p>
	<p>Lock</p> <p>When depressed, the current control tool will stay selected until you select a different control tool</p>

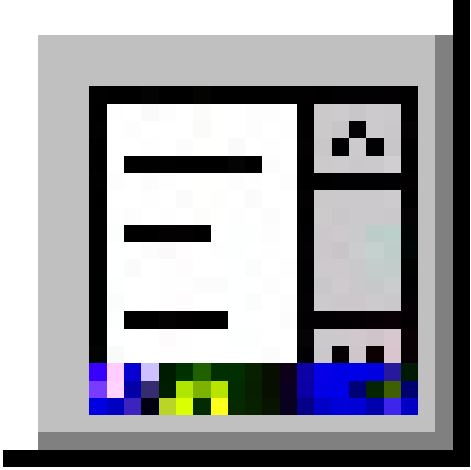
	Static	Creates a text label that can't be edited
	Text	Creates a text entry field that can be edited
	Multi-line text	Creates a multi-line text entry field that can be edited

	Button	Creates a button that evaluates the expression defined for the <i>On Push</i> attribute of the control
	Toggle Button	Creates a toggle button that is depressed when its source is <i>True</i> and not depressed when its source is <i>False</i>
	Check Box	Creates a checkbox that the user can check/uncheck



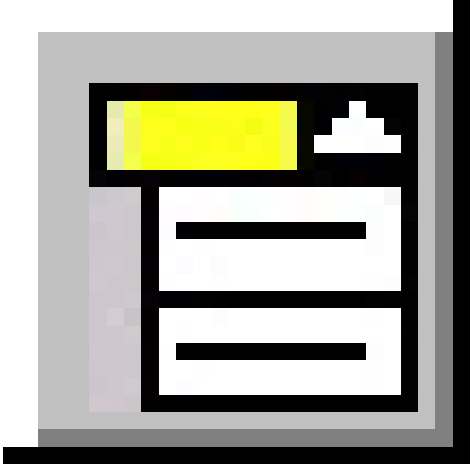
Radio Button

Creates a radio button that the user can turn on/off



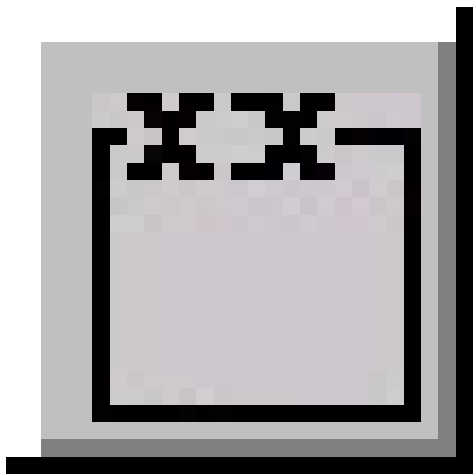
Listbox

Creates a listbox that displays the rows in a table or query, a list of values or a list of fields in a table or query



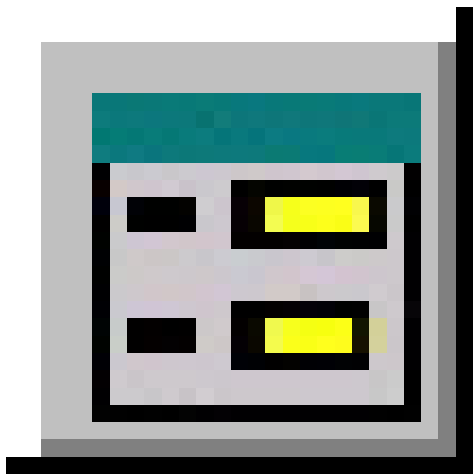
Combo Box

Creates a combo box, which is an entry field combined with a listbox that allows the user to enter a value or select one from the list



Group Box

Creates a box to group toggle buttons, checkboxes or radio buttons



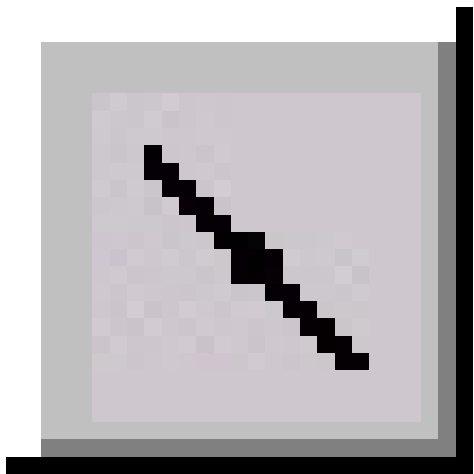
Sub form

Creates a sub form, which is used to display related information from a table or query that is not a data source for the form



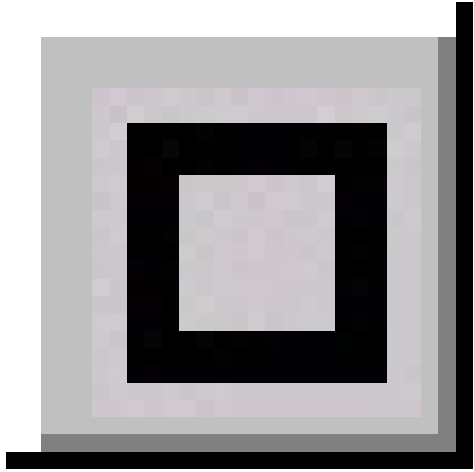
Bitmap

Creates a bitmap control used to display a picture



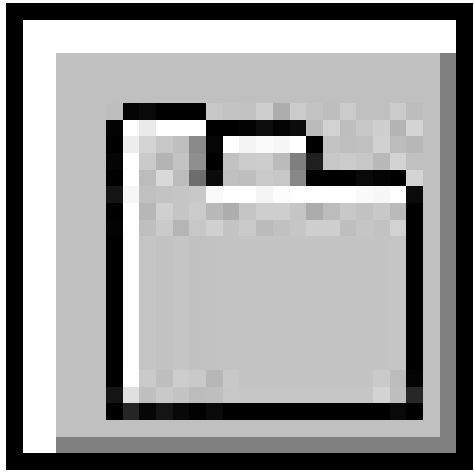
Line

Draws a line on the form



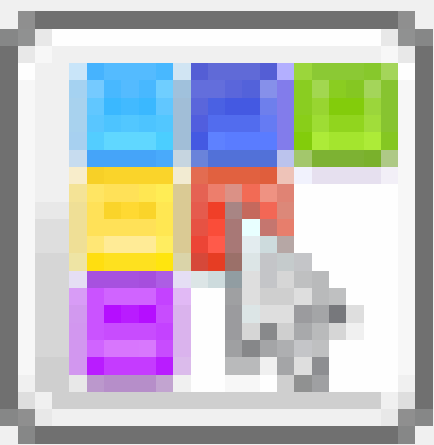
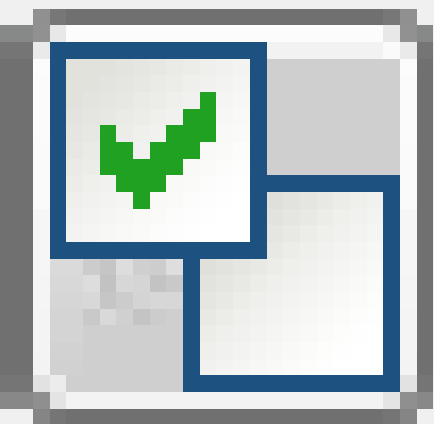

Box

Draws a box on the form



Tab
Control

Creates a tab control allowing related fields to be grouped together on separate tabs


 A pixelated icon showing a color palette with six colored squares (blue, purple, green, yellow, red, purple) and a mouse cursor pointing at the red square.	Color Selector	Presents the user with a color selector
 A pixelated icon showing a blue-bordered window with a green checkmark and a grey background.	Bit Editor	Bit editor to allow user to check boxes for a single integer-based field
 A pixelated icon showing a calendar with a red header and a clock face.	Date and Time Entry	Date and Time Entry control for displaying data

Attribute Window


The attribute window is used to edit the attributes for a form, a section or a control.

Form	
Description	
Source	CDMAudit
Title	CDMAudit
Acting key fields	
Allow copy	
Allow insert	
Begin insert	
Before insert	
After insert	
Cancel insert	
Allow update	
Begin update	
Before update	
After update	
Cancel update	
Allow delete	
Before delete	
After delete	
Cancel delete	
Before current	
After current	


Displaying the Form Attributes

1. Click the white area outside any sections -OR- click the gray square at the top left-corner of the form design window (where the rulers meet).
2. Click the attributes button

 on the toolbar.

Displaying Section Attributes

1. Click the section header.
2. Click the attributes button

 on the toolbar.

Displaying Control Attributes

1. Click the control. To select boxes or group boxes, you must click the border of the box. To select a line, you must click the line itself rather than inside the rectangle that bounds the line.
2. Click the attributes button

 on the toolbar.

Controls

The pages in this section describe how to create and work with controls when designing EDM forms. Examples of controls include: Text Entry Fields, Listboxes, and Radio Buttons.

Topics

[Basic Controls](#)
[Setting Control Attributes](#)
[Selecting Controls](#)
[Moving Controls](#)
[Sizing Controls](#)
[Copying Controls](#)
[Deleting Controls](#)
[Setting the Tab Order for Controls](#)
[Advanced Controls](#)

Basic Controls

The pages in this section contain information about creating basic controls when designing EDM forms.

Topics

[Creating Controls](#)
[Creating Text Entry Controls](#)
[Creating Labels with Static Controls](#)
[Creating Multi-Line Text Controls](#)

Creating Controls

Form controls are the fields on a form, such as labels, data entry fields and buttons. Controls can be added to any section of the form and may be moved and sized any way you like. The following table describes the three types of controls.

Bound Control	Used for displaying and editing data from a field in a table or query
Unbound Control	Typically used when you are creating a form to prompt the user to enter parameters for a query, or when you want the user to enter a value that is used by another part of the form. The <i>Source</i> attribute of unbound controls is blank, which means that it does not display or update data from the database.
Calculated Control	Read-only control that displays the results of a calculation or function call. A calculated control does not update any fields in the database, and users cannot enter any data into it. You can use calculated controls to display information like the total of Quantity * Price when entering order items. To create a calculated control, enter an equal sign (=) in the <i>Source</i> attribute of the control followed by the calculation. For example, to show Quantity * Price, create a text control and enter =Quantity*Price in the <i>Source</i> attribute.

Displaying or Hiding the Control Toolbox

Click the *Toolbox* tool



on the toolbar.

Adding a Control to a Form

1. Select the control button from the control toolbox.
2. Click the form section where you want to place the upper-left corner of the control.

Creating a Bound Control

1. Select the control that is already added to the form.
2. Double-click the field in the *Field List* window to display in the control.

You can also set the *Source attribute* for the control to the name of a field in one of the form's data sources. When a form is displayed, the bound controls display the data from the field identified by their *Source* attribute. When the user changes the value in a bound control, the related field value is updated in the data source.

Creating Several Controls of the Same Type

1. Select the control button from the control toolbox.
2. Click the *Lock* tool



from the control toolbox to depress it.

3. Click the form where you want each control to be placed.
4. Click the *Lock* tool again, so that it is not depressed.
5. Click the *Pointer* tool

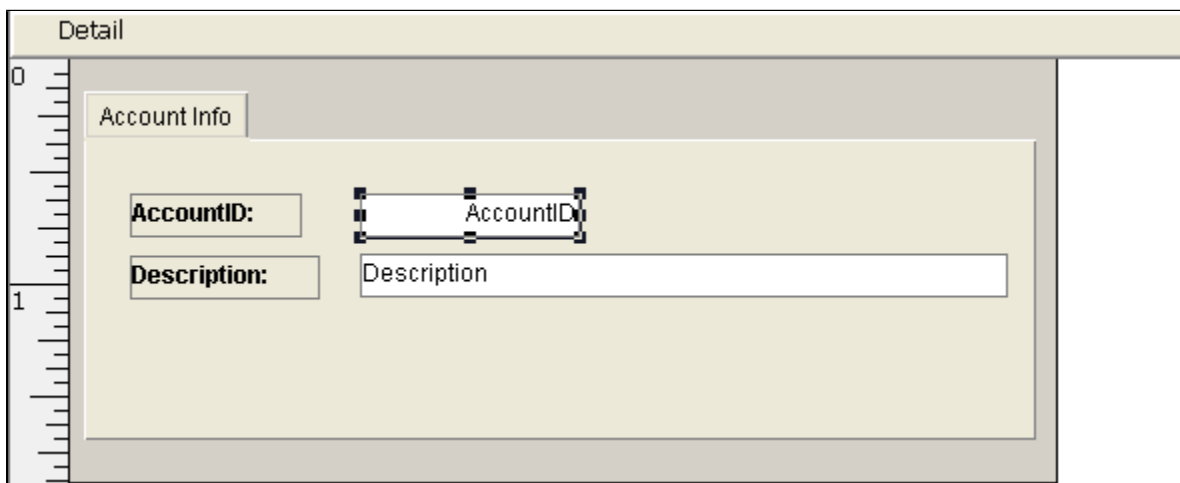


to return to select mode.

Creating Text Entry Controls

Text controls are used to display and edit data. The data may be from the form's data source, the results of a calculation or provided by the user.

The *Source attribute* for a control determines what data is displayed in the field. The following sample screens show a form design with an entry field to edit the *AccountID* field.

A screenshot of a software interface showing a form design. The form has a title bar labeled "Detail". Below the title bar is a vertical ruler with markings from 0 to 1. The form contains a section titled "Account Info". Inside this section, there are two text entry controls. The first control is labeled "AccountID:" and contains the text "AccountID". The second control is labeled "Description:" and contains the text "Description". The "AccountID" control is currently selected, as indicated by a dashed border and small handles around it.

Text	
Name	AccountID
Source	AccountID
Format	
Length	10
Max decimals	0
Assumed decimals	0
Default value	0
Required	No
Status bar text	AccountID
Validation rule	
Validation text	
Begin update	
Before update	
After update	
On enter	
On exit	
On double click	
Tab stop	Yes
Visible	Yes
Enabled	Yes

Creating a Bound Text Control

1. Click the text control tool



in the control toolbox.

2. Click the form where you want the top left-corner of the control to appear.
3. Double-click the field in the Field List window to display in the control. All the attributes that are defined for that field in the table are copied to the control, including the status bar text, validation rule and text, field size and name.

Creating an Unbound Text Control

1. Click the text control tool



in the control toolbox.

2. Click the form where you want the top left-corner of the control to appear.

Creating a Calculated Text Control

1. Click the text control tool



in the control toolbox.

2. Click the form where you want the top left-corner of the control to appear.
3. Type an equal sign (=) followed by the calculation in the *Source attribute* for the control. For example, to create a calculated control that displays the value in the *Quantity* field multiplied by the value in the *Price* field, type =Quantity*Price in the *Source* attribute.

Creating Labels with Static Controls

About Static Controls

Static controls display simple text strings and can be placed anywhere on any section of the form. You can set the size, font, color and border type using the Palette window.

The following sample screens show a form design with a static control as a prompt.

The screenshot shows a form titled "Detail" with a vertical ruler on the left side. The form contains several input fields and a static control:

- Day Part ID:** A text box containing "DayPartID" with a mouse cursor hovering over it.
- Description:** A text box containing "DayPartDesc".
- Start Time:** A text box containing "StartTime".
- End Time:** A text box containing "EndTime".
- Days:** A row of seven checkboxes labeled "Sun", "Mon", "Tue", "Wed", "Thu", "Fri", and "Sat".
- Category:** A dropdown menu containing "CatID".
- Sequence Number:** A text box containing "jenceNum".

The screenshot shows the "Label" property palette for the static control. The properties are as follows:

Property	Value
Name	DayPartID
Caption	
Visible	Yes
Enabled	Yes
Tab control	
Tab page	
Left	356
Top	35
Width	49
Height	15
Help URL	

How to Add a Static Control

1. Click the static control tool



in the control toolbox.

2. Click the form section where you want to place the upper-left corner of the control.
3. Type the text for the control to display in the *Caption* attribute of the control.

Creating Multi-Line Text Controls

Multi-line text controls are used to display and edit text data that consumes multiple lines on the form. The data may be from the form's data source, the results of a calculation or provided by the user. Multi-line text fields are automatically used by the *Quick Form* generator to display data for *Memo* fields.

The following sample screen shows a multi-line text control for long text data.

The screenshot shows the CDMAudit application window with a menu bar (File, Edit, Records, Help) and a toolbar. The main area contains several input fields and controls:

- Record: 1
- From Loc: 0
- To Loc: 1
- Tran Id: 4
- Package Id: 0
- Tran Date: 02/20/2002 10:23:47 AM
- Tran Type: Insert record
- Eff Date: (empty)
- User Id: (empty)
- Table: tbl_ItemMaster
- Table Version: 2
- Status: Committed
- Status Date: 02/20/2002 10:23:47 AM
- Conflict: (empty multi-line text control)
- Undo: (empty multi-line text control)
- Fields: "CDMLOCID", "1"
"ItemNum", "-9"
"BaseltemNum", "3000"
"ModifierNum", "0"
"EntreeCode", "0"

At the bottom, there is a label: "Id of location from which transaction came".

Creating a Multi-Line Text Control

1. Click the multi-line text control tool



in the control toolbox.

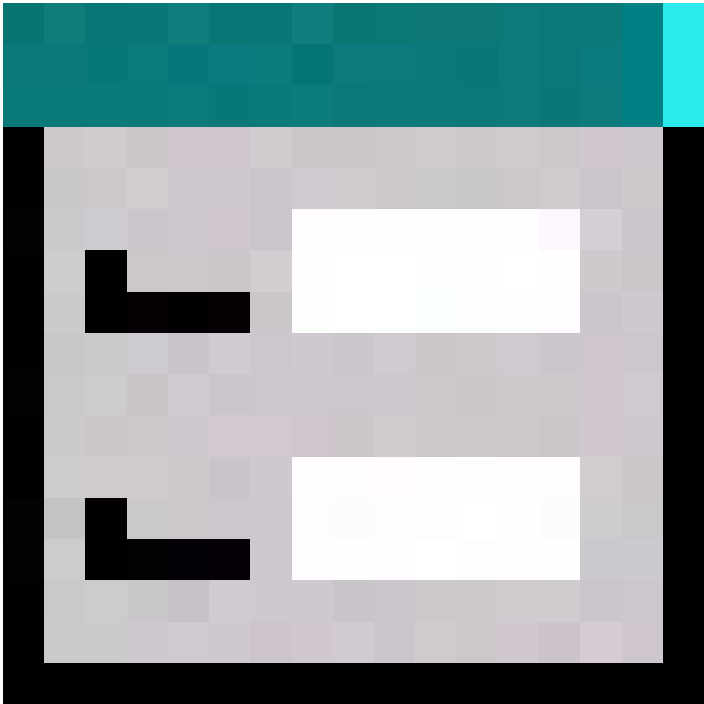
2. Click the form where you want the top left-corner of the control to appear.
3. Double-click the field in the *Field List* window to bind to the control.

Setting Control Attributes

The following describes the toolbar tools used to change control attributes.

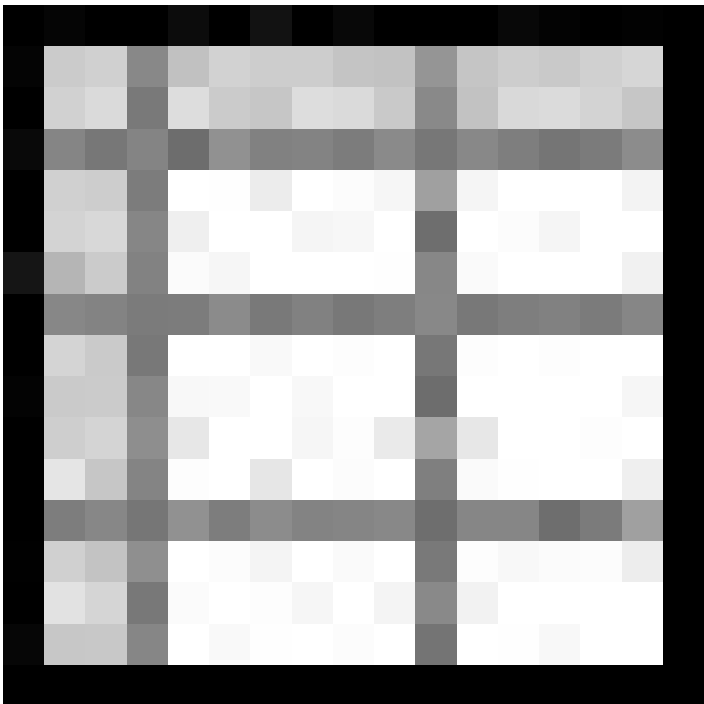


Tool name	What it does
-----------	--------------



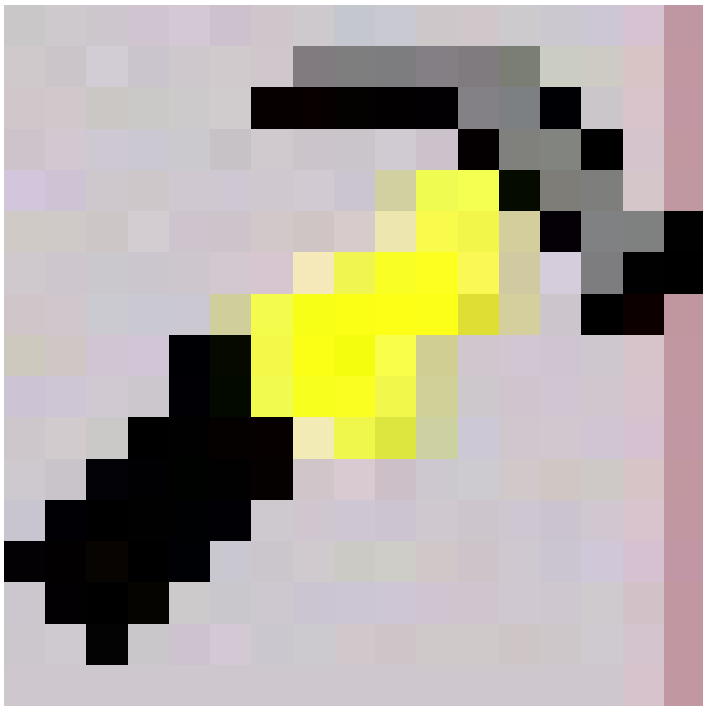
Form

Opens the form in normal view for editing data



Spreadsheet

Opens the form in spreadsheet view for editing data



Tool box

Shows/hides the control toolbox, which is used for creating controls



Field List

Displays the list of fields from the data source tables that are used for the form



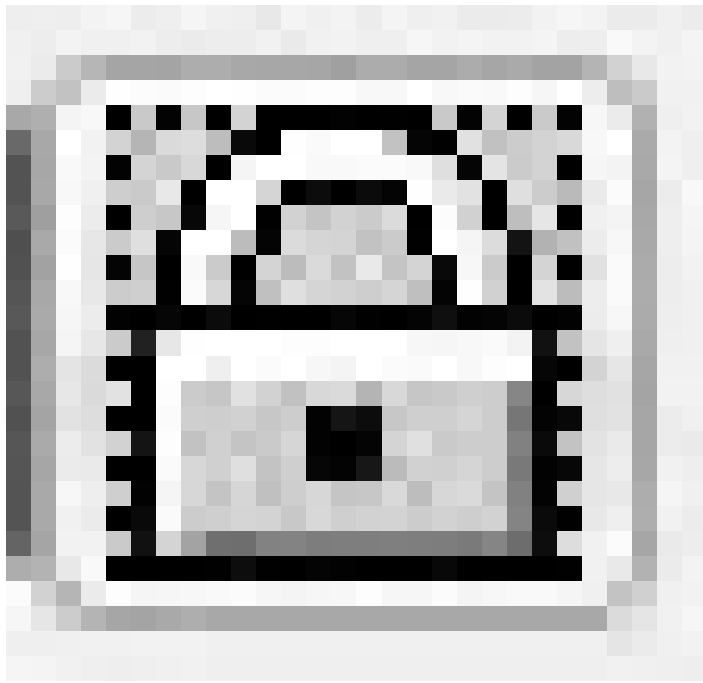
Palette

Changes the color of controls and borders

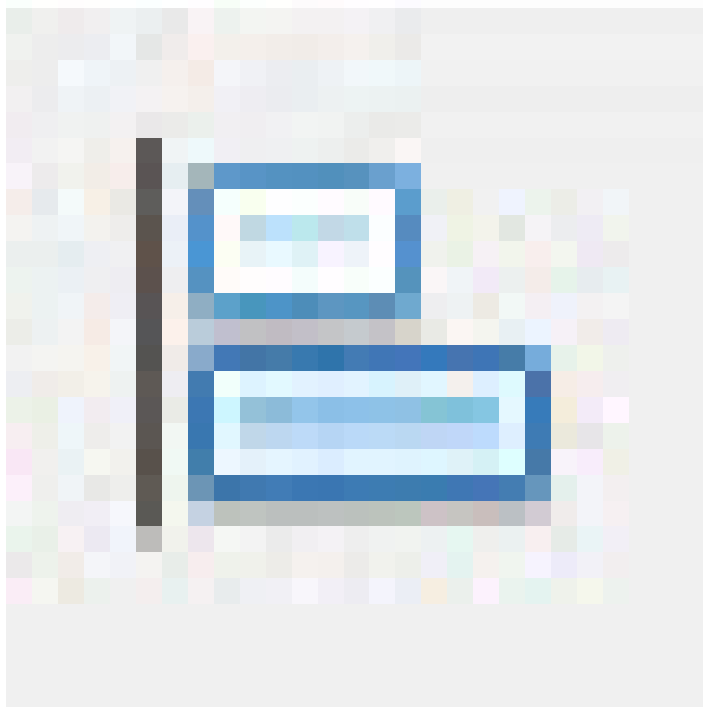


Attributes

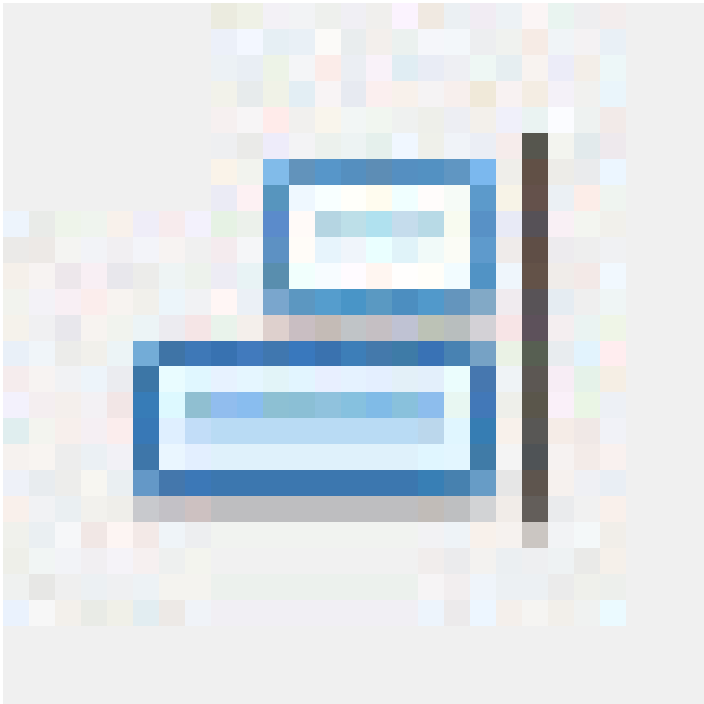
Shows or hide the attribute window



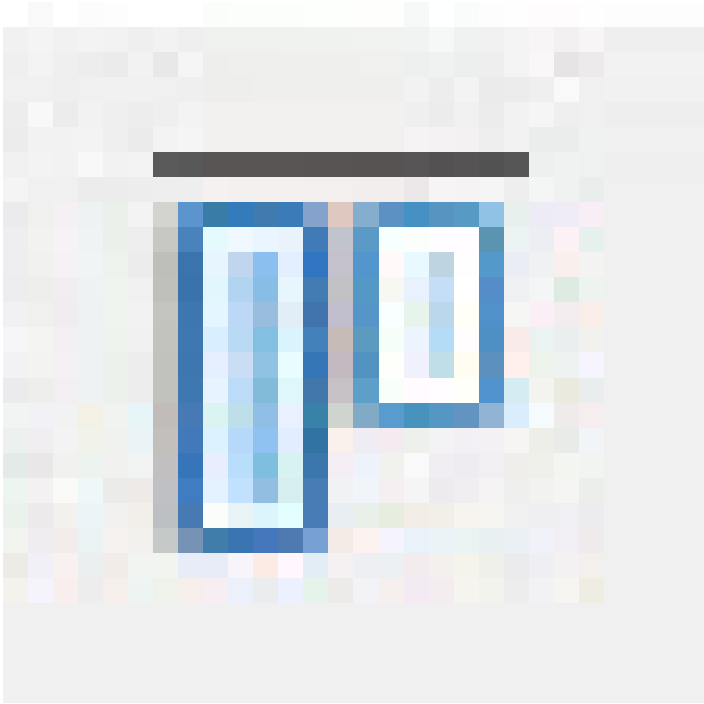
When selected, controls cannot be repositioned or resized



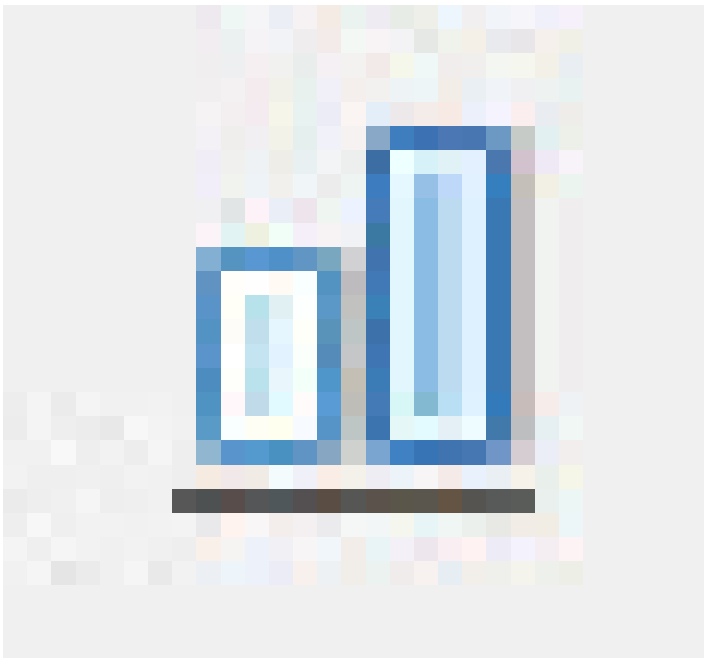
Left-aligns selected controls



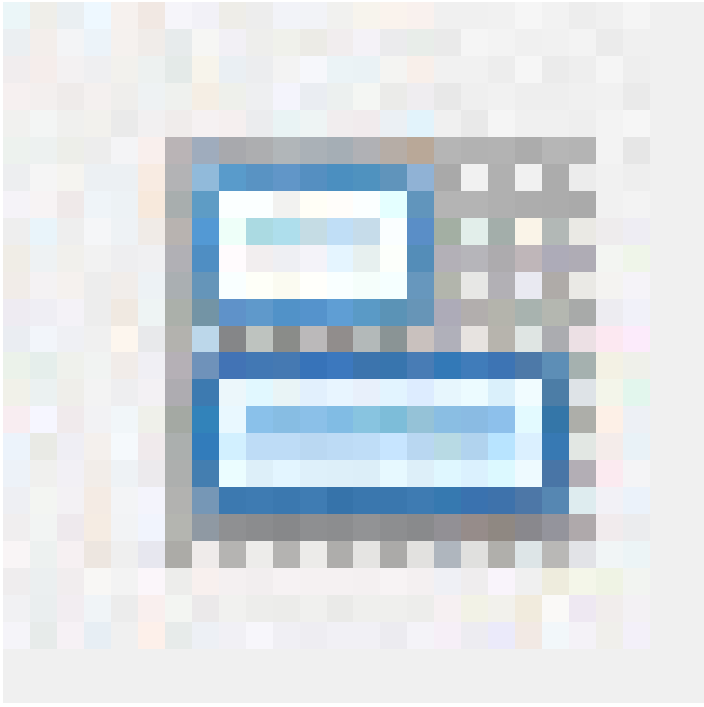
Right-aligns selected controls



Top-aligns selected controls



Bottom-aligns selected controls



Aligns selected controls to the grid



Help

Displays help information

Changing the Attributes for Multiple Controls

1. To select multiple controls: Hold down the left-mouse button and drag the mouse pointer over the controls -OR- hold down the *Ctrl* key on your keyboard and click each control.
2. Click the desired tool from the toolbar.

Control Attributes

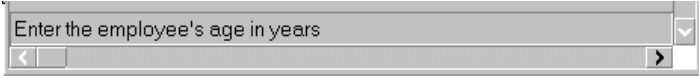
1. Click the control. To select boxes or group boxes, you must click the border of the box. To select a line, you must click the line itself rather than inside the rectangle that bounds the line.
2. Click the *Attributes* tool



on the toolbar.

The following table describes the control attributes that you can configure.

Attribute	Description
Allow null	To force the user to enter data into a field, set the <i>Allow null</i> attribute to <i>No</i> . To allow the user to leave a field blank, set the <i>Allow null</i> attribute to <i>Yes</i> .
Decimal places	Set the <i>Decimal places</i> attribute to the maximum number of decimal places to use in numeric fields. Valid values are 0 to 19.
Default value	Set the <i>Default value</i> attribute to the default value for the field. The user can override the default value by typing over it.
Format	Set the <i>Format</i> attribute to the desired format for the control data, e.g. set the <i>Format</i> attribute to <i>Mmmm d, YYYY</i> to display the date as October 5, 2013. See also Format Strings .
Height	Set the <i>Height</i> attribute to the desired height of the control. Provide the value in hundredths of inches, so if you want a control that is 3 inches high, set the value to 300.
Left	Set the <i>Left</i> attribute to the desired position of the control in relation to the left edge of the form. Provide the value in hundredths of inches, so if you want a control that is 1 inch from the left edge of the form, set the value to 100.
Length	Set the <i>Length</i> attribute to the maximum number of characters to allow in the field

Status bar text	<p>Set the <i>Status bar text</i> attribute to the text to display in the Status Bar when the mouse pointer is hovering over the control. For example, you can display "Enter the employee's age in years" when the mouse pointer is hovering over the <i>Age</i> control.</p> 
Tab stop	Set the <i>Tab stop</i> attribute to <i>No</i> to designate field data as Read-Only.
Top	Set the <i>Top</i> attribute to the desired position of the control in relation to the top edge of the form. Provide the value in hundredths of inches, so if you want a control that is 1 inch from the top edge of the form, set the value to 100.
Validation rule	<p>Use the <i>Validation rule</i> attribute to validate the data that the user types in a field. The <i>Validation rule</i> attribute value is an expression against which the data in the field is validated. For example, to enforce the <i>Married</i> field to contain a Y or N, set the <i>Validation rule</i> attribute to 'Y' or 'N'.</p> <p>You don't include the first operand when defining the expression since the field value is used as the first operand. For example, to validate that the <i>Age</i> field contains a value >21, type >21 in the <i>Validation rule</i> attribute.</p> <p>Validation expressions can be combined using the <i>AND</i> and <i>OR</i> operators. For example, to validate that the <i>Age</i> field contains a value >21 and <65, type >21 AND <65 in the <i>Validation rule</i> attribute.</p> <p>Validation expression format: <i>Operator Operand [AND/OR Condition Expression]</i></p>
Validation text	The <i>Validation text</i> attribute contains the error message to display if the validation rule is not satisfied by the user. If <i>Validation text</i> attribute is blank, then the displayed error message is "Value must be " followed by the <i>Validation rule</i> attribute value.
Visible	Set the <i>Visible</i> attribute to <i>No</i> to hide a control.
Width	Set the <i>Width</i> attribute to the desired width of the control. Provide the value in hundredths of inches, so if you want a control that is 3 inches wide, set the value to 300.

Naming Controls

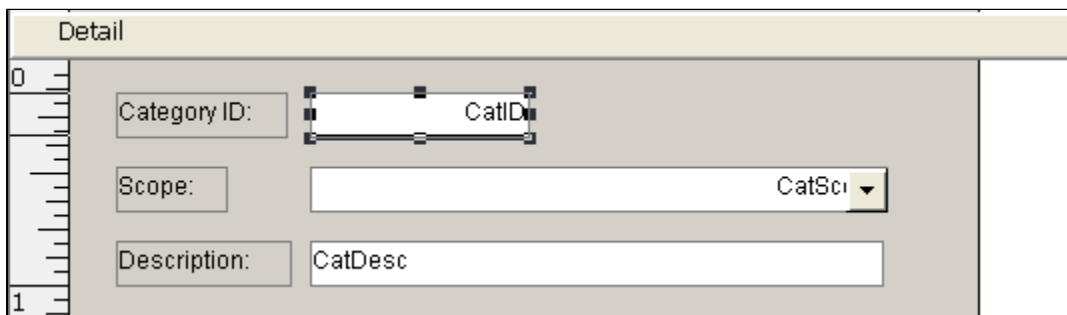
When you add a control to a form, the system assigns it a unique name such as *Field1Field2*, for example. When you bind a control to a field in the Field List, the name of the control is changed to match the field name. You need to be concerned with a control's name when you refer to the control in an expression, so it is recommend you change the control's name to something unique to eliminate confusion.

Selecting Controls

The following describes the different methods for selecting controls on a form.

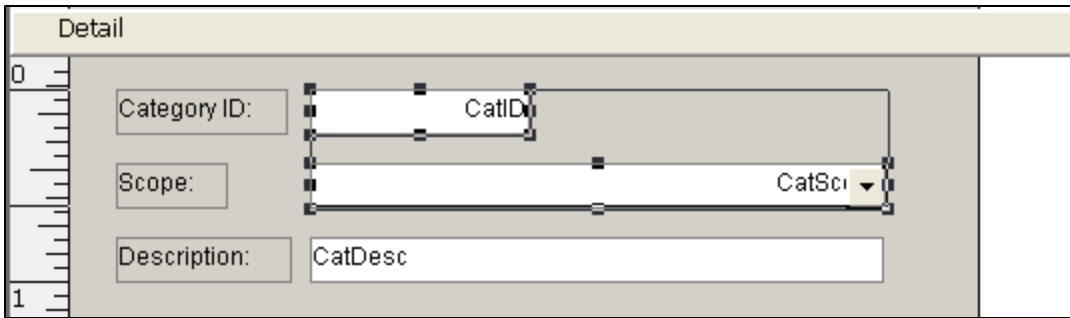
Selecting a Single Control

Click the control with the left mouse button. When a control is selected, it is surrounded by a highlighted rectangle with eight small boxes. The small boxes are drag handles, which are used to resize the control.



Selecting Multiple Controls

To select multiple controls: Hold down the left-mouse button and drag the mouse pointer over the controls -OR- hold down the *Ctrl* key on your keyboard and click each control. Multiple selected controls can be moved, copied or deleted as a group.



Selecting the Next or Previous Control

When a control is selected, use the *Tab* key on your keyboard to select the next control -OR- use the *Shift-Tab* key on your keyboard to select the previous control.

Moving Controls

Moving a Single Control

Click the control and drag it to the desired position while holding down the left mouse button.

Moving Multiple Controls

1. To select multiple controls: Hold down the left-mouse button and drag the mouse pointer over the controls.
2. Hold down the left mouse button over one of the selected controls and drag the controls to the desired position.

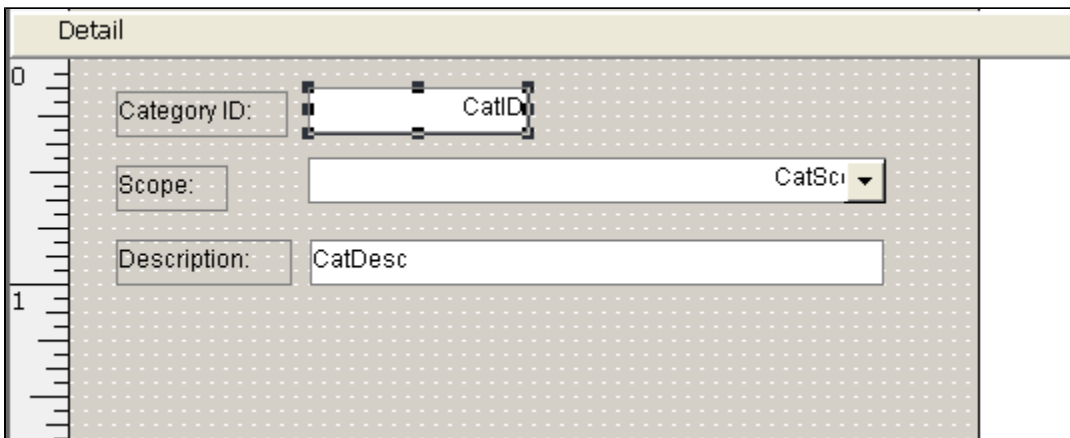
Aligning Controls

A group of controls can be quickly aligned to the grid or aligned to start at the same left, right, bottom or top position. For example, to align the left border of multiple controls:

1. Hold down the left-mouse button and drag the mouse pointer over the controls -OR- hold down the *Ctrl* key on your keyboard and click each control.
2. Select *Align-Left* from the menu.

Using the Grid to Position Controls

The form design window includes a grid of points used to align controls and set their size uniformly. The grid can be visible or invisible, but it is always there. The following sample screen shows you a form with a visible grid.



Hiding/Displaying the Grid

Select *Grid* from the *View* menu of the form designer. By default, the grid is invisible.

Configuring the Grid Points

The distance between the grid points is determined by the *Grid X* and *Grid Y* form attributes. These attribute values can be set to 1 thru 64 grid

points per inch. The default value is 16 grid points per inch, horizontally and vertically.

Enabling Snap-to-Grid

Select *Snap To Grid* from the *Layout* menu of the form designer. By default, snap-to-grid is turned on.

When snap-to-grid is turned on, the position of the control is adjusted to the nearest grid point when you add, move or size a control. This makes it easier to line up controls and size controls uniformly.

Sizing Controls

The following describes the different methods for sizing controls on a form.

Sizing a Control

1. Select the control. When a control is selected, it is surrounded by a highlighted rectangle with eight small boxes. The small boxes are drag handles.
2. Select one of the drag handles and drag it to resize the control. When you hover the mouse pointer over a drag handle, the pointer changes to a double-headed arrow to show you which directions you can size the control.

Enabling Size-to-Fit

Select *Size To Fit* from the *Layout* menu to match the size of selected controls with the size of the data in the controls.

Copying Controls

The following describes how to copy and paste existing controls when designing EDM forms.

1. Select the control to copy. When a control is selected, it is surrounded by a highlighted rectangle.
2. Copy it to the clipboard by pressing the *Ctrl+Insert* keys on your keyboard.
3. Select the section you want to paste the control into by clicking the section header bar.
4. Paste the control into the section by pressing the *Shift+Insert* keys on your keyboard.
5. Drag the pasted control from the upper-left corner of the section to the desired position.

Deleting Controls

The following describes how to delete existing controls when designing EDM forms.

1. Select the control. When a control is selected, it is surrounded by a highlighted rectangle with eight small boxes.
2. Press the *Delete* key on your keyboard. You will not be prompted to confirm the deletion.

Setting the Tab Order for Controls

The following describes how to set the tab order for controls when designing EDM forms.

The tab order of form controls is the sequence in which the cursor moves from control to control when the user presses the *Tab* key on the keyboard. The default tab order is the sequence in which the controls were added to the form.

1. Select *Tab Order* from the *Edit* menu of the form designer. Each control is listed with a box specifying its current tab order.
2. Click the tab order boxes in the desired sequence. If you make a mistake, click the controls a second time to remove them from the tab order, and then click them again in the proper sequence.
3. Select *Tab Order* from the *Edit* menu again to turn off the function.

Detail

0

Category ID: 1 ID

Scope: 2 CatSci

Description: 3

1

Advanced Controls

The pages in this section contain information about creating advanced controls when designing EDM forms.

Topics

Button Controls

Button controls are used to trigger actions in the system. For example, to call a function when the user clicks a button, add a button control to the form and set the *On push* attribute to the expression to evaluate.

Creating a Button Control

1. Click the button control tool



in the control toolbox.

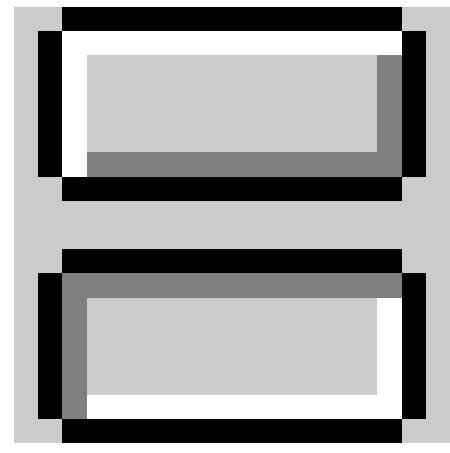
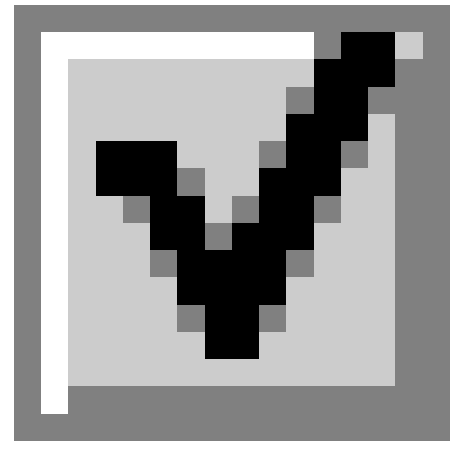
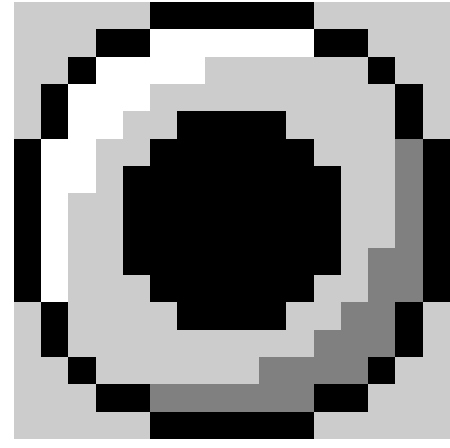
2. Click the form where you want the top-left corner of the button to appear.
3. Type the button label in the *Caption* attribute.
4. Type an equal sign (=) followed by the function call in the *On push* attribute. For example, to open the *Customer* form when the button is pressed, set its *On push* attribute to: `=Open('Form', 'Customer')`
To call more than one function, simply add them together like a mathematical formula as follows: `=Open('Form', 'Customer') + GotoControl('Last name')`

Yes_No Controls

Yes/No controls are used to enable the user to change a field value with a single mouse click or by pressing the space bar. Yes/No controls may be designed as toggle buttons, check boxes or radio buttons. For example, the *Married* field on an employee form could be designed as a toggle button that indicates the employee is married when the button is toggled on.

Married Employed Graduate

Male Senior Over 55

	<p>Toggle buttons</p>	<p>Toggle buttons are used to edit data that can be either True or False. When the user clicks a toggle button control, the button remains depressed and its value is 1. When a toggle button is not depressed, its value is 0.</p>
	<p>Check boxes</p>	<p>Check boxes are small square controls that are either empty or contain a check mark. When the user clicks a check box control, a check mark appears and the control's value is 1. If the user clicks the control again, the check mark disappears and the control's value is 0.</p>
	<p>Radio buttons</p>	<p>Radio buttons are small round controls that are either empty or filled. When the user clicks a radio button, the center of the button is filled and the control's value is 1. If the user clicks the control again, the center is cleared and the control's value is 0.</p>

Creating a Yes/No Control

1. Click the toggle button



, check box



or radio button tool



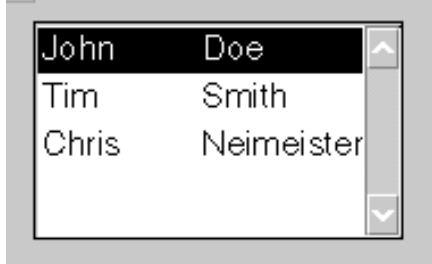
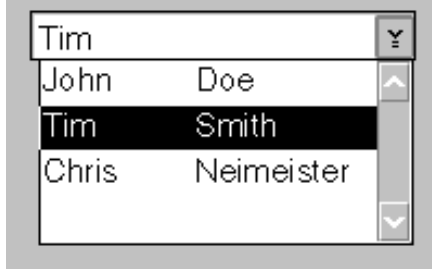
in the control toolbox.

2. Click the form where you want the top left corner of the control to appear.

Listboxes and Combo Box Controls

Listbox and combo box controls enable the user to select a value from a list. For example, an order form that includes a listbox of customer

names enabling the user to select the applicable customer for the order.

Listbox	Listboxes allow the user to scroll through a list of choices to make a selection. The list is always visible on the form even when the cursor is on another field.	
Combo Box	A combo box enables the user to either type a value or select a value from the list. If the user types a value that matches a value in the list, the system automatically highlights that value. For example, if the user types the letter 'S', then the first value in the list that starts with the letter 'S' is automatically highlighted. Pressing <i>Tab</i> on the keyboard selects the highlighted item. To configure the control to accept a value that is not included in the list, set the <i>Limit to list</i> attribute for the control to <i>No</i> .	

Creating a Listbox or Combo Box Control

1. Click the listbox



or combo box



tool in the control toolbox.

2. Click the form where you want the top left corner of the control to appear.
3. From the *Field List* window, double-click the name of the column that you want to bind to the control.

Defining the Rows to Include in the List

The data for the rows in the list can come from the following sources:

- Rows from a table, query or SQL statement
- Field names in a table or query
- User-defined list of values

You can select the columns in each row to include in the list, the width of each column and which column to use as the field value. For example, if your list source has three columns *ID*, *First Name* and *Last Name*, you can include the *First Name* and *Last Name* columns in the list and use the *ID* column as the field value to save in the database.

The following sample screen shows the attribute settings for a combo box that includes state codes.

Combo Box	
Name	StateCode
Source	StateCode
Row source type	Table/Query
Row source	select StateCode from tblStates or
Column count	1
Column widths	
Column headings	
Bound column	1
List rows	4
List width	
Limit to list	Yes
Format	
Length	255
Decimal places	0
Default value	
Required	No
Null item	
Status bar text	StateCode
Validation rule	
Validation text	
Begin update	
Before update	
After update	

Rows from a Table, Query or SQL Statement

1. Select *Table/Query* as the *Row source type* attribute.
2. Type the name of the table or query in the *Row source* attribute. To use a SQL statement, type the SELECT statement in the *Row source* attribute, e.g. *SELECT * from Customer*.

For example, to select a customer from the *Customer* table with the columns *ID*, *First Name* and *Last Name*, set the list attributes as follows:

Attribute	Value	Comment
Row source type	Table/Query	Source of the list rows is a table
Row source	Customer	Get list rows from the <i>Customer</i> table
Column count	3	Use the first three columns in each row
Column widths	0;1 5;5	Hide column 1, col2=1.5", col3=.5"
Bound column	1	Use column 1 as the value of the field

User-defined List of Values

1. Select *Value list* as the *Row source type* attribute.
2. Type the list values in the *Row source* attribute separating each value with a semicolon (;).

For example, to select from a list of region numbers and names, set the list attributes as follows:

Attribute	Value	Comment
Row source type	Value list	List rows are user-defined
Row source	1;East;2;Centreal;3;West	Rows of two-column list
Column count	2	Two columns in each row
Column widths	0;1.5	Hide column 1, column 2 is 1.5"
Bound column	1	Use column 1 as the value of the field

Field Names in a Table or Query

Set the list attributes as follows:

Attribute	Value	Comment
Row source type	Field list	List the columns in a table or query
Row source	Customer	Get the column list from the <i>Customer</i> table
Column count	1	One column in each row
Column widths		Use the default column width (the width of the list)
Bound column	1	Use column 1 as the value of the field

Determining the Bound Column

The *Bound column* of a list is the sequential number of the column to save to the database when the user selects a row from the listbox. The bound column does not have to be displayed in the listbox. For example, the *Row source* of your list is the *Customer* table, and the first three columns of the *Customer* table are: *ID*, *First Name* and *Last Name*. To save the *ID* column value for the user-selected customer, set the *Bound column* attribute to 1. To save the *LastName* column value, set the *Bound column* attribute to 2.

Setting the Width of List Columns

Specify the width of each column using the *Column widths* attribute separating each width value with a semicolon (;). In the following example, the first column will be set to .5 inches wide and the second column will be set to 1-inch wide.

Column widths	.5; 1
---------------	-------

To hide a column, set its width to 0.

Determining the Column Count

The *Column count* attribute is used to specify how many columns of the *Row source* attribute to include in each row of the listbox. For example, if the *Row source* is a table with 15 columns, then you can set the *Column count* from 1 to 15. Set the *Column count* to the highest column number that is used for the control, including hidden bound columns. For example, if columns 3 and 7 are displayed in the listbox and the hidden bound column is 10, then set the *Column count* to 10.

Sizing a Combo Box List

Use the *List rows* and the *List width* attributes to set the size of the control.

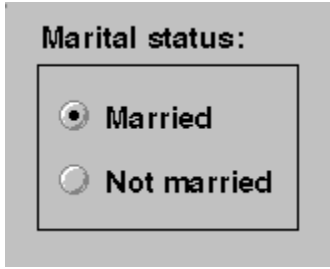
List rows	The number of rows to display when the list is opened; if this attribute value is blank, the default of 4 rows is used
List width	The width of the list in inches; if this attribute value is blank, the width if this attribute value is blank will match the width of the text box

Radio Button Groups

About Group Controls

To simplify data entry and reduce errors, you can provide specific options to the user using Group controls, which are groups of: radio buttons, check boxes or toggle buttons.

In the following example, an option group of radio buttons is provided to the user, where the user can select either *Married* or *Not married*, but not both.




The image shows a form titled "Marital status:" with a light gray background. Inside a black-bordered box, there are two radio buttons. The top one is labeled "Married" and has a black dot in the center, indicating it is selected. The bottom one is labeled "Not married" and has a white dot in the center, indicating it is not selected.

When creating an Group control, you typically bind the control to a field in the Field List while leaving the individual toggle buttons, check boxes and radio buttons unbound (*Source* attribute is blank). When the user selects one of the buttons in the group, the value of the Group control is set to the *Option value* attribute of the selected button.

In the *Marital status* example above, the Group control is bound to the *Married* field in the Field List. The *Option value* attribute of the *Married* radio button is set to 1 and the *Option value* attribute of the *Not married* button is set to 0. The *Source* attribute for each radio button is blank (unbound).

Creating a Group Control

1. Add one toggle button, check box, and/or radio button to the form for each option.
2. Click the Group control tool  in the control toolbox.
3. Click the form where you want the top-left corner of the group box to appear.
4. Drag the bottom-right corner of the group box until the group of buttons/check boxes is inside the box.
5. Set the *Option value* attribute of each button/check box in the group to the desired value when that button/check box is selected.
6. Set the *In group* attribute of each button/check box in the group to the name of the Group control.

The following sample screens show you the attribute settings for a group control and a radio button in the group.

Group Box	
Name	Married
Source	Married
Format	
Length	1
Decimal places	0
Default value	1
Required	Yes
Validation rule	
Validation text	
Begin update	
Before update	
After update	
On enter	
On exit	<input type="text"/>
Visible	Yes
Label control	
Tab control	General
Tab page	General
Left	117
Top	146
Width	100

Radio Button	
Name	Field29
Source	
Option value	
Default value	
Required	No
Status bar text	
Validation rule	
Validation text	
Begin update	
Before update	
After update	
On push	
On enter	
On exit	
Tab stop	Yes
Visible	Yes
Enabled	Yes
Label control	
Tab control	General

A Group control can also be used to simply draw a box around a group of related controls. In this case, the Group control would be unbound (*Source* attribute is blank) and the buttons in the group would be bound (*Source* attribute set to the fields each button updates).

Forms with Subforms


A subform is a form within a form enabling you to view information from more than one table. For example, you may want to edit both orders and the order items on the same form.

Order	Description	Item number	Price	Qty
1	Webb	1001	\$1.99	
1	Spiral	1003	\$1.99	
1	Scales	1004	\$1.99	
1	Flamingo	1006	\$2.99	

Date order was placed

Creating Subforms

Perform the following steps to create a form with a subform.

1. Create the subform first as a normal form adding all the fields to edit in the secondary table. You may want to set the *Default view* attribute of the subform to *Spreadsheet* to display the subform in spreadsheet view.
2. Create the main form with controls for all the fields to edit in the main table.
3. On the main form design, click the *Subform* control tool  in the control toolbox.
4. Click the form where you want the top-left corner of the subform to appear.
5. Drag the lower-right corner of the subform control to the desired size.
6. Set the *Subform name* attribute of the subform control to the name of the subform.
7. Use the *Form link fields* attribute to identify the fields on the main form that must match fields on the subform, separated by semicolons (;).
8. Use the *Subform link fields* attribute to identify the fields on the subform that must match the fields on the main form, separated by semicolons (;).

Editing Data on Forms with Subforms

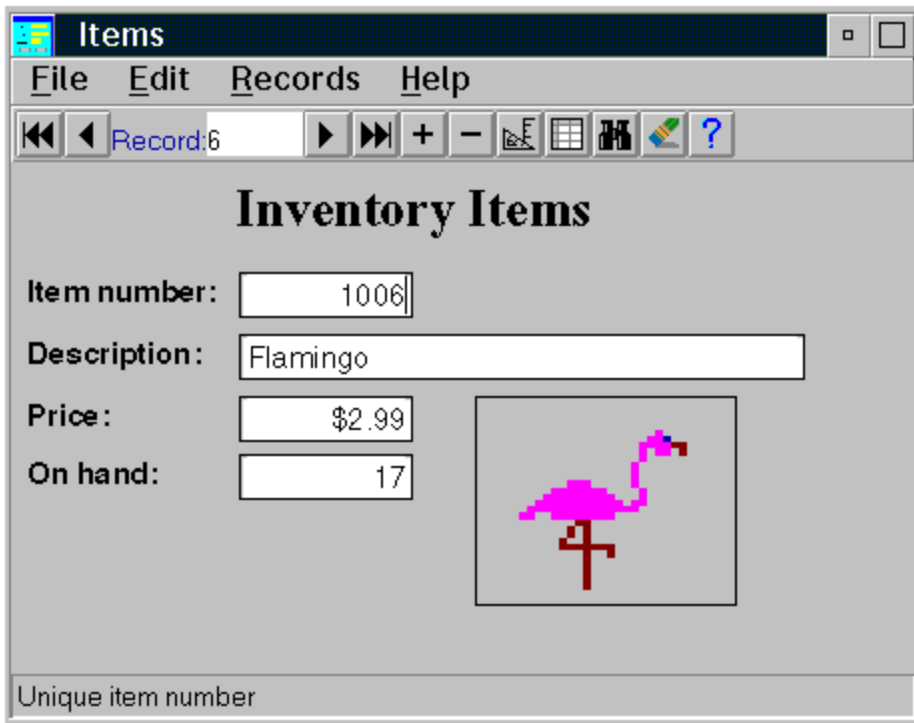
Opening a form with a subform is like opening two forms. The main form is typically used to edit one table, while the subform is used to edit a different table.

When you move to a different record on the main form, the subform is displayed again to show the related records. When you move to a different record on the subform, the main form continues to display the same data. You can add or change as many records as you want on the subform without affecting the record on the main form.

To switch between records on both the form and its subform, place the cursor in one of the fields and use the VCR controls.

Bitmap Controls


Bitmap controls enable the user to display, cut, copy and paste pictures, such as employee photos. A picture can be stored in a Bitmap field in a table or in a file on disk.




If the picture is stored in a bitmap field in a database table, you can bind a bitmap control to the field by selecting it from the Field List window. Pictures larger than 32K must be loaded from a disk file and the file name must be specified in a field in the database table.

Creating a Bitmap Control


The following steps describe how to create a bitmap control that displays a picture stored in a bitmap field in a database table.

1. Click the bitmap control tool  in the control toolbox.
2. Click the form where you want the top-left corner of the bitmap to appear.
3. Double-click the field in the Field List window that contains the picture.

The following steps describe how to create a bitmap control that displays a picture from a filename in a field.

1. Click the bitmap control tool  in the control toolbox.
2. Click the form where you want the top-left corner of the bitmap to appear.
3. Set the *Name* property of the bitmap to the desired name, such as *MyImage*.
4. Click the blank area between the rulers in the upper-left corner of the form designer to display the form properties.
5. Set the *After Current* property of the form to: `=setValue('MyImage', loadPicture(pictureFieldName)`

The following steps describe how to create a bitmap control that displays a picture from a disk file.

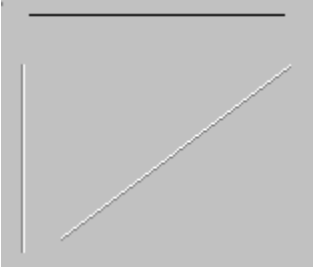
1. Click the bitmap control tool  in the control toolbox.
2. Click the form where you want the top-left corner of the bitmap to appear.
3. Set the *Name* property of the bitmap to the desired name, such as *MyImage*.
4. Click the blank area between the rulers in the upper-left corner of the form designer to display the form properties.
5. Set the *After Open* property of the form to: `=setValue('MyImage', loadPicture('image/logo.jpg')`

Inserting a Picture on a Form

1. Copy a picture from another application, e.g. an icon editor.
2. Paste it into the picture field on the form.

Line Controls

Line controls are used to draw straight lines on forms. The lines may be normal, sunken or raised.



Drawing a Line on a Form

1. Click the line control tool



in the control toolbox.

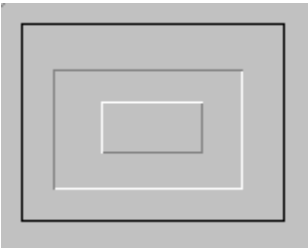
2. Click the form where you want the top-left corner of the line to appear.
3. Stretch the sizing box until the lower-left end of the line is in the desired position.

For very fine control over line size and position, you can set the *Left*, *Top*, *Width* and *Height* attributes to the desired values. For horizontal lines, set the *Height* attribute to 1. For vertical lines, set the *Width* attribute to 1.

By default, lines are drawn from the upper-left corner of the control rectangle to the lower-right corner. To create a diagonal line from the lower-left corner to the upper-right corner, size the control rectangle to the desired size and set the *Line slant* attribute to '/'.

Box Controls

Box controls are used to draw filled or outline boxes on forms. The boxes may be normal, sunken or raised.



Drawing a Box on a Form

1. Click the box control tool



in the control toolbox.

2. Click the form where you want the top-left corner of the box to appear.
3. Drag the lower-right corner of the box to the desired position.

Filling a Box with Color

1. Select the box control.
2. Set the fill color to the desired color on the Palette window.
3. Turn off the *Clear background* option on the Palette window.

Form Sections

This section describes the different sections of a form.

Headers and Footers

The Header and Footer sections of a form typically contain read-only text and/or function buttons. To access the Header or Footer on a form, the user can click one of the fields, press *F6* on the keyboard to go to the next section or press *Shift+F6* to go to the previous section.

The following is a form with Header and Footer sections in design view.

The diagram shows a form layout with three main sections: Form Header, Detail, and Form Footer. The Form Header section contains a 'Reasons:' label and a dropdown menu labeled 'Reason'. The Detail section contains several fields: 'StatusID:' with a dropdown menu labeled 'Statu', 'ReasonID:' with a text box containing 'ReasonID', 'Description:' with a text box containing 'Description', 'Replicated:' with a text box containing 'Replicated', 'RepGUID:' with a text box containing 'RepGUID', and 'ExternalCode:' with a text box containing 'ExternalCode'. The Form Footer section is empty.

The following is a form with Header and Footer sections in form view.

The screenshot shows a software application window with a menu bar (File, Edit, Records, Help) and a toolbar. The toolbar includes icons for grid, save, back, forward, search, and help. The form displays the following data:

Reasons:	1, Personal problems
StatusID:	Terminated
ReasonID:	1
Description:	Refused to work
Replicated:	0
RepGUID:	16AFC91D-73A4-4778-87B0-BE285BD57015
ExternalCode:	<NULL>

Adding/Removing the Header and Footer from a Form

Select *Form Hdr/Ftr* from the *Layout* menu of the form designer.

Adding/Removing the Header and Footer from a Page

Select *Page Hdr/Ftr* from the *Layout* menu of the form designer.

Hiding the Header or Footer

Drag the bottom boundary to the top of the section.

Changing the Height of a Section

1. Hover the mouse pointer over the bottom of the section until it changes to a double-headed vertical arrow.
2. While holding down the left mouse button, drag the section boundary to the desired position.

You can also specify the height using the *Height* attribute of the section.

Changing the Width of a Section

1. Hover the mouse pointer over the right edge of any section until it changes to a double-headed horizontal arrow.
2. While holding down the left mouse button, drag the section boundary to the desired position.

You can also specify the width using the *Section width* attribute of the form. Provide the width in hundredths of inches, e.g. for 4.5 inches, type 450.

Form Attributes

This section describes how to define the various attributes of a form.

Opening the Form Attribute Window

1. Click the *Attributes* tool



on the tool bar.

2. Select the form by clicking the gray block in the upper-left corner of the form (where the horizontal and vertical rulers meet).

Setting the Data Source for a Form

Forms normally display data from a database table or query. When changes are made to the form data, the data is changed in the database table. For example, if the data source for a form is the *Customer* table, then when the user makes changes to the form data, the changes are saved in the *Customer* table.

The data source for a form is specified using the *Source* attribute. The *Field List* of the Form Designer lists all the fields in the specified database table or query.

1. Select the *Source* attribute from the form attribute window.
2. Click the list button at the right end of the field.
3. Select the table or query to use as the data source (tables are marked with a +).

Setting the Form Title

Use the *Title* attribute to specify the form title, which appears in the title bar of the form.

Setting the Form Size and Position

Drag the title bar of the form to the desired position. Drag the sizing border to the desired size -OR- select *Size to Fit* from the *Layout* menu.

You can also use the following attributes to specify the size and position of the form. Specify each of the following values in hundredths of inches, e.g. 150 equals 1.5 inches.

Attribute	Use it to specify
Top	The desired distance from the top of the form to the top of the screen
Left	The desired distance from the left edge of the form to the left edge of the screen
Width	The width of the form
Height	The height of the form

Displaying a Form as a Spreadsheet

Use the *Default view* attribute to set the default view for forms: *Form* or *Spreadsheet*

In *Spreadsheet* view, text, multi-line text, toggle buttons, check boxes and radio buttons are presented as text columns. List boxes and combo boxes are presented as combo box columns. Each column name is set to the name of the respective control.

Showing or Hiding the Menu Bar

To show the menu bar, set the *Menu* attribute to *Yes*. The menu bar is located at the top of the form.

To hide the menu bar, set the *Menu* attribute to *No*. When the menu bar is hidden, the user can access the menu options by:

- Selecting the system menu button from the top-left corner of the form
- Right-clicking anywhere on the form to open the popup menu

Hiding Buttons when Using the Form Viewer

By default, each of these buttons appear on the selection screen when using the form viewer. If inserts or deletes should not be allowed, a modification to the form attribute is required.

To hide the *New* button, set the *Allow insert* attribute to =false

To hide the *Copy* button, set the *Allow copy* attribute to =false

To hide the *Delete* button, set the *Allow delete* attribute to =false

Showing or Hiding the VCR Controls

To show the VCR controls, set the *VCR controls* attribute to *Yes*. The VCR controls are located on the toolbar of a form.

To hide the VCR controls, set the *VCR controls* attribute to *No*.

Showing or Hiding the Toolbar

To show the toolbar, set the *Toolbar* attribute to *Yes*. The toolbar is located at the top of the form.

To hide the toolbar, set the *Toolbar* attribute to *No*.

Showing or Hiding the Scroll bars

If a form is bigger than the window, the scroll bars should be displayed to enable the user to view the parts of the form that are not visible. The following describes the options available for the *Scroll bars* attribute.

Option	Select it to
--------	--------------

Neither	Hide both scroll bars
Horizontal only	Display the horizontal scroll bar
Vertical only	Display the vertical scroll bar
Both	Display both scroll bars

Showing or Hiding the Status Bar

To show the status bar, set the *Status bar* attribute to *Yes*. The status bar is located at the bottom of the form.

To hide the status bar, set the *Status bar* attribute to *No*.

The status bar displays:

- The *Status bar text* attribute value for the currently selected control on the form
- The description of the tool when the user hovers the mouse pointer over one of the toolbar tools

Removing the Title Bar

To show the title bar, set the *Title bar* attribute to *Yes*. The title bar is located at the top of the form.

To hide the title bar, set the *Title bar* attribute to *No*.

Allowing the Form to Be Resized

To allow the user to resize the form, set the *Sizing border* attribute to *Yes*. The sizing border is displayed around the form window.

To NOT allow the user to resize the form, set the *Sizing border* attribute to *No*.

See Also

[Events](#)

Events

This section describes the various form events.

Form Events

Event	Is generated
Begin insert	When the user types the first character of a new record
Before insert	When the user tries to save a new record, but before the record is saved
Cancel insert	When the user selects <i>Undo</i> to cancel an insert
Begin update	When the user types a character into an existing record
Before update	When the user tries to save an updated record, but before the record is saved
After update	After the user updates a record
Cancel update	When the user selects <i>Undo</i> to cancel an update
Before delete	When the user tries to delete a record, but before the record is deleted
After delete	After the user deletes a record
Cancel delete	When a user cancels a delete operation
Before current	When a user requests to move to a new record, but before the record is read

After current	After the user moves to a new record
Before open	Before the form is opened
After open	After a form is opened, but before any data is displayed
Before close	Before a form is closed
After close	After a form is closed

Responding to Form Events

To evaluate expressions when an event occurs, type the equal sign (=) followed by the expression on the form attribute for the event. For example, to display a message when the form is closed, type `=Msg('Form is closing!','Note',0)` in the *On Close* attribute of the form. To call a user-defined function called `CheckData()` before a new record is inserted, type `=CheckData()` in the *Before insert* attribute of the form. You can call the `CancelEvent()` function to cancel some events. For example, if you find a problem with the data before inserting a record, call `CancelEvent()` to not save the record. `CancelEvent()` is not compatible with the following events:

- After insert
- Cancel insert
- After update
- Cancel update
- After delete
- Cancel delete
- After current
- After open
- After close

Control Events

Event	Is generated
Begin update	When the user first types or presses Ctrl-Enter on a control
Before update	After the user changes a control value, but before the value is validated
After Update	After the user changes a control value and the value is validated
On Push	When the user clicks a button, a toggle button, a radio button or a checkbox
On Enter	When the user moves to the control
On Exit	When the user moves to a different control
On Dbl Click	When the user double-clicks a control

Responding to Control Events

To evaluate expressions when an event occurs, type the equal sign (=) followed by the expression on the control attribute for the event. For example, to call a user-defined function called `CheckData()` before a new record is inserted, type `=CheckData()` in the *Before update* attribute of the control. You can call the `CancelEvent()` function to cancel the *Before update* event. For example, if you find a problem with the data before updating a field, call `CancelEvent()` to not save the field data.

Opening Forms

This section describes opening a form.

Opening a Form

1. Select the *Setup* module.
2. Expand the *Application* menu.
3. Expand the *Forms* menu.
4. Select the form to open.

Opening a Form in the Designer

1. Select the *Setup* module.
2. Expand the *Application* menu.
3. Expand the *Forms* menu.
4. Right-click the form, and then select *Design* from the menu that appears.

Form View

In *Form view*, the detail section and the header and footer of the form are displayed and formatted with the data from the first record of the data source. To move between records on the form, click the VCR buttons at the top of the form window.

The screenshot shows a software window titled "File Edit Records Help". The window contains a toolbar with icons for grid, home, left arrow, right arrow, record number (1), double right arrow, plus, minus, refresh, search, up arrow, down arrow, and help. Below the toolbar are two dropdown menus: "Select site by name:" with the value "Price Tier 2 - \$7.29 Lunch, -2" and "Select site by number:" with the value "-2, Price Tier 2 - \$7.29 Lunch". The main area of the form displays the following data:

Location ID:	-2
Name:	Price Tier 2 - \$7.29 Lunch
Transaction version:	7
Custom ID:	<NULL>
Transfer host:	<NULL>
Transfer user id:	<NULL>
Transfer password:	<NULL>
Transfer company:	<NULL>
View Transaction Status	<input checked="" type="checkbox"/>
Location Notes	<NULL>

Spreadsheet View

In *Spreadsheet view*, the records from the data source are displayed on a spreadsheet with one record on each row and one field in each column.

File Edit Records Help

Record: 1 Search:

SITEID	NAME	VERSION	CUSTOMID	HOSTNAME	USERID
* 2	Price Tier 2 - \$7.29 LU	7	<NULL>	<NULL>	<NULL>
-1	Price Tier 1 - \$6.99 LU	-1	<NULL>	<NULL>	<NULL>
0	Ryans	9			
2101	2101 - Greenville #1,	7	<NULL>	<NULL>	<NULL>
2103	2103 - Anderson #2, S	7	<NULL>	<NULL>	<NULL>
2105	2105 - Columbia #1, S	7	<NULL>	<NULL>	<NULL>
2106	2106 - Greenwood, SC	7	<NULL>	<NULL>	<NULL>
2107	2107 - Spartanburg, S	7	<NULL>	<NULL>	<NULL>
2112	2112 - Augusta #1, GA	7	<NULL>	<NULL>	<NULL>
2115	2115 - Columbus, GA	7	<NULL>	<NULL>	<NULL>
2118	2118 - Rome, GA	7	<NULL>	<NULL>	<NULL>
2119	2119 - Hiram, GA	7	<NULL>	<NULL>	<NULL>
2120	2120 - Albany, NY	7			
2122	2122 - Chattanooga, T	7	<NULL>	<NULL>	<NULL>
2126	2126 - Greer, SC	7	<NULL>	<NULL>	<NULL>
2128	2128 - Corydon, IN	7	<NULL>	<NULL>	<NULL>

Unique site id

For instructions on using a spreadsheet to edit data, see [Editing Data](#).

Selecting Form or Spreadsheet View

Click the *Spreadsheet* tool



on the toolbar to view the form in *Spreadsheet* view.

Click the *Form* tool



on the toolbar to view the form in *Form* view.

Navigating Forms

You can move to a control on a form by clicking it. You can move around the form using the keys listed in the following table.

Key	Moves the cursor to
Tab	The next column
Shift+Tab	The previous column
F6	The first control in the next section
Shift-F6	The last control in the previous section
Ctrl+Page Up	The previous record
Ctrl+Page Down	The next record
Ctrl+Home	The first record
Ctrl+End	The last record

Record Selectors

Using Joins

The CDMRecordSelector is used in the form viewer to create the list of records to be selected.

For example, if the table on which the form is based does not have a name field, the CDMRecordSelector can be used to join to the appropriate table that has the names. In this example, the *TNG_dbo_Products* table does not have a name field, but the *TNG_dbo_NLSProducts* table does. A combo-box with the Row Source (shown below) provides the ability to list the English names and numbers of the products.

```
SELECT TNG_dbo_Products.productId, TNG_dbo_NLSProducts.longDescription
FROM TNG_dbo_Products, TNG_dbo_NLSProducts,
TNG_dbo_Products INNER JOIN TNG_dbo_NLSProducts
ON TNG_dbo_NLSProducts.[productId]=TNG_dbo_Products.[productId]
WHERE TNG_dbo_NLSProducts.[locale]='en_US'
and TNG_dbo_Products.[productType] = 1
and TNG_dbo_Products.[productId]>0
```

Column headings will appear on the selection screen respecting the column widths.

Combo Box	
Name	CDMRecordSelector
Source	
Row source type	Table/Query
Row source	SELECT TNG_dbo_Products.prod
Column count	2
Column widths	.4;2
Column headings	ID;Item Name
Bound column	1
List rows	20
List width	

Form Viewer

Overview

The 'view' function is a method used to open forms. Prior to having the 'view' function, the 'open' function was used. The 'view' function, or form viewer, is designed to allow the user to clearly understand when they are editing records that have differences. Prior to having the form viewer, when a form was opened including records with differing values, the user would not be aware of the differences unless the field was selected and the values pop up was viewed. Fields viewed on a form having differences might also enable and allow changes to fields for all selected sites, regardless of the fact that their values should not allow these changes.

To identify how a form is being accessed, right-click on the menu option and select 'Edit'. In the 'Actions' field, either Open or View will appear at the start of the line used to access a form. For example, view('form', 'MyForm').

Note

The Viewer is not used for opening functions like Menu Button Editors, which are custom coded and do not use Open or View when they are accessed. These editors are often extremely complex, accessing data from multiple tables that are intricately

related.

Differences between View and Open

View	Open
Prompts the user to select a record to edit.	Displays the first record for editing.
Provides a powerful filter tool enabling users to filter the list of records to find the ones they want to edit.	Uses only 'find by' (name or number) dropdown lists for selecting records
Can show all differences between groups of sites that have different versions of the data	Only shows data differences if the user selects a field to check.
Identifies groups have sites that have identical data and prompts the user to edit them separately from groups of sites that have different data values	Requires the user to press Ctrl-Enter or click the update button on each field to see data differences.
Enables users to edit data without the Ctrl-Enter or update button required by open because all the sites with the selected version of the data have the same value in every field	Requires the user to press Ctrl-Enter or click the update button on each field to see data differences.

Allows users to click Ok to save their changes or Cancel to cancel all changes including changes made to sub-forms	Changes data live, as each field is modified. This creates multiple update queries in the audit table.
Does not display a toolbar or menu at the top of the form but the user can access advanced options by clicking the tool button in the lower right corner.	Displays a toolbar at the top of the form where the user can choose actions.
Does not allow users to move directly to the next record like open does. Users click Ok or Cancel and then choose the next record they want to see.	Allows the user to move to the next record on the

See Also

[Using the Form Viewer](#)

Using the Form Viewer

Using the form viewer, when sites and a package are selected, a list of selectable values appears. The example below shows 'Number' and 'Name' as examples. Buttons on the bottom of the screen allow the user to edit, insert new, delete or copy records. More information about setting selection criteria and available buttons is available in the section called 'Form Edits for View Compatibility'. Button use is described in the section called 'Record Actions'.

Discounts Search:

Number	Name
1	FOH EMP
2	xxxKITCHEN
3	MGR/MIT
4	\$5 MGR CRD
5	CS FREQ CD
6	OTHER FOOD
7	OTHER LIQ
8	TRADE CERT
9	10% DISC
10	30% DISC
11	15% DISC
12	TRAIN MEAL
13	50% DISC
14	REP REWARD
15	\$10BONUSGC
16	TRADE QUIC

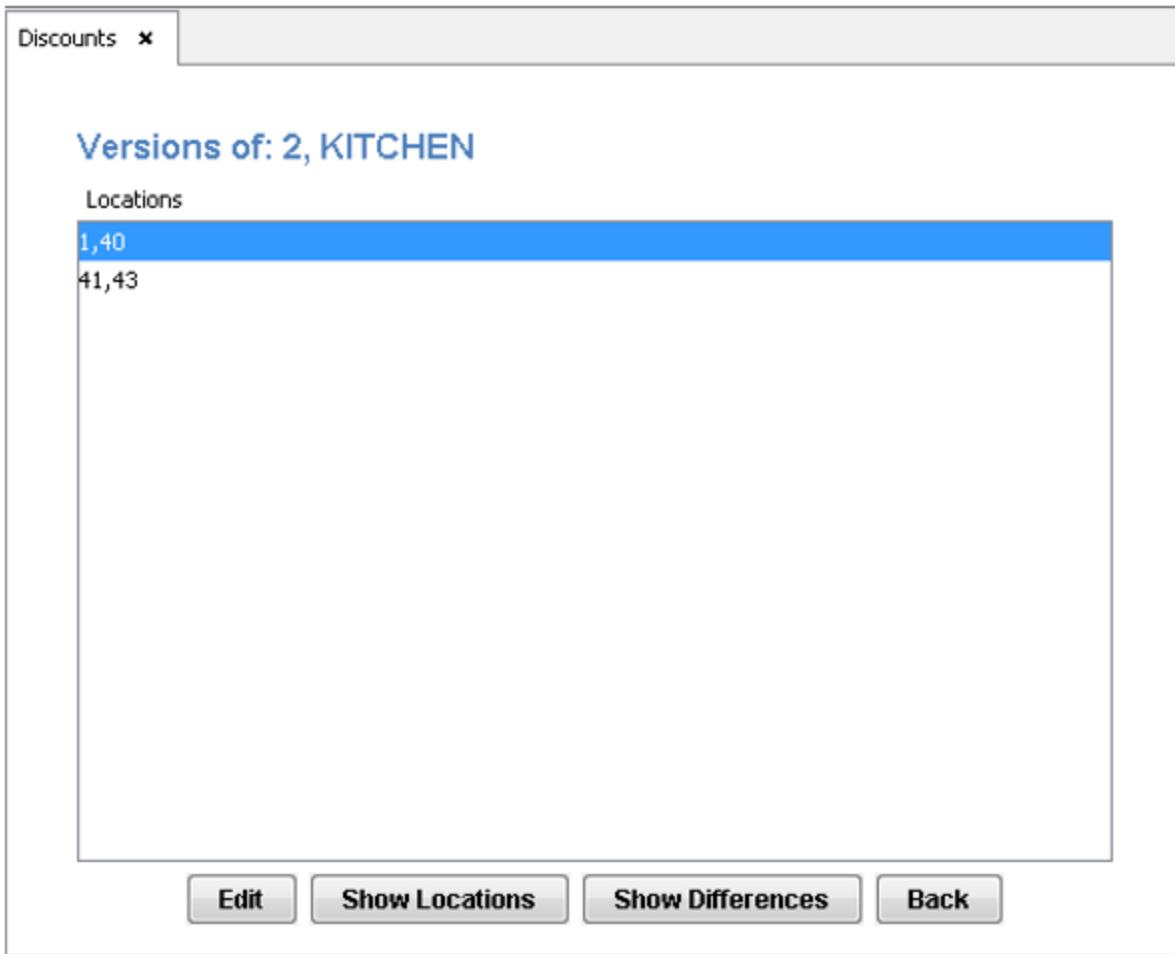
Edit New Delete Show Locations Copy Record
Copy to Sites Close

See Also

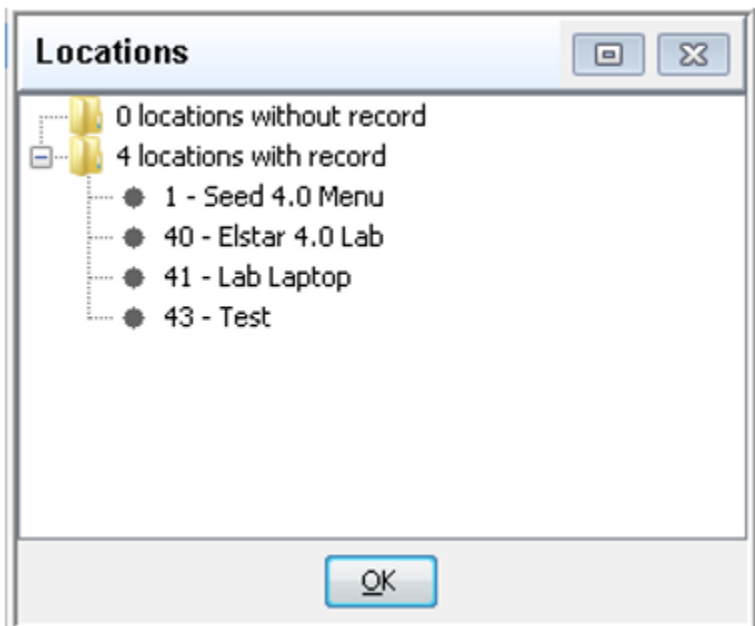
[Versioning](#)
[Editing Data](#)

Versioning - Form Viewer

When the desired record is selected, if the main-form records and sub-form records are not identical at all selected sites, a 'version' screen appears, which divides the selected sites into groups that have identical field values.



The 'show locations' button lists all sites that have the selected version, and a separate list of sites that don't have the record at all.



The lists will only contain sites that were selected before opening the form. The user can select any two groups, using <ctrl> click and choose 'show differences' to see what differs between the groups (fig. fv4).

Differences

ELSTAR_dbo_Discount differences between 1 - Seed 4.0 Menu and 41 - Lab Laptop

<u>iDiscountID</u>	<u>Column</u>	<u>Record Group 41</u>	<u>Record Group 1</u>
2	sName	xxxKITCHEN	KITCHEN

The groups appear in the differences screen having the name 'group' followed by the lowest site number in the group. The user may only select one site group at a time in order to access the form and edit values.

Note

When opening custom functions like Menu Button Editors, the Versioning that appears is not the standard functionality. It is custom coded because of the multiple, intricately related tables and functions required to create what is displayed. It may not behave exactly the same as the standard Viewer Versioning.

Editing Data - Form Viewer

When the record or site group is selected for editing, the user can edit data in any enabled field by simply typing in the field or, in the case of dropdown menus, selecting a value from the list. No additional key codes are required to edit records in the viewer.

Edit All Using Form Override

If the user would like to edit ALL sites that were selected, regardless of if they have differences, the <ctrl><enter> key combination can be used to update records in all selected sites. An example of this might be when the Name of an item must be modified, though the item has differing values in other fields. The different values for the selected field will appear in the popup.

The dialog box titled "Location Values" contains a list with two entries: "KITCHEN" and "xxxKITCHEN". The "KITCHEN" entry is selected. Below the list are three buttons: "Set to NULL", "Locations", and "OK".

Clicking the Locations button will display the locations having the selected value.

Locations

1 - Seed 4.0 Menu
40 - Elstar 4.0 Lab

(2) sites

OK

To change all values, each displayed line, which may represent one or more site records, should be changed. Please note that use of the <ctrl><enter> key combination circumvents the Viewer's purpose and is only intended to allow the user to make broad changes across differing sites, with the knowledge that they might be overriding form restrictions.

For example, a business rule built into the form might only allow the 'print signature line' field to be changed if the 'payment type' value is in a specified range. The purpose is to prevent a user from adding a signature line to a cash payment type. In this case, if the user selected <ctrl><enter> on the 'print signature line' field, they could edit the field without regard for the value in the payment type field.

Record Actions

When using the form viewer, upon accessing a form and before selecting a particular record to edit, the user sees the Record Selection screen.

Discounts Search:

Number	Name
1	FOH EMP
2	xxxKITCHEN
3	MGR/MIT
4	\$5 MGR CRD
5	CS FREQ CD
6	OTHER FOOD
7	OTHER LIQ
8	TRADE CERT
9	10% DISC
10	30% DISC
11	15% DISC
12	TRAIN MEAL
13	50% DISC
14	REP REWARD
15	\$10BONUSGC
16	TRADE CERT

This screen will contain action buttons and functions that will differ based on certain criteria.

Button/Function	Description
Sort	The columns on the screen are sortable by clicking the column header. The order alternates between ascending and descending each time the header is clicked. The initial sort order is based on the form selection criteria.
Edit	Opens the selected form for editing (or the versioning screen, as required.)
New	Opens a blank form where the user can enter data that will be added to all selected sites that do not already have the record.
Show Locations	Lists all sites in the package, listed by whether the site has the selected record.
Delete	Deletes the selected record from all selected sites. Pops an 'are you sure' message before deleting.
Copy Record	Copies the selected record to all selected sites. If necessary, prompts the user to enter key field data.
Copy to Sites	Only appears if more than one site is selected. Copies the selected record to any sites that do not already have the record.

Form Edits for Viewer Compatibility

The Viewer uses the CDMRecordSelector combo-box to populate the sortable, searchable columns of the Record Selector screen. And most buttons that appear on the Record Selector screen appear based on form settings. Main forms can be modified from the menu by selecting 'Design Item' if the user has permissions to edit forms. Subforms must be edited from Setup>>Applications>>Forms.

Form Name

On existing forms, the form Title is often the name of the table that the form is based on. The Title can be changed to be more user friendly.

Search Columns

Most forms that are currently accessed using 'Open' in the menu action already use a CDMRecordSelector to provide a 'Find by Name' or 'Find by

Number' selection list. The following field changes can make the Viewer more user friendly.

Combo Box	
Name	CDMRecordSelector
Source	
Row source type	Table/Query
Row source	select IDField,NameField from TableName
Column count	2
Column widths	.5;2
Column headings	Number; Name
Bound column	1

Row Source	The fields in the query of this combo-box object define the data that appears on the screen. The first field will be the sort field. Even if 'order by' is defined in the query, the 'order by' field should also be selected first.
Column Count	Must be the count of the fields being selected in the row source query.
Column widths	Provide widths, in inches, for each column, separated by a semicolon. Use decimals to define fractions of an inch and 0 to hide a column.
Column headings	If desired, provide user-friendly column names separated by semi-colons. If left blank, the field names from the query will be used.
Bound column	Identify which field, in sequence of the row source query, is the key field on the database table.

Buttons

Buttons appear on the Record Selection screen based, primarily, on certain form values. They are enabled unless otherwise defined.

Allow Copy	if '=false', removes the 'Copy Record' button.
Allow Insert	if '=false', removes the 'New' button.
Allow Delete	if '=false', removes the 'Delete' button.
Allow Edit	if '=false', removes that 'Edit' button. Users can still double-click the record and view it, but may not edit any fields.

Expressions

Expressions are used to calculate a value in a query column or a form control. Expressions can be combined using the *AND* and *OR* operators.

Expressions are defined using the following syntax: *Operand Operator Operand [AND/OR Expression]*

Examples

To create a query column called *NewSalary* with the value *Salary*1.2*: type *NewSalary: Salary*1.2*

To display an employee salary multiplied by 1.2 in a form control: type *=Salary*1.2* in the *Source* attribute of the control.

Topics

- [Condition Expressions](#)
- [Constants](#)
- [Referring to Object Values](#)
- [Expression Operators](#)
- [Expression Examples](#)
- [Functions](#)

Constants

You can use the following types of constants in [expressions](#).

String constants	'Smith', "Jones"
Numeric constants	7, 1.5
Date constants	'1/24/95' or '1/24/95 13:45'
Binary constants	'1FD59A' (hex characters 0..9 and A..F)

Referring to Object Values

The pages in this section contain information about referring to object values in [Expressions](#).

Topics

[Referring to Query Columns](#)

[Referring to Form Values](#)

[Object References](#)

Referring to Query Columns

[Expressions](#) in a query can refer to column values from the same query. Expressions in a form can refer to column values in a query that is used as the data source for the form.

Using a Query Column Value in an Expression

Type the name of the column -OR- the name of the table and the name of the column separated by a period without spaces, i.e. *TableName.ColumnName*.

If the table or column name contains spaces, enclose the name in brackets, e.g. *[Last name]*.

Examples

- ID
- [Account number]
- Customer.ID
- [Line Item].[Qty ordered]
- Salary * 1.2
- > Birthdate

Referring to Form Values

[Expressions](#) can refer to control values on forms. If both the expression and the control value are on the same form, simply type the name of the control in the expression.

For example, if you want the *NewSalary* control to display the value of the *Salary* control on the same form multiplied by 1.2: type `=Salary*1.2` in the *Source* attribute of the *NewSalary* control.

Use object references to refer to control values on other forms and queries.

For example, if you want to query a value in the *ID* field on the *Customer* form: type `=Forms:Customer:ID` in the *Condition* row of the *ID* column of the query.

When the query is run, `Forms:Customer:ID` is replaced with the value of the *ID* field on the *Customer* form. To refer to the *ID* field on the active form, type `=ActiveForm:ID` on the *Condition* row.

Object References

You can refer to objects in [Expressions](#) using the following format: *Object Name:Object Member*

Available Objects

ActiveForm	Refers to the currently active form
Forms	Refers to a list of open forms with a member for each form

Refer to an object that belongs to another object by separating the object names with a colon, e.g. *ActiveForm:Customer* refers to the open form named *Customer*.

Refer to an attribute of an object by entering the object followed by a period and the attribute name, e.g. *ActiveForm.Title*.

Object Reference Examples

ActiveForm.Title	Refers to the <i>Title</i> attribute of the currently active form
ActiveForm:[Order date]	Refers to the value of the <i>[Order date]</i> column on the currently active form
ActiveForm:[Order date].[Default value]	Refers to the <i>[Default value]</i> attribute of the <i>[Order date]</i> control on the currently active form
Forms:Customer.Title	Refers to the title of the open <i>Customer</i> form
Forms:Customer:[Account number]	Refers to the value of the <i>[Account number]</i> control on the open <i>Customer</i> form
Forms:Customer:ID.[Default value]	Refers to the <i>[Default value]</i> attribute of the <i>ID</i> control on the open <i>Customer</i> form

Expression Operators

The following table describes the operators that you can use in [Expressions](#).

Operator	What it does
+	Adds values
-	Subtracts values
*	Multiplies values
/	Divides values
%	Divides values and returns the remainder
&	Concatenates two strings
=	Verifies if the values are equal
<>	Verifies if the values are not equal
>	Verifies if the first value is greater than second value
>=	Verifies if the first value is greater than or equal to second value
<	Verifies if the first value is less than second value
<=	Verifies if the first value is less than or equal to second value
IS NULL	Verifies if the value is NULL
IS NOT NULL	Verifies if the value is not NULL
BETWEEN expr1 AND expr2	Validates that the value is between <i>expr1</i> and <i>expr2</i>
NOT BETWEEN expr1 AND expr2	Validates that the value is outside the range of <i>expr1</i> to <i>expr2</i>
IN (expr1, expr2, ... exprN)	Validates that the value is in the list of values
NOT IN (expr1, expr2, ... exprN)	Validates that the value is not in the list of values
LIKE 'string'	Verifies that the value matches the string (use % to match any number of characters and use _ to match any single character)
NOT LIKE 'string'	Verifies that the value does not match the string (use % to match any number of characters and use _ to match any single character)

Expression Examples

The following is a list of examples of [Expressions](#).

- Salary + 500
- [Old price] * 1.1
- [First name] & " " & [Last name]

- "Page" & 52
- Name = "Smith"
- [Old price] <> [New price]
- [Middle initial] IS NULL
- Salary BETWEEN 20000 AND 30000
- Price IN (1.99, 2.99, 3.99)
- [Last name] LIKE "SM%" (matches *Smith* and *Smattering*, but not *Snow*)
- [Last name] LIKE "SM____" (matches *Smith* and *Smurf*, but not *Smalls*)

Functions

Functions can be called in any [expression](#). For example, a query column defined as *DateValue: DateToNumber(hiredate)* would create a column in the query called *DateValue* that contains the Julian value of the *hiredate* column in each row.

You could also set the conditions of the *hiredate* column to *<Date('GD')-45*, which would only select rows where the hire date is more than 45 days ago.

See [Client Functions](#) for more detailed information about available functions and their parameters.

Client Functions

This section describes the available client functions in EDM.

Calling Functions in Expressions

Type the function name followed by its parameters. Enclose the parameters in parentheses ().

Calling a Function on a Control or Form Event

Precede the function call with an equal = sign. For example, to close the active window when the user clicks a button, type *=CloseActiveWindow()* in the *On push* attribute.

To call more than one function, connect the function names with operators like the plus + sign. For example, to close the active window and open the Customer form when the user clicks a button, type *=CloseActiveWindow() + Open('form', 'Customer')* in the *On push* attribute.

Function Names

- Function names must be followed by parentheses even when there are no parameters.
- Function names are not case sensitive.

Function Parameters

- String parameters must be enclosed in either single or double quotes.
- Numeric parameters should NOT be enclosed in quotes.
- True/False parameters may be entered as true or false without quotes, or as 1 or 0 (1=true 0=false).

Alphabetic Function Index

The following table lists all the functions in alphabetical order.

Function	What it does
abs	Get the absolute value of the given number. If the number is null, a <i>DataNumber</i> with the value null is returned. If the number is blank, 0 is returned.
addDataSource	Add a new data source to the system configuration.
addMultipleRows	Add multiple rows to an open form based on a list or combo box
addRow	Insert a row in a table for a given list of sites that do not already have the row. For example, when entering a
addSystemEventListener	Add the listener in the given class to the list of system event listeners.
addThisSiteToCentral	Call the central web service to add this remote site to the central database.

applyEffectiveMasterDataTrans	Queries the audit table for committed transactions on master data tables whose effective date has arrived and calls the appropriate master data change listener for each transaction then changes the status of the transaction to Successful.
ascii	Converts the first character of a string to an ASCII value
asciiToChar	Converts an ASCII code to a character
attachMasterTables	Attach tables from an existing database so it can be edited
attachRemoteTable	Attach a table from an existing database so it can be edited
attachTable	Attach tables from an existing database so it can be edited
callEvaluateExpressionWebMethod	Call the EvaluateExpression web method.
cancelEvent	Cancel current event on the active form.
changePackageStatus	Change status of the transactions for sites in a package to a new status.
changePassword	Change the current user's password.
clearSessionPackage	Remove the selected session package on the client to users have to select a package when making changes.
close	Closes the specified open object window (form view, design view, or print preview).
closeActiveWindow	Closes the active window.
commitTransactions	Commit packages of transactions so they are sent to selected locations
configureApplicationVersions	Configure known versions of application data sources.
copy	Copies an object to the clipboard.
copyAndRefreshSiteData	Copy data for selected tables from one site to other sites and refresh the tables at the other sites on the effective date.
copyCompany	Create a new company by copying the configuration of an existing company.
copyConfigurationVersionData	Copy the data from one configuration version to another version of the same configuration overwriting any existing data.
copyFoodCostPeriodData	Copy food cost data from one period to another.
copyLocationData	Copy site data for a list of tables to other sites. DEPRECATED Use copySiteData instead.
copySelectedObject	Copies the application object selected on the main menu to the clipboard.
copySiteData	Copy data for selected tables from one location to another location
copySiteSubscriptions	Copy the configuration version subscriptions from one site to other sites leaving the version ID null. If the source site has no subscriptions or the list of configuration ID's to copy is empty, do nothing.
copyVersionData	Copy the data from one version to another version for all configurations overwriting any existing data.
createConfigurationVersionDataSite	Creates a data site to hold version data for configuration version. The ID of the new site will be an unused, negative site number. The name of the site will be the name of the configuration plus a hyphen plus the name of the version. Each table in the configuration type will be added to the list of managed site tables. If there is a previous version of the configuration then the data from the previous version will be copied to the new version.
createConfigurationVersionsCmd	Create configuration versions for the given configuration type and configuration.
createRemoteInstaller	Create an installation program that can be downloaded and used to install EDM at a remote site with pre-configured settings for the current company.
createTransactionTriggers	Create SQL scripts so triggers on application tables at remote sites can generate EDM transactions.
cut	Cuts an object to the clipboard.

cutSelectedObject	Cuts the application objected selected on the main menu to the clipboard.
date	Get current date.
dateToNumber	Converts a date to a number
dateToString	Converts a date to a character string
dateToTimestamp	Converts a date to a timestamp
day	Returns the day value of a date
delete	Deletes an object.
deleteConfiguration	Delete a configuration with the given configuration type and configuration. Deletes are cascaded to all associated Configuration Versions and DataSites.
deleteDataSource	Delete the data source with the given name from the list of data sources for this company.
deleteFoodCostPeriodData	Delete food cost data for one period in a fiscal year.
deleteLocationData	Delete all data from selected tables at selected locations.
deleteOrphanedRecords	Delete menus, attachments, prices, and category links that refer to non-existent items.
deleteRecord	Deletes the selected records (or the current record if none selected) on the active form.
deleteRow	Delete a row or rows from a table for a given list of sites that match the given columns and values in the table with the given name.
deleteSelectedObject	Deletes the application objected selected on the main menu.
deleteTable	Deletes a table.
deleteTableList	Delete a table list.
deleteTransactionFilesFromAmazon	Delete transaction files (*.xml and *.fil) from the Amazon Simple Storage Service (S3).
deployMasterChanges	Deploy master data to application tables in the table map.
design	Opens an object in design view.
designForm	Opens a form in design view.
designNew	Opens a new object in design view.
disableControls	Disable the controls with the given names on the active form.
disableFormControls	Disable the controls with the given names on the given form.
displayHelp	Display help for the specified topic.
dumpEmmaCoverageData	Dump Emma coverage data into the specified file
dumpProfile	Disable profiling and write profile report to log.
editCommitListeners	Edit list of email addresses to be notified when packages are committed.
editConfigurationSubscriptions	Edit Configuration Subscriptions.
editConfigurationTypes	Edit Configuration Types.
editConfigurationVersions	Edit Configuration Versions.
editEmailList	Edit a named list of email addresses.
editIrisConfig	Edit IRIS configuration.
editIrisConfigGroups	Edit IRIS config groups.
editIrisItemMenus	Edit IRIS item menus and buttons.
editIrisMenus	Edit IRIS menus and buttons.

editIrisTaxRules	Edit IRIS tax rules.
EditMobileCategoryItemSequence	Edit Mobile Category Item Display Sequence.
EditMobileCategorySequence	Edit Mobile Category Display Sequence.
EditMobileModifierSequence	Edit Mobile Modifier Display Sequence.
editRoles	View and modify roles to which system users may be assigned.
editScheduledTasks	Open form to edit tasks scheduled on the server.
editServerTextFile	Edit a text file on the server.
editSharedTableGroups	Edit groups of tables that are shared together.
editSharing	View and modify table sharing among locations.
editUsers	View and modify system users.
editVersions	Edit Versions.
editVersionSubscriptions	Edit Version Subscriptions.
enableControls	Enable the controls with the given names on the active form.
enableFormControls	Enable the controls with the given names on the given form.
enableProfile	Enable profiling.
encryptAmazonCredentials	Encrypt a property file containing Amazon Web Services (AWS) credentials.
evaluateExpressionAtRemoteSites	Send an expression to a list of remote sites which will evaluate the expression on the server. Only server-side functions may be included in the expression.
executeDirectSQL	Directly execute the given SQL statement on the given data source and return the results in a DataSet. If query is not a select statement, execute the SQL statement and return empty DataSet. Use caution to avoid exceeding available memory.
exit	Exits from the program.
exportData	Exports data from a table or query to a text file.
exportFoodCostFiles	Create ascii export files for CKE food cost system
exportSecurityPolicyToXml	Export users, roles, and other security settings to XML file.
exportSelectedDataToFile	Exports data from a table or query to a text file. Can be run unattended.
extractTestDatabaseChanges	Extract changes to a test database to transaction files
findNextRecord	Finds next record on the current form or spreadsheet using current search options.
findRecord	Finds record on the current form or spreadsheet.
generateFunctionHelp	Generate HTML help file for a list of PluginLoader classes.
generateSnapshot	Generate snapshot of data in snapshot tables
generateTestDatabase	Generate a test database with data from selected locations
getAppObject	Returns application object with specified type and name or null if not found.
getBackgroundProcessViewer	See a window of processes that are running.
getClientPropertyBoolean	Gets a client property associated with the given property name.
getClientPropertyInteger	Gets a client property associated with the given property name.
getClientPropertyString	Gets a client property associated with the given property name.
getCompanyName	Returns the name of the currently selected company.
getConfigProperty	Get configuration property value.

getTransactionViewer	See a dialog of recent file transactions that are either in yellow or red status.
getUnupdatableSites	Return comma-delimited list of unupdatable sites for the active form.
getUpdatableSites	Return comma-delimited list of updatable sites for the active form.
getUserInput	Prompts user to enter a value.
getValue	Gets data for the specified identifier.
getValueListOfUsersWithRole	Get a value list of users with a given role separated by semicolons.
getVersion	Return the current version of EDM
gotoControl	Moves cursor to a control.
gotoFirstRecord	Moves cursor to the first record on the active form.
gotoFormControl	Moves cursor to a control.
gotoLastRecord	Moves cursor to the last record on the active form.
gotoNewRecord	Moves cursor to the new empty record on the active form.
gotoNextRecord	Moves cursor to the next record on the active form.
gotoPrevRecord	Moves cursor to the previous record on the active form.
gotoRecord	Moves cursor to a specific record on the active form.
greatest	Returns the greater of two numbers
hasPermission	Returns true if the current user has the given permission type for the given object type and name.
iif	Returns one of two values bases on expression
importData	Imports a text file to a table.
importFoodCostProducts	Import ascii file containing products for the food cost system.
importSitesInFileList	Import sites listed in given siteldFile from given company.
importTableRelations	Import table relations from a text file
importTransactionsFromFile	Import transactions from a CSV file.
insertTableList	Insert a new table list.
isOnNewRow	True if the currently active form is on a new row, else False.
least	Returns the lesser of two numbers
left	Returns the leftmost characters of a string
len	Gets the length of a string
loadAllTableData	Load data for the specified location from all tables in the table list
loadCentralData	Load schema for a central database table
loadCentralSchema	Load schema for a central database table
loadPicture	Loads a GIF, JPEG, or PNG image from the specified file
login	Log a user into the system.
logout	Logs the current user out of the system.
lower	Changes each letter of a string to lower case
maximizeContentArea	Maximizes the content area in the main EDM window.
mod	Divides two values and returns the remainder

month	Returns the month value of a date
moveTransactions	Moves transactions for the specified locations from the local send directory to the specified directory.
msg	Displays a message box.
notifyClients	Send a notification to connected clients.
nullValue	Return a null value which can be used in other functions. If you want to set the value of a form control to null, use setFormValueToNull() or setValueToNull().
numberToDate	Converts a number to a date
numberToInt	Returns the integer part of a number string.
numberToString	Converts a number to a string
numberToTime	Converts a number to a time
numberToTimestamp	Converts a number to a timestamp
open	Opens an object.
openForm	Ask server to open a form. All parameters must be complete, user will be prompted for routing information if necessary.
openRowSetForm	Open form to edit row sets to restrict which table rows and columns users may edit.
openSitePropertyValueEditor	Open the Site Property Values form.
partitionTable	Partition a table and set a new partition expression and reorganize the rows into the correct partitions.
paste	Pastes an object from the clipboard.
playbackTestScript	Play a test script back.
pow	Raises a number to a power
processForcedTransactions	Receive and apply pending transactions whose effective date has arrived and whose Force flag is true for all sites.
processTransactions	Receive and apply pending transactions from selected locations whose effective date has arrived
promptForTransactionReport	Display dialog for users to set filters and run transaction report.
publishMasterDataTableList	Publish data from the central database, with open and future transactions rolled back, to .csv files in the given directory.
publishMasterDataTables	Publish data from the central database, with open and future transactions rolled back, to .csv files in the given directory.
publishXCEDDataTableList	Publish data from the central database, with open and future transactions rolled back, to .csv files in the given directory.
publishXCEDDataTables	Publish data from the central database, with open and future transactions rolled back, to .csv files in the given directory.
queryAvg	Gets the average of the values of a column in a table or query using a where clause.
queryCount	Gets the count of the values of a column in a table or query using a where clause.
queryFirst	Gets the first value of a column in a table or query using a where clause.
queryLast	Gets the last value of a column in a table or query using a where clause.
queryMax	Gets the maximum value of a column in a table or query using a where clause.
queryMin	Gets the minimum value of a column in a table or query using a where clause.
querySum	Gets the sum of the values of a column in a table or query using a where clause.

queryValue	Directly execute an SQL statement and return the value of the first column in the first row. If query is not a select statement, execute the SQL statement and return a blank string. Use caution to avoid exceeding available memory.
random	Returns a random number between 0 and the passed number
receiveSiteFilesViaWeb	Receive transaction files over HTTP web connection.
receiveTransactionFilesFromAmazon	Receive transaction files from Amazon Simple Storage Service (S3).
receiveTransactionFilesFromJMS	Receive files from the configured JMS queue into the local incoming folder.
reloadCachedData	Reload cached data such as configuration, site list, etc.
rename	Renames an object to a new name.
replaceAll	Replaces each substring of this string that matches the given regular expression with the given replacement.
replaceTransWithRefresh	Replaces transaction files for the selected locations with a refresh table transaction using the specified snapshot.
requery	Requeries the records on the active form.
requeryControl	Requeries the records of the control's source on the active form.
requestDatedTableRefresh	Send request to location to send table data to replace existing data
requestFile	Copy a file to other locations
requestTableRefresh	Send request to location to send table data to replace existing data
resetSite	Deprecated - use resetSites instead. Send Reset transaction to reset a site to its newly installed state.
resetSites	Send transactions to reset remote sites to their newly installed state and commit the transactions.
restoreContentArea	Restores the content area in the main EDM window.
restoreTestCheckpoint	Disconnect all database connections, restore checkpoint 1, and re-initialize server
retryTest	Retry a test that failed during runTest.
right	Returns the rightmost characters of a string
round	Rounds a number at the specified number of digits past the decimal
runApp	Runs an application as if from the command line. For example, to open the config.xml file in notepad when transactions are processed, set the filespec to <code>c:\windows\system32\notepad.exe</code> , the arguments to <code>c:\EDMweb\config.xml</code> , and the wait flag to <code>>false</code> . To run a batch file you could set the filespec to <code>cmd</code> and the arguments to <code>c c:\EDMweb\go.bat</code> and the wait flag to <code>>false</code> .
runJUnitTest	Run the JUnit tests in the given class.
runPriceChangeByPriceGroupReport	Run price change by price group report with user-specified filters and sorting.
runPriceChangeByPriceReport	Run price change by price report with user-specified filters and sorting.
runPriceChangeReport	Run price report with user-specified filters and sorting.
runReport	Run a report.
runSQL	Runs an SQL statement.
runTaxChangeReport	Run tax change report with user-specified filters and sorting.
runTest	Invoke each method whose name starts with 'test' in the given test class.
runTestMethod	Invoke the given test method.

runTestMethods	Invoke the given test method and all methods that follow it in the given class.
runTestUI	Invoke each method whose name starts with 'test' in the given test class.
saveRecord	Saves changes to the current record on the active form.
scheduleVersionAssignments	Schedules a version of site subscriptions. Only sites having the Site Property Name and Value will be schedulable.
select	Selects either an open object or an object in the application window.
selectAllRecords	Selects all records on the active form.
selectLocation	Select one remote location from the list of remote locations maintained in a central database.
selectLocations	Prompt user to select locations from a location tree.
selectLocationsAndPackage	Prompt user to select locations from a location tree and a package for transactions.
selectPackage	Prompt user to select from a list of transaction packages or create a new one.
selectRecord	Selects current record on the active form.
selectSessionPackage	Specify the package to use for all changes until a different session package is selected.
sendAddSchemasTransaction	Send an transaction to update the table list and schema at selected sites.
sendDatedTableRefresh	Send table data to locations to replace existing data
sendDeleteFileTransaction	Send transaction to delete a file.
sendEvaluateTransaction	Send a transaction to cause the receiver to evaluate an expression.
sendExecuteTransaction	Send transaction to execute an operation system command at selected sites.
sendFile	Send a file to other locations
sendGetSchemasTransaction	Send a transaction to get the table list and schema from selected sites.
sendRefreshToTestSite	Send table refresh transaction from a site to a lab/test site.
sendRemoteFiles	Send files listed in remote files table to sites.
sendRenameFileTransaction	Send transaction to rename a file.
sendSiteFilesViaWeb	Send transaction files over HTTP web connection.
sendSynchronizeDirectoryTransaction	Send a transaction to synchronize a local directory with a directory at other sites.
sendTableRefresh	Send table data to locations to replace existing data
sendTransactionFilesToAmazon	Send transaction files to Amazon Simple Storage Service (S3).
sendTransactionFilesToJMS	Send outgoing files for the given sites to a JMS queue.
setClientPropertyBoolean	Sets a client property value with the given name.
setClientPropertyInteger	Sets a client property value with the given name.
setClientPropertyString	Sets a client property value with the given name.
setConfigProperty	Set configuration property value.
setControlProperty	Sets a property value for a control on the active form.
setFormControlProperty	Sets a property value for a control on a form.
setFormRowSource	Sets the row source value for a control on the active form.
setFormSiteValue	Sets site specific data for a control on the specified form.
setFormSiteValueToNull	Sets site specific data for a control on the specified form to null.
setFormValue	Sets data for a control on a form.

setFormValueToNull	Sets data for controls on a form to null.
setLastReceivedTranFileNumberToIncoming	Set the last received transaction file number for a site to the lowest transaction file number in the incoming directory - 1.
setLogConsoleLevel	Sets level of log messages written to console.
setLogFileLevel	Sets level of log messages written to log file.
setPropertyFileValue	Set the value of a property in a property file.
setRecordModified	Sets the current record as modified so that it will be saved.
setRowSource	Sets the row source value for a control on the active form.
setSiteValue	Sets site specific data for a control on the active form.
setSiteValueToNull	Sets site specific data for a control on the active form to null.
setValue	Sets data for a control on the active form.
setValueToNull	Sets data for a control on the active form to null.
shareLocationData	Share data for selected tables from one location to another location
showAboutDialog	Displays the About dialog with system name and copyright info.
showCopySiteSubscriptionsForm	Show the UI for selecting a source site, target sites, and configurations.
showCreateSiteSubscriptionsForm	Show the UI for creating site subscriptions.
showDocument	Display a document in a browser window. Applets can only display documents on the same server they were started from.
showErrors	Turn error messages on or off
showLdapPropertyForm	Displays the LDAP configuration form.
showSiteManager	Display site manager window.
showWarnings	Turn warning messages on or off
sleep	Causes thread to sleep for the specified number of milliseconds.
space	Creates a string of blanks
startRecordingTestScript	Tell server to start recording test script with the next login command.
stopRecordingTestScript	Tell server to stop recording test script.
stringToDate	Converts a date string to a date value
stringToInt	Converts a string to an integer.
stringToNumber	Converts a string to a number
stringToTime	Converts a time string to a time value
stringToTimestamp	Converts a timestamp string to a timestamp value
substring	Returns characters from the middle of a string
synchronizeSitePropertyValues	Import and sync the site property values from an outside data source
testJOptionPane	Test JOptionPane for slow closing and going behind the browser window.
time	Gets current time of day
timestamp	Get current date and time.
timestampToDate	Converts a timestamp to a date
timestampToNumber	Converts a timestamp to a number

timestampToString	Converts a timestamp to a character string
timestampToTime	Converts a timestamp to a time
timeToNumber	Converts a time to a number
timeToString	Converts a time to a character string
timeToTimestamp	Converts a time to a timestamp
transferLocationFilesViaFTP	Transfer files over FTP connection.
transferLocationFilesViaRAS	Transfer files over network connection.
transferLocationFilesViaWeb	Transfer files over HTTP web connection.
transferSiteFilesViaFTP	Transfer transaction files over FTP connection.
transferSiteFilesViaRAS	Transfer transaction files over network connection.
transferSiteFilesViaWeb	Transfer transaction files over HTTP web connection.
trim	Trims the blanks from both ends of a string
trimLeft	Trims the blanks from the left end of a string
trimRight	Trims the blanks from the right end of a string
undo	Performs undo operation in the active window.
unlockAppObjects	Prompt user to unlock selected application objects that are locked while there design is being edited.
unshareLocationData	Stop sharing data for selected tables from one location to another location
updateAmazonDirToMatchLocalDir	Update an Amazon S3 directory to match a local directory.
updateLocalDirToMatchAmazonDir	Update a local directory to match a directory on Amazon S3.
updateLocalDirToMatchWebDir	Update a local directory to match a directory on the EDM web server.
updateRows	Update selected rows for selected sites and generate transactions for the changes.
updateTableList	Update a table list.
updateTransferSettings	Save the given transfer host, user ID, password, and company name used to connect to a central EDM server
updateWebDirToMatchLocalDir	Update an directory on the EDM web server to match a local directory.
upper	Changes each letter of a string to upper case
validateTableList	Validate references between tables
validateTables	Validate references between tables
view	Opens an Table, Query, or Form on the client.
viewForm	Ask server to open a form on the client that is visible in its default view. All parameters must be complete, user will be prompted for routing information if necessary.
waitForServerJobs	Wait for all server jobs to complete. If not server jobs are running when the method is called, wait up to one second for server jobs to start. After waiting for a server job to finish, wait up to one second for another server job to start. Total minimum wait time is two seconds.
year	Returns the year value of a date

Functions by Category

String Functions

Function	What it does...
ascii	Converts the first character of a string to an ASCII value
asciiToChar	Converts an ASCII code to a character
getVersion	Return the current version of EDM
left	Returns the leftmost characters of a string
len	Gets the length of a string
lower	Changes each letter of a string to lower case
replaceAll	Replaces each substring of this string that matches the given regular expression with the given replacement.
right	Returns the rightmost characters of a string
space	Creates a string of blanks
substring	Returns characters from the middle of a string
trim	Trims the blanks from both ends of a string
trimLeft	Trims the blanks from the left end of a string
trimRight	Trims the blanks from the right end of a string
upper	Changes each letter of a string to upper case

Numeric Functions

Function	What it does...
abs	Get the absolute value of the given number. If the number is null, a DataNumber with the value null is returned. If the number is blank, 0 is returned.
greatest	Returns the greater of two numbers
least	Returns the lesser of two numbers
mod	Divides two values and returns the remainder
pow	Raises a number to a power
random	Returns a random number between 0 and the passed number
round	Rounds a number at the specified number of digits past the decimal

Date Functions

Function	What it does...
date	Get current date.
day	Returns the day value of a date
month	Returns the month value of a date
time	Gets current time of day
timestamp	Get current date and time.
year	Returns the year value of a date

Conversion Functions

Function	What it does...
dateToNumber	Converts a date to a number
dateToString	Converts a date to a character string
dateToTimestamp	Converts a date to a timestamp
numberToDate	Converts a number to a date
numberToInt	Returns the integer part of a number string.
numberToString	Converts a number to a string
numberToTime	Converts a number to a time
numberToTimestamp	Converts a number to a timestamp
stringToDate	Converts a date string to a date value
stringToInt	Converts a string to an integer.
stringToNumber	Converts a string to a number
stringToTime	Converts a time string to a time value
stringToTimestamp	Converts a timestamp string to a timestamp value
timestampToDate	Converts a timestamp to a date
timestampToNumber	Converts a timestamp to a number
timestampToString	Converts a timestamp to a character string
timestampToTime	Converts a timestamp to a time
timeToNumber	Converts a time to a number
timeToString	Converts a time to a character string
timeToTimestamp	Converts a time to a timestamp

Object Functions

Function	What it does...
close	Closes the specified open object window (form view, design view, or print preview).
closeActiveWindow	Closes the active window.
copy	Copies an object to the clipboard.
copySelectedObject	Copies the application object selected on the main menu to the clipboard.
cut	Cuts an object to the clipboard.
cutSelectedObject	Cuts the application object selected on the main menu to the clipboard.
delete	Deletes an object.
deleteSelectedObject	Deletes the application object selected on the main menu.
deleteTable	Deletes a table.
design	Opens an object in design view.
designForm	Opens a form in design view.
designNew	Opens a new object in design view.
getAppObject	Returns application object with specified type and name or null if not found.

open	Opens an object.
paste	Pastes an object from the clipboard.
rename	Renames an object to a new name.
select	Selects either an open object or an object in the application window.
unlockAppObjects	Prompt user to unlock selected application objects that are locked while there design is being edited.
view	Opens an Table, Query, or Form on the client.

Record Functions

Function	What it does...
deleteRecord	Deletes the selected records (or the current record if none selected) on the active form.
findNextRecord	Finds next record on the current form or spreadsheet using current search options.
findRecord	Finds record on the current form or spreadsheet.
gotoFirstRecord	Moves cursor to the first record on the active form.
gotoLastRecord	Moves cursor to the last record on the active form.
gotoNewRecord	Moves cursor to the new empty record on the active form.
gotoNextRecord	Moves cursor to the next record on the active form.
gotoPrevRecord	Moves cursor to the previous record on the active form.
gotoRecord	Moves cursor to a specific record on the active form.
saveRecord	Saves changes to the current record on the active form.
selectAllRecords	Selects all records on the active form.
selectRecord	Selects current record on the active form.

Form Functions

Function	What it does...
addMultipleRows	Add multiple rows to an open form based on a list or combo box
cancelEvent	Cancel current event on the active form.
deleteOrphanedRecords	Delete menus, attachments, prices, and category links that refer to non-existent items.
editIrisConfig	Edit IRIS configuration.
editIrisConfigGroups	Edit IRIS config groups.
editIrisItemMenus	Edit IRIS item menus and buttons.
editIrisMenus	Edit IRIS menus and buttons.
editIrisTaxRules	Edit IRIS tax rules.
getClientPropertyBoolean	Gets a client property associated with the given property name.
getClientPropertyInteger	Gets a client property associated with the given property name.
getClientPropertyString	Gets a client property associated with the given property name.
getEstimatedRowCount	Get estimated number of rows in the data source of the given form. Will use active form is no form name given.

getFormNamedControlValues	Get a string containing the names and values of a list of controls from a form. Example: "name1","value1","name2","value2",...
getFormValue	Gets the value of the given control on the given form.
getNamedControlValues	Get a string containing the names and values of a list of controls from the active form. Example: "name1","value1","name2","value2",...
getText	Returns the text from the given control.
getValue	Gets data for the specified identifier.
gotoControl	Moves cursor to a control.
gotoFormControl	Moves cursor to a control.
query	Requeries the records on the active form.
queryControl	Requeries the records of the control's source on the active form.
runPriceChangeByPriceGroupReport	Run price change by price group report with user-specified filters and sorting.
runPriceChangeByPriceReport	Run price change by price report with user-specified filters and sorting.
runPriceChangeReport	Run price report with user-specified filters and sorting.
runTaxChangeReport	Run tax change report with user-specified filters and sorting.
setClientPropertyBoolean	Sets a client property value with the given name.
setClientPropertyInteger	Sets a client property value with the given name.
setClientPropertyString	Sets a client property value with the given name.
setControlProperty	Sets a property value for a control on the active form.
setFormControlProperty	Sets a property value for a control on a form.
setFormRowSource	Sets the row source value for a control on the active form.
setFormSiteValue	Sets site specific data for a control on the specified form.
setFormSiteValueToNull	Sets site specific data for a control on the specified form to null.
setFormValue	Sets data for a control on a form.
setFormValueToNull	Sets data for controls on a form to null.
setRecordModified	Sets the current record as modified so that it will be saved.
setRowSource	Sets the row source value for a control on the active form.
setSiteValue	Sets site specific data for a control on the active form.
setSiteValueToNull	Sets site specific data for a control on the active form to null.
setValue	Sets data for a control on the active form.
setValueToNull	Sets data for a control on the active form to null.
undo	Performs undo operation in the active window.

Lookup Functions

Function	What it does...
queryAvg	Gets the average of the values of a column in a table or query using a where clause.
queryCount	Gets the count of the values of a column in a table or query using a where clause.
queryFirst	Gets the first value of a column in a table or query using a where clause.

queryLast	Gets the last value of a column in a table or query using a where clause.
queryMax	Gets the maximum value of a column in a table or query using a where clause.
queryMin	Gets the minimum value of a column in a table or query using a where clause.
querySum	Gets the sum of the values of a column in a table or query using a where clause.

System Functions

Function	What it does...
addRow	Insert a row in a table for a given list of sites that do not already have the row. For example, when entering a
addSystemEventListener	Add the listener in the given class to the list of system event listeners.
addThisSiteToCentral	Call the central web service to add this remote site to the central database.
attachMasterTables	Attach tables from an existing database so it can be edited
attachTable	Attach tables from an existing database so it can be edited
callEvaluateExpressionWebMethod	Call the EvaluateExpression web method.
createTransactionTriggers	Create SQL scripts so triggers on application tables at remote sites can generate EDM transactions.
deleteRow	Delete a row or rows from a table for a given list of sites that match the given columns and values in the table with the given name.
deleteTableList	Delete a table list.
displayHelp	Display help for the specified topic.
dumpEmmaCoverageData	Dump Emma coverage data into the specified file
dumpProfile	Disable profiling and write profile report to log.
editEmailList	Edit a named list of email addresses.
enableProfile	Enable profiling.
evaluateExpressionAtRemoteSites	Send an expression to a list of remote sites which will evaluate the expression on the server. Only server-side functions may be included in the expression.
executeDirectSQL	Directly execute the given SQL statement on the given data source and return the results in a DataSet. If query is not a select statement, execute the SQL statement and return empty DataSet. Use caution to avoid exceeding available memory.
exit	Exits from the program.
exportData	Exports data from a table or query to a text file.
exportSecurityPolicyToXml	Export users, roles, and other security settings to XML file.
getBackgroundProcessViewer	See a window of processes that are running.
getCurrentUserId	Return the ID of the user that is currently logged in or "" if no user is logged in.
getIniValue	Returns value for the given key from the given ini file or null if not found.
getServerCallsPerSecond	Returns the average number of calls per second that can be made to the server after sending as many calls to the server as possible for 5 seconds.
getServerFile	Return the bytes of a file from the server.
getServerFileList	Gets a list of files in the specified directory on the server.
getServerTime	Returns the current system time on the server in milliseconds since 1/1/1970 UTC.
getSessions	Returns a list of the current active sessions on the server.

getSystemProperty	Get the value of a property in the system.properties file.
getTableAndQueryList	Get a sorted list of tables and queries in this application separated by semicolons.
getTableList	Get a sorted list of tables in this application separated by semicolons.
getTransactionViewer	See a dialog of recent file transactions that are either in yellow or red status.
getUserInput	Prompts user to enter a value.
hasPermission	Returns true if the current user has the given permission type for the given object type and name.
iif	Returns one of two values bases on expression
importData	Imports a text file to a table.
importTransactionsFromFile	Import transactions from a CSV file.
insertTableList	Insert a new table list.
loadPicture	Loads a GIF, JPEG, or PNG image from the specified file
login	Log a user into the system.
logout	Logs the current user out of the system.
msg	Displays a message box.
openSitePropertyValueEditor	Open the Site Property Values form.
partitionTable	Partition a table and set a new partition expression and reorganize the rows into the correct partitions.
playbackTestScript	Play a test script back.
queryValue	Directly execute an SQL statement and return the value of the first column in the first row. If query is not a select statement, execute the SQL statement and return a blank string. Use caution to avoid exceeding available memory.
retryTest	Retry a test that failed during runTest.
runApp	Runs an application as if from the command line. For example, to open the config.xml file in notepad when transactions are processed, set the filespec to <code>c:\windows\system32\notepad.exe</code> , the arguments to <code>c:\EDMweb\config.xml</code> , and the wait flag to <code>false</code> . To run a batch file you could set the filespec to <code>cmd</code> and the arguments to <code>c:\EDMweb\go.bat</code> and the wait flag to <code>false</code> .
runJUnitTest	Run the JUnit tests in the given class.
runReport	Run a report.
runSQL	Runs an SQL statement.
runTest	Invoke each method whose name starts with 'test' in the given test class.
runTestMethod	Invoke the given test method.
runTestMethods	Invoke the given test method and all methods that follow it in the given class.
runTestUI	Invoke each method whose name starts with 'test' in the given test class.
selectLocation	Select one remote location from the list of remote locations maintained in a central database.
setLogConsoleLevel	Sets level of log messages written to console.
setLogFileLevel	Sets level of log messages written to log file.
showAboutDialog	Displays the About dialog with system name and copyright info.
showDocument	Display a document in a browser window. Applets can only display documents on the same server they were started from.
showErrors	Turn error messages on or off

showLdapPropertyForm	Displays the LDAP configuration form.
showWarnings	Turn warning messages on or off
sleep	Causes thread to sleep for the specified number of milliseconds.
startRecordingTestScript	Tell server to start recording test script with the next login command.
stopRecordingTestScript	Tell server to stop recording test script.
updateTableList	Update a table list.
waitForServerJobs	Wait for all server jobs to complete. If not server jobs are running when the method is called, wait up to one second for server jobs to start. After waiting for a server job to finish, wait up to one second for another server job to start. Total minimum wait time is two seconds.

Enterprise Data Manager Functions

Function	What it does...
addDataSource	Add a new data source to the system configuration.
attachRemoteTable	Attach a table from an existing database so it can be edited
changePackageStatus	Change status of the transactions for sites in a package to a new status.
changePassword	Change the current user's password.
clearSessionPackage	Remove the selected session package on the client to users have to select a package when making changes.
commitTransactions	Commit packages of transactions so they are sent to selected locations
configureApplicationVersions	Configure known versions of application data sources.
copyAndRefreshSiteData	Copy data for selected tables from one site to other sites and refresh the tables at the other sites on the effective date.
copyCompany	Create a new company by copying the configuration of an existing company.
copyFoodCostPeriodData	Copy food cost data from one period to another.
copyLocationData	Copy site data for a list of tables to other sites. DEPRECATED Use copySiteData instead.
copySiteData	Copy data for selected tables from one location to another location
createRemoteInstaller	Create an installation program that can be downloaded and used to install EDM at a remote site with pre-configured settings for the current company.
deleteDataSource	Delete the data source with the given name from the list of data sources for this company.
deleteFoodCostPeriodData	Delete food cost data for one period in a fiscal year.
deleteLocationData	Delete all data from selected tables at selected locations.
deleteTransactionFilesFromAmazon	Delete transaction files (*.xml and *.fil) from the Amazon Simple Storage Service (S3).
disableControls	Disable the controls with the given names on the active form.
disableFormControls	Disable the controls with the given names on the given form.
editCommitListeners	Edit list of email addresses to be notified when packages are committed.
EditMobileCategoryItemSequence	Edit Mobile Category Item Display Sequence.
EditMobileCategorySequence	Edit Mobile Category Display Sequence.
EditMobileModifierSequence	Edit Mobile Modifier Display Sequence.
editRoles	View and modify roles to which system users may be assigned.
editScheduledTasks	Open form to edit tasks scheduled on the server.

editServerTextFile	Edit a text file on the server.
editSharedTableGroups	Edit groups of tables that are shared together.
editSharing	View and modify table sharing among locations.
editUsers	View and modify system users.
enableControls	Enable the controls with the given names on the active form.
enableFormControls	Enable the controls with the given names on the given form.
encryptAmazonCredentials	Encrypt a property file containing Amazon Web Services (AWS) credentials.
exportFoodCostFiles	Create ascii export files for CKE food cost system
exportSelectedDataToFile	Exports data from a table or query to a text file. Can be run unattended.
extractTestDatabaseChanges	Extract changes to a test database to transaction files
generateFunctionHelp	Generate HTML help file for a list of PluginLoader classes.
generateSnapshot	Generate snapshot of data in snapshot tables
generateTestDatabase	Generate a test database with data from selected locations
getCompanyName	Returns the name of the currently selected company.
getConfigProperty	Get configuration property value.
getConfigPropertyWithDefault	Get configuration property value.
getControlValueRow	Create row with column values from the given list of controls on the active form.
getEffectiveDate	Get date/time transactions becomes effective for the active form.
getForceProcessing	Get flag for whether transactions for the active form should be processed when only processing forced transactions.
getNextUnusedId	Gets the next unused ID for the given table and column, optionally within a range.
getPackageDescription	Get description of package that contains transaction for the active form.
getPackageName	Get name of package that contains transaction for the active form.
getPropertyFileValue	Get the value of a property in a property file.
getSchemaVersions	Returns a list of known application database schemas supported by the system.
getSelectedSites	Get the selected sites for the active form.
getSessionPackageName	Returns the name of the currently selected package.
getSessionPackageVersionID	Returns the ID of the currently selected package's version or null if there is no selected package or if the selected package doesn't have a version.
getSessionPackageVersionName	Returns the name of the currently selected package's version or null if there is no selected package or if the selected package doesn't have a version.
getTerminationDate	Get date/time transactions terminate for the active form.
getUnupdatableSites	Return comma-delimited list of unupdatable sites for the active form.
getUpdatableSites	Return comma-delimited list of updatable sites for the active form.
getValueListOfUsersWithRole	Get a value list of users with a given role separated by semicolons.
importFoodCostProducts	Import ascii file containing products for the food cost system.
importSitesInFileList	Import sites listed in given siteldFile from given company.
importTableRelations	Import table relations from a text file
isOnNewRow	True if the currently active form is on a new row, else False.

loadAllTableData	Load data for the specified location from all tables in the table list
loadCentralData	Load schema for a central database table
loadCentralSchema	Load schema for a central database table
maximizeContentArea	Maximizes the content area in the main EDM window.
moveTransactions	Moves transactions for the specified locations from the local send directory to the specified directory.
notifyClients	Send a notification to connected clients.
nullValue	Return a null value which can be used in other functions. If you want to set the value of a form control to null, use setFormValueToNull() or setValueToNull().
openForm	Ask server to open a form. All parameters must be complete, user will be prompted for routing information if necessary.
openRowSetForm	Open form to edit row sets to restrict which table rows and columns users may edit.
processForcedTransactions	Receive and apply pending transactions whose effective date has arrived and whose Force flag is true for all sites.
processTransactions	Receive and apply pending transactions from selected locations whose effective date has arrived
promptForTransactionReport	Display dialog for users to set filters and run transaction report.
receiveSiteFilesViaWeb	Receive transaction files over HTTP web connection.
receiveTransactionFilesFromAmazon	Receive transaction files from Amazon Simple Storage Service (S3).
receiveTransactionFilesFromJMS	Receive files from the configured JMS queue into the local incoming folder.
reloadCachedData	Reload cached data such as configuration, site list, etc.
replaceTransWithRefresh	Replaces transaction files for the selected locations with a refresh table transaction using the specified snapshot.
requestDatedTableRefresh	Send request to location to send table data to replace existing data
requestFile	Copy a file to other locations
requestTableRefresh	Send request to location to send table data to replace existing data
resetSite	Deprecated - use resetSites instead. Send Reset transaction to reset a site to its newly installed state.
resetSites	Send transactions to reset remote sites to their newly installed state and commit the transactions.
restoreContentArea	Restores the content area in the main EDM window.
restoreTestCheckpoint	Disconnect all database connections, restore checkpoint 1, and re-initialize server
selectLocations	Prompt user to select locations from a location tree.
selectLocationsAndPackage	Prompt user to select locations from a location tree and a package for transactions.
selectPackage	Prompt user to select from a list of transaction packages or create a new one.
selectSessionPackage	Specify the package to use for all changes until a different session package is selected.
sendAddSchemasTransaction	Send an transaction to update the table list and schema at selected sites.
sendDatedTableRefresh	Send table data to locations to replace existing data
sendDeleteFileTransaction	Send transaction to delete a file.
sendEvaluateTransaction	Send a transaction to cause the receiver to evaluate an expression.
sendExecuteTransaction	Send transaction to execute an operation system command at selected sites.

sendFile	Send a file to other locations
sendGetSchemasTransaction	Send a transaction to get the table list and schema from selected sites.
sendRefreshToTestSite	Send table refresh transaction from a site to a lab/test site.
sendRemoteFiles	Send files listed in remote files table to sites.
sendRenameFileTransaction	Send transaction to rename a file.
sendSiteFilesViaWeb	Send transaction files over HTTP web connection.
sendSynchronizeDirectoryTransaction	Send a transaction to synchronize a local directory with a directory at other sites.
sendTableRefresh	Send table data to locations to replace existing data
sendTransactionFilesToAmazon	Send transaction files to Amazon Simple Storage Service (S3).
sendTransactionFilesToJMS	Send outgoing files for the given sites to a JMS queue.
setConfigProperty	Set configuration property value.
setLastReceivedTranFileNumberToIncoming	Set the last received transaction file number for a site to the lowest transaction file number in the incoming directory - 1.
setPropertyFileValue	Set the value of a property in a property file.
shareLocationData	Share data for selected tables from one location to another location
showSiteManager	Display site manager window.
synchronizeSitePropertyValues	Import and sync the site property values from an outside data source
testJOptionPane	Test JOptionPane for slow closing and going behind the browser window.
transferLocationFilesViaFTP	Transfer files over FTP connection.
transferLocationFilesViaRAS	Transfer files over network connection.
transferLocationFilesViaWeb	Transfer files over HTTP web connection.
transferSiteFilesViaFTP	Transfer transaction files over FTP connection.
transferSiteFilesViaRAS	Transfer transaction files over network connection.
transferSiteFilesViaWeb	Transfer transaction files over HTTP web connection.
unshareLocationData	Stop sharing data for selected tables from one location to another location
updateAmazonDirToMatchLocalDir	Update an Amazon S3 directory to match a local directory.
updateLocalDirToMatchAmazonDir	Update a local directory to match a directory on Amazon S3.
updateLocalDirToMatchWebDir	Update a local directory to match a directory on the EDM web server.
updateRows	Update selected rows for selected sites and generate transactions for the changes.
updateTransferSettings	Save the given transfer host, user ID, password, and company name used to connect to a central EDM server
updateWebDirToMatchLocalDir	Update an directory on the EDM web server to match a local directory.
validateTableList	Validate references between tables
validateTables	Validate references between tables
viewForm	Ask server to open a form on the client that is visible in its default view. All parameters must be complete, user will be prompted for routing information if necessary.

Master Data Management Functions

Function	What it does...
----------	-----------------

applyEffectiveMasterDataTrans	Queries the audit table for committed transactions on master data tables whose effective date has arrived and calls the appropriate master data change listener for each transaction then changes the status of the transaction to Successful.
copyConfigurationVersionData	Copy the data from one configuration version to another version of the same configuration overwriting any existing data.
copySiteSubscriptions	Copy the configuration version subscriptions from one site to other sites leaving the version ID null. If the source site has no subscriptions or the list of configuration ID's to copy is empty, do nothing.
copyVersionData	Copy the data from one version to another version for all configurations overwriting any existing data.
createConfigurationVersionDataSite	Creates a data site to hold version data for configuration version. The ID of the new site will be an unused, negative site number. The name of the site will be the name of the configuration plus a hyphen plus the name of the version. Each table in the configuration type will be added to the list of managed site tables. If there is a previous version of the configuration then the data from the previous version will be copied to the new version.
createConfigurationVersionsCmd	Create configuration versions for the given configuration type and configuration.
deleteConfiguration	Delete a configuration with the given configuration type and configuration. Deletes are cascaded to all associated Configuration Versions and DataSites.
deployMasterChanges	Deploy master data to application tables in the table map.
editConfigurationSubscriptions	Edit Configuration Subscriptions.
editConfigurationTypes	Edit Configuration Types.
editConfigurationVersions	Edit Configuration Versions.
editVersions	Edit Versions.
editVersionSubscriptions	Edit Version Subscriptions.
getLastDeploymentInfo	Get the last deployment date and user name with the given deployment type.
getLastVersionChangeInfo	Get the last change date and user name with the given version.
getMasterTableConfigurationType	Gets the Configuration Type for the given table with the given name.
getNumberOfAssignedSites	Get the number of sites assigned to the given version.
getNumberOfUnassignedSites	Get the number of sites that are not assigned to the given version.
publishMasterDataTableList	Publish data from the central database, with open and future transactions rolled back, to .csv files in the given directory.
publishMasterDataTables	Publish data from the central database, with open and future transactions rolled back, to .csv files in the given directory.
publishXCEDDataTableList	Publish data from the central database, with open and future transactions rolled back, to .csv files in the given directory.
publishXCEDDataTables	Publish data from the central database, with open and future transactions rolled back, to .csv files in the given directory.
scheduleVersionAssignments	Schedules a version of site subscriptions. Only sites having the Site Property Name and Value will be schedulable.
showCopySiteSubscriptionsForm	Show the UI for selecting a source site, target sites, and configurations.
showCreateSiteSubscriptionsForm	Show the UI for creating site subscriptions.

abs

Get the absolute value of the given number. If the number is null, a DataNumber with the value null is returned. If the number is blank, 0 is returned.

Syntax:

abs('number')

Parameters:

number - Number whose absolute value is to be returned.

Returns:

Absolute value of number

Class:

EDMcommon.function.CommonFunctions

addDataSource

Add a new data source to the system configuration.

Syntax:

addDataSource('name', 'version', 'type', 'connectionString', 'encryptConnectionString', 'database', 'server', 'trusted', 'user', 'password', 'encryptPassword', 'isSystemDataSource')

Parameters:

name - Data source name used by the system.

version - Application-specific database version number.

type - Data source type, see DbType class.

connectionString - Connection string for data source. If specified, it will be used as-is to connect to the data source. If not specified, it will be built from other attributes. This allows users to customize the connection string as necessary. Care should be taken that the database and server names in the connection string match the database and server name from the other attributes.

encryptConnectionString - 'Y' if connection string should be encrypted when saved.

database - Database name.

server - Server name, may have :PORT or :SID suffixes.

trusted - 'Y' if this is ODBC data source that uses Windows login, else 'N'.

user - User name used to log into data source.

password - Password used to log into data source.

encryptPassword - 'Y' if password should be encrypted when saved.

isSystemDataSource - 'Y' if this is the system data source.

Returns:

Void.

Class:

EDMclient.ClientFunctions

addMultipleRows

Add multiple rows to an open form based on a list or combo box

Syntax:

addMultipleRows('formName', 'controlName')

Parameters:

formName - Name of the form

controlName - Name of the list of combo control

Returns:

Void.

Class:

EDMclient.ClientFunctions

addRow

Insert a row in a table for a given list of sites that do not already have the row. For example, when entering a

product, this function can be called in the AfterInsert event to add a row to the product type table. For example, to add rows to the TNG_dbo_ProductTypes table after the user inserts a row in the TNG_dbo_Products table, set the afterInsert property of the Product form to: "`=addRow('TNG_dbo_ProductTypes', getNamedControlValues('id,name,description'), getSelectedSites())"`"

Syntax:

`addRow('tableName', 'namedColumnValues', 'siteIdList')`

Parameters:

tableName - Name of a table.

namedColumnValues - List of column names and values of the form: "name1","value1","name2","value2"...

siteIdList -

Returns:

Void.

Class:

EDMcommon.cmd.AddRowCmd

addSystemEventListener

Add the listener in the given class to the list of system event listeners.

Syntax:

`addSystemEventListener('listenerClassName')`

Parameters:

listenerClassName - Name of listener class.

Returns:

Void.

Class:

EDMcommon.cmd.AddSystemEventListenerCmd

addThisSiteToCentral

Call the central web service to add this remote site to the central database.

If the site does not exist in the central database it will be added. If it is the first site in the company, a table list will be requested. If the site data has not been loaded then a table refresh will be requested.

Syntax:

`addThisSiteToCentral('url', 'userId', 'password')`

Parameters:

url - URL of the central server such as <http://xpress.xpient.com/EDM>.

userId - User ID used to log in to central system.

password - Password used to log into central system.

Returns:

Void.

Class:

EDMcommon.cmd.AddThisSiteToCentralCmd

applyEffectiveMasterDataTrans

Queries the audit table for committed transactions on master data tables whose effective date has arrived and calls the appropriate master data change listener for each transaction then changes the status of the transaction to Successful.

Syntax:

applyEffectiveMasterDataTrans()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.plugin.mdm.cmd.ApplyEffectiveMasterDataTransCmd

ascii

Converts the first character of a string to an ASCII value

Syntax:

ascii('astring')

Parameters:

astring - String whose first character is to be converted

Returns:

ASCII value of the first character of the string

Class:

EDMcommon.function.CommonFunctions

asciiToChar

Converts an ASCII code to a character

Syntax:

asciiToChar(asciiCode)

Parameters:

asciiCode - ASCII code to convert

Returns:

Character that is assigned to the ASCII code

Class:

EDMcommon.function.CommonFunctions

attachMasterTables

Attach tables from an existing database so it can be edited

Attaches master tables by creating a central table with the CDMLOCID column added to the table and all indexes. Adds a remote table record for the central site with a version of '1'. Adds a remote table record for every remote site with a version of null so transactions are not generated to the site. Adds a remote data source version for the central site of '1'. Adds a remote data source version for every remote site of '0'. Adds the tables to a table list called 'Attached Master Tables'. Adds a data source version 0 to the convert file and a version 1 which adds all the tables to the convert file.

Syntax:

attachMasterTables('tableNames', 'dataSourceName')

Parameters:

tableNames - Comma-delimited list of table names to attach (blank means prompt user).

dataSourceName - Name of the data source in the configuration that contains the tables (blank means prompt user).

Returns:

Void.

Class:

EDMclient.plugin.mdm.cmd.AttachMasterTablesCmd

attachRemoteTable

Attach a table from an existing database so it can be edited

Syntax:

attachRemoteTable('tableName', 'dataSourceName', 'tableVersion', 'ignoreInserts', 'ignoreUpdates', 'ignoreDeletes')

Parameters:

tableName - Name of table to be attached (blank to prompt user).

dataSourceName - Name of data source from which to attach the table (blank to prompt user).

tableVersion - Version number of table to be attached (blank to prompt user).

ignoreInserts - True if central database should not accept inserts from remote location (blank to prompt user).

ignoreUpdates - True if central database should not accept updates from remote location (blank to prompt user).

ignoreDeletes - True if central database should not accept deletes from remote location (blank to prompt user).

Returns:

Void.

Class:

EDMcommon.cmd.AttachTableCmd

attachTable

Attach tables from an existing database so it can be edited

Syntax:

attachTable('tableNames', 'dataSourceName')

Parameters:

tableNames - Comma-separated list of table names to be attached (blank to prompt user).
dataSourceName - Name of data source from which to attach the tables (blank to prompt user).

Returns:

Void.

Class:

EDMcommon.cmd.AttachTableCmd

callEvaluateExpressionWebMethod

Call the EvaluateExpression web method.

Syntax:

callEvaluateExpressionWebMethod('expression', 'userId', 'encryptedPassword', 'company', 'wsdlUrl')

Parameters:

expression - Expression to evaluate. For example: processTransactions('*')
userId - ID of the user making the call.
encryptedPassword - Encrypted password for the user. Can be found in policy.xml.
company - Name of company where expression is to be evaluated.
wsdlUrl - URL of the WSDL for the web service.

Returns:

Void.

Class:

EDMcommon.cmd.CallEvaluateExpressionWebMethodCmd

cancelEvent

Cancels current event on the active form.

Syntax:

cancelEvent()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.ClientFunctions

changePackageStatus

Change status of the transactions for sites in a package to a new status.

Syntax:

changePackageStatus('oldStatus', 'newStatus', 'packageName', 'siteIdList')

Parameters:

oldStatus - Transactions with this status type will have their status changed (Open, Committed, Sent, Received, Successful, Failed, Closed, or blank to prompt user to select).

newStatus - Transactions will be changed to this status type (Open, Committed, Sent, Received, Successful, Failed, Closed, or blank to prompt user to select).

packageName - Name of package containing transactions to change (blank means prompt user to select).

siteIdList - Comma-delimited list of site IDs whose transactions are to be changed (blank means prompt user to select).

Returns:

Void.

Class:

EDMcommon.cmd.ChangePackageStatusCmd

changePassword

Change the current user's password.

Syntax:

changePassword()

Parameters:

None.

Returns:

Void.

Class:

EDMcommon.cmd.ChangePasswordCmd

clearSessionPackage

Remove the selected session package on the client to users have to select a package when making changes.

Syntax:

clearSessionPackage()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.function.SelectSessionPackageFunction

close

Closes the specified open object window (form view, design view, or print preview).

Syntax:

close('objectType', 'objectName')

Parameters:

objectType - Type of object (table, query, form, report or function).

objectName - Name of the object.

Returns:

Void.

Class:

EDMclient.ClientFunctions

closeActiveWindow

Closes the active window.

Syntax:

closeActiveWindow()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.ClientFunctions

commitTransactions

Commit packages of transactions so they are sent to selected locations

Syntax:

commitTransactions('packageNameList', 'locationIdList', promptToCommit)

Parameters:

packageNameList - Semicolon separated list of package names to commit (if blank, prompt user to select packages)

locationIdList - Semicolon separated list of location numbers to commit packages (if blank, prompt user to select locations for each package)

promptToCommit - If true, ask user if he wants to commit transactions before prompting for locations and packages.

Returns:

Void.

Class:

EDMcommon.cmd.CommitCmd

configureApplicationVersions

Configure known versions of application data sources.

Syntax:

configureApplicationVersions()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.ClientFunctions

copy

Copies an object to the clipboard.

Syntax:

copy('objectType', 'objectName')

Parameters:

objectType - Type of object (table, query, form, report or function).

objectName - Name of the object.

Returns:

Void.

Class:

EDMclient.ClientFunctions

copyAndRefreshSiteData

Copy data for selected tables from one site to other sites and refresh the tables at the other sites on the effective date.

Syntax:

copyAndRefreshSiteData('fromSiteId', 'toSiteIds', 'tableList', overwrite, copySharing, 'effectiveDate', forceProcessing, autoCommit)

Parameters:

fromSiteId - Site id to copy from (blank means prompt user).

toSiteIds - Comma-separated list of site IDs to copy to (blank means prompt user).

tableList - Name of a table list or a comma-separated list of table names to copy (blank means prompt user).

overwrite - True if destination site data should be overwritten if it exists.

copySharing - True if copied tables that are shared by the source site should be shared by the destination site.

effectiveDate - Date/time when data will be copied and refresh transaction sent out (local-specific format).

forceProcessing - True if transaction should be processed when only processing forced transactions.

autoCommit - True if refresh transaction should be committed as soon as it is generated.

Returns:

Void.

Class:

EDMcommon.cmd.CopyAndRefreshSiteDataCmd

copyCompany

Create a new company by copying the configuration of an existing company.

<p>A unique company ID is calculated by removing all characters from the company name except letters, digits, and underscores and adding digits if needed to make the name unique. <p>The configuration files for the company are copied from the sub-directory of the classes directory with the same name as the template company ID. <p>The company database is named EDMHQ_ + companyId. <p>NOTE: The system data source in the config file template must not contain a connect string because it contains the database name which is not replaced. <p>The company's security information (users, roles, etc) will be loaded from the policy file in the default company files directory. <p>There must be a role called 'Company Administrator' in the policy.xml file.

Syntax:

```
copyCompany('companyName', 'userName', 'userId', 'password', 'passwordConfirmation', 'templateCompanyName', 'authorizationClassName', 'authorizationParameters')
```

Parameters:

companyName - Name of the company to create.

userName - User's name.

userId - Login user ID which will be given the 'Company Administrator' role. Null or blank means don't create a user.

password - Login password.

passwordConfirmation - Login password confirmation.

templateCompanyName - Name of the existing company whose configuration will be copied to the new company.

authorizationClassName - Name of the class used to authorize the customer to create a company based on the customer ID.

authorizationParameters - Delimited ascii string of parameters sent to authorization method to determine if the user is authorized to create a company. Each parameter must be surrounded by double quotes, use two double quote characters to represent a double quote within the parameter.

Returns:

Unique ID of new company which can be used for company-specific directory or queue names.

Class:

EDMcommon.cmd.CopyCompanyCmd

copyConfigurationVersionData

Copy the data from one configuration version to another version of the same configuration overwriting any existing data.

Syntax:

```
copyConfigurationVersionData(configurationTypeId, configurationId, sourceVersionId, destinationVersionId)
```

Parameters:

configurationTypeId - ID of the configuration type for which the version data is being copied.

configurationId - ID of the configuration for which the version data is being copied.

sourceVersionId - ID of the version whose data will be copied.

destinationVersionId - ID of the version to which the data will be copied.

Returns:

Site number of the destination version data site.

Class:

EDMclient.plugin.mdm.cmd.CopyConfigVersionDataCmd

copyFoodCostPeriodData

Copy food cost data from one period to another.

Syntax:

copyFoodCostPeriodData('siteIdList', fromFiscalYear, fromPeriodNumber, toFiscalYear, toPeriodNumber)

Parameters:

siteIdList - Comma-separated list of site numbers whose date is to be copied (null, "", or '?' means prompt user to select, '*' means all locations)
fromFiscalYear - Fiscal year to copy from.
fromPeriodNumber - Period number (1-13) to copy from.
toFiscalYear - Fiscal year to copy from.
toPeriodNumber - Period number (1-13) to copy from.

Returns:

Void.

Class:

EDMclient.plugin.UserClientPlugins

copyLocationData

Copy site data for a list of tables to other sites. DEPRECATED Use copySiteData instead.

Syntax:

copyLocationData('fromLocation', 'toLocations', 'tableList')

Parameters:

fromLocation - Id of the location to copy from (blank means prompt user).
toLocations - List of comma-separated location id's of the locations to copy to (blank means prompt user).
tableList - List of comma-separated table names of the tables to whose location data is to be copied

Returns:

Void.

Class:

EDMcommon.cmd.CopySiteDataCmd

copySelectedObject

Copies the application objected selected on the main menu to the clipboard.

Syntax:

copySelectedObject()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.ClientFunctions

copySiteData

Copy data for selected tables from one location to another location

Syntax:

copySiteData('fromSiteId', 'toLocations', 'tableList', overwrite, copySharing)

Parameters:

fromSiteId - Site id to copy from (blank means prompt user).

toLocations - Comma-separated list of site IDs to copy to (blank means prompt user).

tableList - Name of a table list or a comma-separated list of table names to copy (blank means prompt user).

overwrite - True if destination site data should be overwritten if it exists.

copySharing - True if copied tables that are shared by the source site should be shared by the destination site.

Returns:

Void.

Class:

EDMcommon.cmd.CopySiteDataCmd

copySiteSubscriptions

Copy the configuration version subscriptions from one site to other sites leaving the version ID null. If the source site has no subscriptions or the list of configuration ID's to copy is empty, do nothing.

Syntax:

copySiteSubscriptions('sourceSiteId', 'destinationSiteIdList', 'configurationIdList')

Parameters:

sourceSiteId - ID of the site whose subscriptions are to be copied (blank means prompt user).

destinationSiteIdList - Comma-delimited list of destination site IDs that will be subscribed to the same configurations as the source site (blank means prompt user).

configurationIdList - Comma-delimited list of configuration type ID and configuration ID pairs (typeID,configID) for the configurations to be copied. For example, to copy the Eastern configuration (config ID = 2) of the Items configuration type (ID = 1) and the TierA configuration (config ID = 3) of the Prices configuration type (ID = 4), the string would be "1,2,4,3" (Items,Eastern,Prices,TierA) (blank means prompt user).

Returns:

None.

Class:

EDMclient.plugin.mdm.cmd.CopySiteSubscriptionsCmd

copyVersionData

Copy the data from one version to another version for all configurations overwriting any existing data.

Syntax:

copyVersionData('sourceVersionId', 'destinatinVersionId')

Parameters:

sourceVersionId - ID of the version whose data will be copied.

destinatinVersionId - ID of the version to which the data will be copied.

Returns:

None.

Class:

EDMclient.plugin.mdm.cmd.CopyVersionDataCmd

createConfigurationVersionDataSite

Creates a data site to hold version data for configuration version. The ID of the new site will be an unused, negative site number. The name of the site will be the name of the configuration plus a hyphen plus the name of the version. Each table in the configuration type will be added to the list of managed site tables. If there is a previous version of the configuration then the data from the previous version will be copied to the new version.

Syntax:

```
createConfigurationVersionDataSite( configurationTypeId, configurationId, versionId)
```

Parameters:

configurationTypeId - ID of the configuration type for which the version data site is being created.
configurationId - ID of the configuration for which the version data site is being created.
versionId - ID of the version for which the version data site is being created.

Returns:

Site number of the newly created version data site.

Class:

```
EDMclient.plugin.mdm.cmd.CreateConfigVersionDataSiteCmd
```

createConfigurationVersionsCmd

Create configuration versions for the given configuration type and configuration.

Syntax:

```
createConfigurationVersionsCmd( configurationTypeId, configurationId)
```

Parameters:

configurationTypeId - ID of the configuration type.
configurationId - ID of the configuration.

Returns:

Void.

Class:

```
EDMclient.plugin.mdm.cmd.CreateConfigurationVersionsCmd
```

createRemoteInstaller

Create an installation program that can be downloaded and used to install EDM at a remote site with pre-configured settings for the current company.

<p>The function copies CompileInfo.txt from webapps/EDM/company/[companyId](#)/RemoteInstaller to webapps/EDM/WEB-INF/RemoteInstaller then executes webapps/EDM/WEB-INF/RemoteInstaller/build.bat. The CompileInfo.txt file in each company directory contains company-specific information needed to create the remote installation program.</p><p>See the Xpient wiki page "Automated Setup of New Remote Sites" in the "EDM Documentation" section for more information.</p>

Syntax:

```
createRemoteInstaller()
```

Parameters:

None.

Returns:

URL of page where installer can be downloaded.

Class:

EDMcommon.cmd.CreateRemoteInstallerCmd

createTransactionTriggers

Create SQL scripts so triggers on application tables at remote sites can generate EDM transactions.

Creates a SQL script called EDM_Transaction_Functions.sql that creates functions in the EDM database that can be called by triggers in the remote application database to generate transactions. Also creates a script called App_Transaction_Triggers.sql to create triggers on the tables in the given table list that generate transactions in the EDM audit table on insert, update, and delete. Currently this function only supports SQL Server databases.

Syntax:

```
createTransactionTriggers( 'tableListName')
```

Parameters:

tableListName - Name of the table list containing the names of the tables to which triggers should be added.

Returns:

Void.

Class:

EDMcommon.cmd.CreateTransactionTriggersCmd

cut

Cuts an object to the clipboard.

Syntax:

```
cut( 'objectType', 'objectName')
```

Parameters:

objectType - Type of object (table, query, form, report or function).

objectName - Name of the object.

Returns:

Void.

Class:

EDMclient.ClientFunctions

cutSelectedObject

Cuts the application object selected on the main menu to the clipboard.

Syntax:

```
cutSelectedObject()
```

Parameters:

None.

Returns:

Void.

Class:

EDMclient.ClientFunctions

date

Get current date.

Syntax:

date('format')

Parameters:

format - Format string describing how the date should be formatted.

Returns:

String containing current date.

Class:

EDMcommon.function.CommonFunctions

dateToNumber

Converts a date to a number

Syntax:

dateToNumber('date')

Parameters:

date - Date to convert

Returns:

Number representing the date

Class:

EDMcommon.function.CommonFunctions

dateToString

Converts a date to a character string

Syntax:

dateToString('date', 'format')

Parameters:

date - Date to convert

format - Format string to use

Returns:

String representation of the date

Class:

EDMcommon.function.CommonFunctions

dateToTimestamp

Converts a date to a timestamp

Syntax:

dateToTimestamp('date')

Parameters:

date - Date to convert

Returns:

Timestamp representing the date with the time set to 00:00:00

Class:

EDMcommon.function.CommonFunctions

day

Returns the day value of a date

Syntax:

day('date')

Parameters:

date - Date to get the day from

Returns:

Day number (1-31)

Class:

EDMcommon.function.CommonFunctions

delete

Deletes an object.

Syntax:

delete('objectType', 'objectName')

Parameters:

objectType - Type of object (table, query, form, report or function).
objectName - Name of the object.

Returns:

Void.

Class:

EDMclient.ClientFunctions

deleteConfiguration

Delete a configuration with the given configuration type and configuration. Deletes are cascaded to all associated Configuration Versions and DataSites.

Syntax:

```
deleteConfiguration( configurationTypeId, configurationId)
```

Parameters:

configurationTypeId - ID of the configuration type.

configurationId - ID of the configuration.

Returns:

Void.

Class:

EDMclient.plugin.mdm.cmd.DeleteConfigurationCmd

deleteDataSource

Delete the data source with the given name from the list of data sources for this company.

Syntax:

```
deleteDataSource( 'name')
```

Parameters:

name - Data source name used by the system.

Returns:

True if data source was deleted otherwise false.

Class:

EDMclient.ClientFunctions

deleteFoodCostPeriodData

Delete food cost data for one period in a fiscal year.

Syntax:

```
deleteFoodCostPeriodData( 'siteIdList', fiscalYear, periodNumber)
```

Parameters:

siteIdList - Comma-separated list of site numbers whose date is to be deleted (null, "", or '?' means prompt user to select, '*' means all locations)

fiscalYear - Fiscal year to delete.

periodNumber - Period number (1-13) to delete.

Returns:

Void.

Class:

EDMclient.plugin.UserClientPlugins

deleteLocationData

Delete all data from selected tables at selected locations.

Syntax:

deleteLocationData('locationIdList', 'tableList')

Parameters:

locationIdList - Comma-separated list of location id's from which data should be deleted (null, "", or '?' means prompt user to select, '*' means all locations).

tableList - List of table names whose data is to be deleted (null, "", or '?' means prompt user to select).

Returns:

Void

Class:

EDMcommon.cmd.DeleteLocationDataCmd

deleteOrphanedRecords

Delete menus, attachments, prices, and category links that refer to non-existent items.

Syntax:

deleteOrphanedRecords()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.plugin.UserClientPlugins

deleteRecord

Deletes the selected records (or the current record if none selected) on the active form.

Syntax:

deleteRecord()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.ClientFunctions

deleteRow

Delete a row or rows from a table for a given list of sites that match the given columns and values in the table with the given name.

This function can be called in the AfterDelete event to delete a row in the product type table. For example, to delete rows in the TNG_dbo_ProductTypes table after the user inserts a row in the TNG_dbo_Products table, set the `afterInsert` property of the Product form to: `"=addRow('TNG_dbo_ProductTypes', getNamedControlValues('id,name,description'), getSelectedSites())"`

Syntax:

```
deleteRow( 'tableName', 'namedColumnValues', 'siteIdList')
```

Parameters:

tableName - Name of a table.

namedColumnValues - List of column names and values of the form: "name1","value1","name2","value2"...

siteIdList -

Returns:

Void.

Class:

EDMcommon.cmd.DeleteRowCmd

deleteSelectedObject

Deletes the application objected selected on the main menu.

Syntax:

```
deleteSelectedObject()
```

Parameters:

None.

Returns:

Void.

Class:

EDMclient.ClientFunctions

deleteTable

Deletes a table.

Syntax:

```
deleteTable( 'tableName', deletePhysicalTable)
```

Parameters:

tableName - Name of the table.

deletePhysicalTable - True if system should delete the physical table from the database as well.

Returns:

Void.

Class:

EDMclient.ClientFunctions

deleteTableList

Delete a table list.

Syntax:

deleteTableList('tableListName')

Parameters:

tableListName - Name of table list to delete.

Returns:

Deleted table list or null if not found.

Class:

EDMcommon.cmd.EditSnapshotTablesCmd

deleteTransactionFilesFromAmazon

Delete transaction files (*.xml and *.fil) from the Amazon Simple Storage Service (S3).

Syntax:

deleteTransactionFilesFromAmazon('siteIdList', 'bucketName', 'prefix', 'encryptedCredentials')

Parameters:

siteIdList - Comma-separated list of site IDs to delete (blank to prompt user, * for all locations).

bucketName - Name of Amazon S3 bucket containing incoming and outgoing folders.

prefix - Common prefix of all transaction file names to be deleted without the site ID path segment, e.g. "outgoing/".

encryptedCredentials - AWS credentials encrypted with the EncryptAmazonCredentialsCmd object.

Returns:

Void

Class:

EDMcommon.cmd.DeleteTransactionFilesFromAmazonCmd

deployMasterChanges

Deploy master data to application tables in the table map.

Syntax:

deployMasterChanges('deployTypeid', setAppliedToNull, 'versionId', 'appldList')

Parameters:

deployTypeid - ID of the type of deployment being made which determines which site property and value is used to identify sites included in the deployment and the data sources and schemas used for the application tables.

setAppliedToNull - True if the applied date of the subscription records for sites included in the deployment should be set to null to force deployment of all tables.

versionId - ID of the version to be deployed, null or blank means deploy data for all subscribed versions.

appldList - Comma-delimited list of app ID's to deploy or "*" to deploy to all apps.

Returns:

Number of sites that were deployed.

Class:

EDMclient.plugin.mdm.cmd.DeployMasterChangesCmd

design

Opens an object in design view.

Syntax:

design('objectType', 'objectName')

Parameters:

objectType - Type of object (table, query, form, report or function).

objectName - Name of the object.

Returns:

Void.

Class:

EDMcommon.cmd.EditObjectCmd

designForm

Opens a form in design view.

Syntax:

designForm('objectName')

Parameters:

objectName - Name of the form.

Returns:

Void.

Class:

EDMcommon.cmd.EditObjectCmd

designNew

Opens a new object in design view.

Syntax:

designNew('objectType')

Parameters:

objectType - Type of object (table, query, form, report or function).

Returns:

Void.

Class:

EDMclient.ClientFunctions

disableControls

Disable the controls with the given names on the active form.

Syntax:

disableControls('controlNames')

Parameters:

controlNames - Comma-delimited list of control names.

Returns:

None

Class:

EDMclient.function.SetControlPropertyFunction

disableFormControls

Disable the controls with the given names on the given form.

Syntax:

disableFormControls('formName', 'controlNames')

Parameters:

formName - Name of the form.

controlNames - Comma-delimited list of control names.

Returns:

None

Class:

EDMclient.function.SetControlPropertyFunction

displayHelp

Display help for the specified topic.

Syntax:

displayHelp('helpTopic')

Parameters:

helpTopic - Help topic to display.

Returns:

Void.

Class:

EDMclient.ClientFunctions

dumpEmmaCoverageData

Dump Emma coverage data into the specified file

Syntax:

```
dumpEmmaCoverageData( 'fileName')
```

Parameters:

fileName - Name of file to dump Emma coverage data into.

Returns:

Void.

Class:

EDMcommon.cmd.DumpEmmaCoverageDataCmd

dumpProfile

Disable profiling and write profile report to log.

Syntax:

```
dumpProfile()
```

Parameters:

None.

Returns:

Void.

Class:

EDMcommon.cmd.DumpProfileCmd

editCommitListeners

Edit list of email addresses to be notified when packages are committed.

Syntax:

```
editCommitListeners()
```

Parameters:

None.

Returns:

Void.

Class:

EDMclient.ClientFunctions

editConfigurationSubscriptions

Edit Configuration Subscriptions.

Syntax:

editConfigurationSubscriptions()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.plugin.mdm.cmd.EditConfigurationSubscriptionsFunction

editConfigurationTypes

Edit Configuration Types.

Syntax:

editConfigurationTypes()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.plugin.mdm.cmd.EditConfigurationTypesFunction

editConfigurationVersions

Edit Configuration Versions.

Syntax:

editConfigurationVersions()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.plugin.mdm.cmd.EditConfigurationVersionsFunction

editEmailList

Edit a named list of email addresses.

Syntax:

editEmailList('tabTitle', 'listLabel', 'emailListName')

Parameters:

tabTitle - Text to display on the title portion of the client tab.

listLabel - Text to display above the list of rows user can select to edit.

emailListName - Name of the email list (must be valid XML element name).

Returns:

Void.

Class:

EDMclient.ClientFunctions

editIrisConfig

Edit IRIS configuration.

Syntax:

editIrisConfig()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.plugin.iris.EditIrisComboAndShakePricesFunction

editIrisConfigGroups

Edit IRIS config groups.

Syntax:

editIrisConfigGroups()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.plugin.iris.configgroupeditor.ConfigGroupEditorFunction

editIrisItemMenus

Edit IRIS item menus and buttons.

Syntax:

editIrisItemMenus('useMacros', 'range')

Parameters:

useMacros - True (1) if using a version of IRIS that allows buttons to be assigned to menu macros.

range - The range of item menu numbers that the user is allowed to edit. An example would be '1,3,5-10' to exclude item menu numbers 2 and 4 in the range 1-10.

Returns:

Void.

Class:

EDMclient.plugin.iris.EditIrisItemMenusFunction

editIrisMenus

Edit IRIS menus and buttons.

Syntax:

editIrisMenus('useMacros')

Parameters:

useMacros - True (1) if using a version of IRIS that allows buttons to be assigned to menu macros.

Returns:

Void.

Class:

EDMclient.plugin.iris.EditIrisMenusFunction

editIrisTaxRules

Edit IRIS tax rules.

Syntax:

editIrisTaxRules()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.plugin.iris.taxeditor.TaxRuleEditorFunction

EditMobileCategoryItemSequence

Edit Mobile Category Item Display Sequence.

Syntax:

EditMobileCategoryItemSequence()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.plugin.tacobell.mobilecategoryitem.EditMobileCategoryItemSequenceFunction

EditMobileCategorySequence

Edit Mobile Category Display Sequence.

Syntax:

EditMobileCategorySequence()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.plugin.tacobell.mobilecategory.EditMobileCategorySequenceFunction

EditMobileModifierSequence

Edit Mobile Modifier Display Sequence.

Syntax:

EditMobileModifierSequence()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.plugin.tacobell.mobileaddon.EditMobileAddOnSequenceFunction

editRoles

View and modify roles to which system users may be assigned.

Syntax:

editRoles()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.ClientFunctions

editScheduledTasks

Open form to edit tasks scheduled on the server.

Syntax:

editScheduledTasks()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.function.EditScheduledTasksFunction

editServerTextFile

Edit a text file on the server.

Syntax:

editServerTextFile('filePath')

Parameters:

filePath - Path and name of the file to edit. Blank means edit a new empty file.

Returns:

Void.

Class:

EDMclient.function.EditServerTextFileFunction

editSharedTableGroups

Edit groups of tables that are shared together.

Syntax:

editSharedTableGroups()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.ClientFunctions

editSharing

View and modify table sharing among locations.

Syntax:

editSharing()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.ClientFunctions

editUsers

View and modify system users.

Syntax:

editUsers()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.ClientFunctions

editVersions

Edit Versions.

Syntax:

editVersions()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.plugin.mdm.cmd.EditVersionsFunction

editVersionSubscriptions

Edit Version Subscriptions.

Syntax:

```
editVersionSubscriptions()
```

Parameters:

None.

Returns:

Void.

Class:

EDMclient.plugin.mdm.cmd.EditVersionSubscriptionsFunction

enableControls

Enable the controls with the given names on the active form.

Syntax:

```
enableControls( 'controlNames')
```

Parameters:

controlNames - Comma-delimited list of control names.

Returns:

None

Class:

EDMclient.function.SetControlPropertyFunction

enableFormControls

Enable the controls with the given names on the given form.

Syntax:

```
enableFormControls( 'formName', 'controlNames')
```

Parameters:

formName - Name of the form.

controlNames - Comma-delimited list of control names.

Returns:

None

Class:

EDMclient.function.SetControlPropertyFunction

enableProfile

Enable profiling.

Syntax:

enableProfile()

Parameters:

None.

Returns:

True if profiling was previously enable.

Class:

EDMcommon.cmd.EnableProfileCmd

encryptAmazonCredentials

Encrypt a property file containing Amazon Web Services (AWS) credentials.

Syntax:

encryptAmazonCredentials('propertyFile')

Parameters:

propertyFile - Property file containing AWS accessKey and secretKey properties.

Returns:

Void

Class:

EDMcommon.cmd.EncryptAmazonCredentialsCmd

evaluateExpressionAtRemoteSites

Send an expression to a list of remote sites which will evaluate the expression on the server. Only server-side functions may be included in the expression.

Syntax:

evaluateExpressionAtRemoteSites('expression', 'siteIdList')

Parameters:

expression - Expression to evaluate. For example: processTransactions('*')
siteIdList - Comma-delimited list of remote site IDs that are to evaluate the expression.

Returns:

Void.

Class:

EDMclient.function.EvaluateExpressionAtRemoteSitesFunction

executeDirectSQL

Directly execute the given SQL statement on the given data source and return the results in a DataSet. If query is not a select statement, execute the SQL statement and return empty DataSet. Use caution to avoid exceeding available memory.

Syntax:

```
executeDirectSQL( 'dataSourceName', 'sql')
```

Parameters:

dataSourceName - Name of the EDM data source where the query is to execute (null or blank means use system data source).
sql - SQL statement to execute.

Returns:

DataSet

Class:

EDMcommon.cmd.ExecuteQueryCmd

exit

Exits from the program.

Syntax:

```
exit()
```

Parameters:

None.

Returns:

Void.

Class:

EDMclient.ClientFunctions

exportData

Exports data from a table or query to a text file.

Syntax:

```
exportData( 'filename', addNameRow, isDelimited, 'delimiter', 'columnLengths', 'dateFormat', 'source')
```

Parameters:

filename - Name of the text file in which to store the data.
addNameRow - T/F should the first row of the text file contain column names?.
isDelimited - T/F should the text file be delimited (as opposed to fixed length fields)?.
delimiter - Character that separates the fields (Comma, Tab, \ASCII code or character).
columnLengths - Length of each field separated by semicolons ';' for fixed length fields.
dateFormat - Format string that describes how dates should appear in the text file.
source - Name of the table or query to export.

Returns:

Void.

Class:

EDMclient.ClientFunctions

exportFoodCostFiles

Create ascii export files for CKE food cost system

Syntax:

```
exportFoodCostFiles( 'siteIdList', 'fiscalYear', 'periodNumber')
```

Parameters:

siteIdList - Comma-separated list of site numbers whose files are to be created (null, "", or '?' means prompt user to select, '*' means all locations)

fiscalYear - Fiscal year to export (null, "", or '?' means get from Forms:ExportFoodCostFiles:FiscalYear)

periodNumber - Period number (1-13) to export (null, "", or '?' means get from Forms:ExportFoodCostFiles:PeriodNumber)

Returns:

Void.

Class:

EDMclient.plugin.UserClientPlugins

exportSecurityPolicyToXml

Export users, roles, and other security settings to XML file.

Syntax:

```
exportSecurityPolicyToXml( 'file')
```

Parameters:

file - File path and name.

Returns:

Void.

Class:

EDMcommon.cmd.ExportSecurityPolicyToXmlCmd

exportSelectedDataToFile

Exports data from a table or query to a text file. Can be run unattended.

Syntax:

```
exportSelectedDataToFile( 'siteList', useShareGroupNumbers, 'filePathAndName', addNameRow, isDelimited, 'delimiter', 'columnLengths', 'dateFormat', 'source')
```

Parameters:

siteList - Comma delimited list of the sites to be exported.

useShareGroupNumbers - True/False whether or not to use the share group numbers rather than the site list numbers

filePathAndName - Name of the text file in which to store the data.

addNameRow - T/F should the first row of the text file contain column names?.

isDelimited - T/F should the text file be delimited (as opposed to fixed length fields)?.

delimiter - Character that separates the fields (Comma, Tab, \ASCII code or character).

columnLengths - Length of each field separated by semicolons ';' for fixed length fields.

dateFormat - Format string that describes how dates should appear in the text file.

source - Name of the table or query to export.

Returns:

Void.

Class:

EDMclient.ClientFunctions

extractTestDatabaseChanges

Extract changes to a test database to transaction files

Syntax:

extractTestDatabaseChanges('snapshotName', 'packageName', 'packageDescription')

Parameters:

snapshotName - Name of snapshot containing list of tables to refresh (blank means prompt user to select).

packageName - Name of package to hold transactions (blank means prompt user to select).

packageDescription - Description of package if a new package is to be created

Returns:

Void.

Class:

EDMclient.ClientFunctions

findNextRecord

Finds next record on the current form or spreadsheet using current search options.

Syntax:

findNextRecord()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.ClientFunctions

findRecord

Finds record on the current form or spreadsheet.

Syntax:

findRecord('searchString', caseSensitive, allFields, partOfField, forward, checkCurrentRecord)

Parameters:

searchString - String to search for.

caseSensitive - T/F should search be case sensitive.

allFields - T/F search all fields.

partOfField - 0=whole field, 1=beginning of field, 2=any part of field.

forward - T/F search forward.

checkCurrentRecord - T/F check current record.

Returns:

Void.

Class:

EDMclient.ClientFunctions

generateFunctionHelp

Generate HTML help file for a list of PluginLoader classes.

Syntax:

generateFunctionHelp('pluginLoaderClasses', 'destinationHtmlFile', isClassListForClient)

Parameters:

pluginLoaderClasses - Comma-delimited list of PluginLoader class names whose functions should be included in the help file.

destinationHtmlFile - Path and name of HTML help file to generate.

isClassListForClient - True if the functions are in client side classes, else False if they are in server side classes.

Returns:

Void

Class:

EDMcommon.cmd.GenerateFunctionHelpCmd

generateSnapshot

Generate snapshot of data in snapshot tables

Syntax:

generateSnapshot('snapshotName')

Parameters:

snapshotName - Name of the snapshot to generate (blank to prompt user).

Returns:

Void.

Class:

EDMclient.ClientFunctions

generateTestDatabase

Generate a test database with data from selected locations

Syntax:

generateTestDatabase('snapshotName', 'selectedLocations', 'effectiveDate', 'terminationDate', allowMultipleLocations, 'tableShareGroups')

Parameters:

snapshotName - Name of snapshot containing list of tables to refresh (blank means prompt user to select).

selectedLocations - List of comma-separated location ID's of the locations from which to get the test records (blank means prompt user to select).

effectiveDate - Date that changes made to the test database (if any) will be applied to the system database.

terminationDate - Date that changes made to the test database (if any) will be reversed in the system database.

allowMultipleLocations - If the selected locations list is blank, this controls whether or not the user can select multiple locations when prompted.

tableShareGroups - Name of the table whose share groups are to be shown if user is prompted to select locations.

Returns:

Void.

Class:

EDMcommon.cmd.GenerateTestDatabaseCmd

getAppObject

Returns application object with specified type and name or null if not found.

Syntax:

getAppObject('objectType', 'objectName')

Parameters:

objectType - Type of application object (0=table, 1=query, 2=form, 3=report, 4=function).

objectName - Name of application object.

Returns:

Application object or null if not found.

Class:

EDMcommon.cmd.GetAppObjectCmd

getBackgroundProcessViewer

See a window of processes that are running.

Syntax:

getBackgroundProcessViewer()

Parameters:

None.

Returns:

Void

Class:

EDMclient.ClientFunctions

getClientPropertyBoolean

Gets a client property associated with the given property name.

Syntax:

getClientPropertyBoolean('name', defaultValue)

Parameters:

name - Name of the client property.

defaultValue - Default boolean value of the client property.

Returns:

Default value if no property exists with the given name.

Class:

EDMClient.function.SetClientPropertyFunction

getClientPropertyInteger

Gets a client property associated with the given property name.

Syntax:

getClientPropertyInteger('name', defaultValue)

Parameters:

name - Name of the client property.

defaultValue - Default integer value of the client property.

Returns:

Default value if no property exists with the given name.

Class:

EDMClient.function.SetClientPropertyFunction

getClientPropertyString

Gets a client property associated with the given property name.

Syntax:

getClientPropertyString('name', 'defaultValue')

Parameters:

name - Name of the client property.

defaultValue - Default value of the client property.

Returns:

Default value if no property exists with the given name.

Class:

EDMClient.function.SetClientPropertyFunction

getCompanyName

Returns the name of the currently selected company.

Syntax:

getCompanyName()

Parameters:

None.

Returns:

String.

Class:

EDMclient.function.GetCompanyNameFunction

getConfigProperty

Get configuration property value.

Get value of the property in the configuration file with the given case-sensitive name or null if not found.

Syntax:

```
getConfigProperty( 'propertyName')
```

Parameters:

propertyName - Name of the property.

Returns:

Property value or null if not found.

Class:

EDMcommon.cmd.GetConfigPropertyCmd

getConfigPropertyWithDefault

Get configuration property value.

Get value of the property in the configuration file with the given case-sensitive name or the given default value if not found.

Syntax:

```
getConfigPropertyWithDefault( 'propertyName', 'defaultValue')
```

Parameters:

propertyName - Name of the property.

defaultValue - Default value to return if property not found.

Returns:

Property value or the given default value if not found.

Class:

EDMcommon.cmd.GetConfigPropertyCmd

getControlValueRow

Create row with column values from the given list of controls on the active form.

Syntax:

```
getControlValueRow( 'controlNames')
```

Parameters:

controlNames - Comma-delimited list of names of the controls to use for row values.

Returns:

String containing delimited ascii values.

Class:

EDMclient.function.GetControlValueRowFunction

getCurrentUserId

Return the ID of the user that is currently logged in or "" if no user is logged in.

Syntax:

getCurrentUserId()

Parameters:

None.

Returns:

ID of current user or "" if no user logged in.

Class:

EDMclient.ClientFunctions

getEffectiveDate

Get date/time transactions becomes effective for the active form.

Syntax:

getEffectiveDate()

Parameters:

None.

Returns:

Effective date in default system format.

Class:

EDMclient.function.GetEffectiveDateFunction

getEstimatedRowCount

Get estimated number of rows in the data source of the given form. Will use active form is no form name given.

Syntax:

getEstimatedRowCount('formName')

Parameters:

formName - Name of the form.

Returns:

Estimated number of rows.

Class:

EDMclient.function.GetEstimatedRowCountFunction

getForceProcessing

Get flag for whether transactions for the active form should be processed when only processing forced transactions.

Syntax:

getForceProcessing()

Parameters:

None.

Returns:

True if transactions for the active form should be processed when only processing forced transactions.

Class:

EDMclient.function.GetForceProcessingFunction

getFormNamedControlValues

Get a string containing the names and values of a list of controls from a form. Example: "name1","value1","name2","value2",...

Syntax:

getFormNamedControlValues('formName', 'controlNameList')

Parameters:

formName - Name of the form with the controls.

controlNameList - Comma-delimited list of control names whose names and values will be added to the string.

Returns:

String.

Class:

EDMclient.function.ControlValueFunctions

getFormValue

Gets the value of the given control on the given form.

Syntax:

getFormValue('formName', 'controlName')

Parameters:

formName - Name of the form the control is on.

controlName - Control whose value is to be returned.

Returns:

String.

Class:

EDMclient.function.ControlValueFunctions

getIniValue

Returns value for the given key from the given ini file or null if not found.

Syntax:

```
getIniValue( 'iniFileName', 'key')
```

Parameters:

iniFileName - Path and name of ini file. Path may be absolute or relative.
key - Key whose value is to be returned.

Returns:

Ini value or null if not found.

Class:

EDMcommon.cmd.GetIniFileValueCmd

getLastDeploymentInfo

Get the last deployment date and user name with the given deployment type.

Syntax:

```
getLastDeploymentInfo( 'deployTypeid', 'versionId')
```

Parameters:

deployTypeid - The ID of the deployment type, which is used to filter the sites.
versionId - The ID of the version.

Returns:

String.

Class:

EDMclient.plugin.mdm.cmd.GetLastDeploymentInfoCmd

getLastVersionChangeInfo

Get the last change date and user name with the given version.

Syntax:

```
getLastVersionChangeInfo( 'versionId')
```

Parameters:

versionId - The ID of the version.

Returns:

String.

Class:

EDMclient.plugin.mdm.cmd.GetLastVersionChangeInfoCmd

getMasterTableConfigurationType

Gets the Configuration Type for the given table with the given name.

Syntax:

```
getMasterTableConfigurationType( 'tableName')
```

Parameters:

tableName - The name of the table to check.

Returns:

The ConfigurationType for the given table with the given name.

Class:

EDMclient.plugin.mdm.cmd.GetMasterTableConfigurationTypeCmd

getNamedControlValues

Get a string containing the names and values of a list of controls from the active form. Example: "name1","value1","name2","value2",...

Syntax:

```
getNamedControlValues( 'controlNameList')
```

Parameters:

controlNameList - Comma-delimited list of control names whose names and values will be added to the string.

Returns:

String.

Class:

EDMclient.function.ControlValueFunctions

getNextUnusedId

Gets the next unused ID for the given table and column, optionally within a range.

Syntax:

```
getNextUnusedId( 'tableName', 'columnName', 'minValue', 'maxValue')
```

Parameters:

tableName - The name of the table containing the column.

columnName - The name of the column to return the next ID value for.

minValue - The minimum value to return. If not supplied, will use 1.

maxValue - The maximum value to return. If not supplied, will use Integer.MAX_VALUE.

Returns:

Integer.

Class:

EDMcommon.cmd.GetNextUnusedIdCmd

getNumberOfAssignedSites

Get the number of sites assigned to the given version.

Syntax:

```
getNumberOfAssignedSites( 'deployTypeId', 'versionId')
```

Parameters:

deployTypeId - The ID of the deployment type, which is used to filter the sites.
versionId - The ID of the version.

Returns:

Integer.

Class:

EDMclient.plugin.mdm.cmd.GetNumberOfAssignedSitesCmd

getNumberOfUnassignedSites

Get the number of sites that are not assigned to the given version.

Syntax:

```
getNumberOfUnassignedSites( 'deployTypeId', 'versionId')
```

Parameters:

deployTypeId - The ID of the deployment type, which is used to filter the sites.
versionId - The ID of the version.

Returns:

Integer.

Class:

EDMclient.plugin.mdm.cmd.GetNumberOfUnassignedSitesCmd

getPackageDescription

Get description of package that contains transaction for the active form.

Syntax:

```
getPackageDescription()
```

Parameters:

None.

Returns:

Package description.

Class:

EDMclient.function.GetPackageDescriptionFunction

getPackageName

Get name of package that contains transaction for the active form.

Syntax:

getPackageName()

Parameters:

None.

Returns:

Package name.

Class:

EDMclient.function.GetPackageNameFunction

getPropertyFileValue

Get the value of a property in a property file.

Syntax:

getPropertyFileValue('propertyName', 'propertyFileName')

Parameters:

propertyName - Name of the property whose value is to be returned.

propertyFileName - Path and name of property file.

Returns:

Property value

Class:

EDMcommon.cmd.GetPropertyFileValueCmd

getSchemaVersions

Returns a list of known application database schemas supported by the system.

Syntax:

getSchemaVersions()

Parameters:

None.

Returns:

List of known application database schemas.

Class:

EDMclient.ClientFunctions

getSelectedSites

Get the selected sites for the active form.

Syntax:

getSelectedSites()

Parameters:

None.

Returns:

Comma delimited list of selected site IDs for the active form.

Class:

EDMclient.function.GetSelectedSitesFunction

getServerCallsPerSecond

Returns the average number of calls per second that can be made to the server after sending as many calls to the server as possible for 5 seconds.

Syntax:

getServerCallsPerSecond()

Parameters:

None.

Returns:

Integer.

Class:

EDMcommon.cmd.GetServerTimeCmd

getServerFile

Return the bytes of a file from the server.

Syntax:

getServerFile('filename')

Parameters:

filename - Name and relative path of file to be returned.

Returns:

Bytes in the specified file from the server.

Class:

EDMclient.ClientFunctions

getServerFileList

Gets a list of files in the specified directory on the server.

Syntax:

getServerFileList('directory')

Parameters:

directory - Directory on the server (relative to application base directory) whose files are to be returned.

Returns:

Array of objects containing file attributes.

Class:

EDMclient.ClientFunctions

getServerTime

Returns the current system time on the server in milliseconds since 1/1/1970 UTC.

Syntax:

getServerTime()

Parameters:

None.

Returns:

Integer.

Class:

EDMcommon.cmd.GetServerTimeCmd

getSessionPackageName

Returns the name of the currently selected package.

Syntax:

getSessionPackageName()

Parameters:

None.

Returns:

String.

Class:

EDMclient.function.GetSessionPackageNameFunction

getSessionPackageVersionID

Returns the ID of the currently selected package's version or null if there is no selected package or if the selected package doesn't have a version.

Syntax:

getSessionPackageVersionID()

Parameters:

None.

Returns:

Integer.

Class:

EDMclient.function.GetSessionPackageNameFunction

getSessionPackageName

Returns the name of the currently selected package's version or null if there is no selected package or if the selected package doesn't have a version.

Syntax:

getSessionPackageName()

Parameters:

None.

Returns:

String.

Class:

EDMclient.function.GetSessionPackageNameFunction

getSessions

Returns a list of the current active sessions on the server.

Syntax:

getSessions()

Parameters:

None.

Returns:

String.

Class:

EDMcommon.cmd.GetSessionsCmd

getSystemProperty

Get the value of a property in the system.properties file.

Syntax:

getSystemProperty('propertyName', 'DefaultValue')

Parameters:

propertyName - Name of the property.

DefaultValue - Value to return if the property is not found.

Returns:

Value of system property if found, otherwise the default value.

Class:

EDMclient.ClientFunctions

getTableAndQueryList

Get a sorted list of tables and queries in this application separated by semicolons.

Syntax:

getTableAndQueryList()

Parameters:

None.

Returns:

Sorted list of tables and queries separated by semicolons.

Class:

EDMclient.ClientFunctions

getTableList

Get a sorted list of tables in this application separated by semicolons.

Syntax:

getTableList()

Parameters:

None.

Returns:

Sorted list of tables and queries separated by semicolons.

Class:

EDMclient.ClientFunctions

getTerminationDate

Get date/time transactions terminate for the active form.

Syntax:

getTerminationDate()

Parameters:

None.

Returns:

Termination date in default system format.

Class:

EDMclient.function.GetTerminationDateFunction

getText

Returns the text from the given control.

Syntax:

getText('controlName')

Parameters:

controlName - Name of the control.

Returns:

Text from the given control.

Class:

EDMclient.function.ControlValueFunctions

getTransactionViewer

See a dialog of recent file transactions that are either in yellow or red status.

Syntax:

getTransactionViewer()

Parameters:

None.

Returns:

Void

Class:

EDMclient.ClientFunctions

getUnupdatableSites

Return comma-delimited list of unupdatable sites for the active form.

Syntax:

getUnupdatableSites()

Parameters:

None.

Returns:

List of sites.

Class:

EDMclient.ClientFunctions

getUpdatableSites

Return comma-delimited list of updatable sites for the active form.

Syntax:

```
getUpdatableSites()
```

Parameters:

None.

Returns:

List of sites.

Class:

EDMclient.ClientFunctions

getUserInput

Prompts user to enter a value.

Syntax:

```
getUserInput( 'title', 'prompt', 'defaultValue', password)
```

Parameters:

title - Dialog box title.

prompt - Prompt text to tell user what to enter.

defaultValue - Default value in the input field.

password - T/F is this a password field that should not echo the characters entered.

Returns:

Value entered by the user or blank if cancelled.

Class:

EDMclient.ClientFunctions

getValue

Gets data for the specified identifier.

Syntax:

```
getValue( 'identifier')
```

Parameters:

identifier - Name of the identifier.

Returns:

Identifier value if successful, otherwise null.

Class:

EDMclient.function.ControlValueFunctions

getValueListOfUsersWithRole

Get a value list of users with a given role separated by semicolons.

Syntax:

```
getValueListOfUsersWithRole( 'roleName')
```

Parameters:

roleName - Name of role.

Returns:

String.

Class:

EDMcommon.cmd.GetValueListOfUsersWithRoleCmd

getVersion

Return the current version of EDM

Syntax:

```
getVersion()
```

Parameters:

None.

Returns:

The current version of EDM

Class:

EDMcommon.function.CommonFunctions

gotoControl

Moves cursor to a control.

Syntax:

```
gotoControl( 'controlName')
```

Parameters:

controlName - Control to move to.

Returns:

Void.

Class:

EDMclient.ClientFunctions

gotoFirstRecord

Moves cursor to the first record on the active form.

Syntax:

gotoFirstRecord()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.ClientFunctions

gotoFormControl

Moves cursor to a control.

Syntax:

gotoFormControl('formName', 'controlName')

Parameters:

formName - Form containing control to move to.

controlName - Control to move to.

Returns:

Void.

Class:

EDMclient.ClientFunctions

gotoLastRecord

Moves cursor to the last record on the active form.

Syntax:

gotoLastRecord()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.ClientFunctions

gotoNewRecord

Moves cursor to the new empty record on the active form.

Syntax:

gotoNewRecord()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.ClientFunctions

gotoNextRecord

Moves cursor to the next record on the active form.

Syntax:

gotoNextRecord()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.ClientFunctions

gotoPrevRecord

Moves cursor to the previous record on the active form.

Syntax:

gotoPrevRecord()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.ClientFunctions

gotoRecord

Moves cursor to a specific record on the active form.

Syntax:

gotoRecord(recordNumber)

Parameters:

recordNumber - Record number to go to.

Returns:

Void.

Class:

EDMclient.ClientFunctions

greatest

Returns the greater of two numbers

Syntax:

greatest('firstNumber', 'secondNumber')

Parameters:

firstNumber - First number to compare

secondNumber - Second number to compare

Returns:

Greater of two numbers

Class:

EDMcommon.function.CommonFunctions

hasPermission

Returns true if the current user has the given permission type for the given object type and name.

Syntax:

hasPermission('permissionTypes', 'objectType', 'objectName')

Parameters:

permissionTypes - Comma-delimited list of permission types: VIEW, ADD, EDIT, DELETE, and/or EXECUTE.

objectType - Object type: function, data, company, etc.

objectName - Name of the object.

Returns:

Boolean.

Class:

EDMclient.function.HasPermissionFunction

iif

Returns one of two values bases on expression

Syntax:

iif('expression', 'trueValue', 'falseValue')

Parameters:

expression - Expression to evaluate

trueValue - Value to return if the expression is TRUE
falseValue - Value to return if the expression is FALSE

Returns:

First value if expression is true, otherwise second value

Class:

EDMcommon.function.CommonFunctions

importData

Imports a text file to a table.

Syntax:

```
importData( 'filename', HasNameRow, isDelimited, 'delimiter', 'columnLengths', 'dateFormat', 'tableName')
```

Parameters:

filename - Name of the text file to import.
HasNameRow - T/F does the first row of the text file contain column names?.
isDelimited - T/F is the text file delimited (as opposed to fixed length fields)?.
delimiter - Character that separates the fields (Comma, Tab, \ASCII code or character).
columnLengths - Length of each field separated by semicolons ';' for fixed length fields.
dateFormat - Format string that describes how dates appear in the text file.
tableName - Name of the table to which the data is to be added.

Returns:

Void.

Class:

EDMclient.ClientFunctions

importFoodCostProducts

Import ascii file containing products for the food cost system.

Syntax:

```
importFoodCostProducts( 'siteId', 'fileName')
```

Parameters:

siteId - Id of the site to where data should be imported. If not a valid integer, user will be prompted to select a site.
fileName - Name of the ascii file to import. Null or blank means prompt user to enter file name.

Returns:

Void.

Class:

EDMclient.plugin.UserClientPlugins

importSitesInFileList

Import sites listed in given siteIdFile from given company.

Syntax:

importSitesInFileList('fromCompanyName', 'siteIdFileName', 'transactionVersion', 'shareGroupId', 'shareTableList')

Parameters:

fromCompanyName - Name of company from which sites are imported.

siteIdFileName - Name of text file containing one or more rows of comm-delimited site IDs to import.

transactionVersion - Transaction version to use on imported live sites (version > 0), if blank leave unchanged.

shareGroupId - Id of share group whose tables are shared with all imported sites, null or blank means do not share.

shareTableList - Name of table list from snapshot.xml file with list of table shareGroup should share with imported site. Null or blank means do not share.

Returns:

Void.

Class:

EDMclient.ClientFunctions

importTableRelations

Import table relations from a text file

Syntax:

importTableRelations('filename')

Parameters:

filename - Name of text file to import

Returns:

Void.

Class:

EDMclient.ClientFunctions

importTransactionsFromFile

Import transactions from a CSV file.

Syntax:

importTransactionsFromFile('filename')

Parameters:

filename - Name of the CSV file on the server containing transaction information.

Returns:

Void.

Class:

EDMcommon.cmd.ImportTransactionsFromFileCmd

insertTableList

Insert a new table list.

Syntax:

insertTableList(tableList, index)

Parameters:

tableList - New table list to insert.

index - 0-based index where table list should be inserted among other table lists, if index is < 0 or >= list size, the new table list is added to the end of the list.

Returns:

Void.

Class:

EDMcommon.cmd.EditSnapshotTablesCmd

isOnNewRow

True if the currently active form is on a new row, else False.

Syntax:

isOnNewRow()

Parameters:

None.

Returns:

Boolean.

Class:

EDMclient.function.IsOnNewRowFunction

least

Returns the lesser of two numbers

Syntax:

least('First number', 'Second number')

Parameters:

First number - First number to compare

Second number - Second number to compare

Returns:

Lesser of two numbers

Class:

EDMcommon.function.CommonFunctions

left

Returns the leftmost characters of a string

Syntax:

left('String', Count)

Parameters:

String - String to get the characters from

Count - Number of characters to get

Returns:

Requested characters

Class:

EDMcommon.function.CommonFunctions

len

Gets the length of a string

Syntax:

len('String')

Parameters:

String - String you want the length of

Returns:

Length of the string

Class:

EDMcommon.function.CommonFunctions

loadAllTableData

Load data for the specified location from all tables in the table list

Syntax:

loadAllTableData('Location List')

Parameters:

Location List - List of comma-separated location id's of the locations whose data is to be deleted

Returns:

Void.

Class:

EDMclient.ClientFunctions

loadCentralData

Load schema for a central database table

Syntax:

loadCentralData('tableNames', 'locationNumber', 'dataSourceName')

Parameters:

tableNames - Comma-separated list of table names whose data is to be loaded (blank means prompt user, * means all).

locationNumber - Location number of the location whose data is being loaded (blank means prompt user).

dataSourceName - Name of data source from which to load table data (blank means prompt user).

Returns:

Void.

Class:

EDMclient.ClientFunctions

loadCentralSchema

Load schema for a central database table

Syntax:

loadCentralSchema('tableName', 'dataSourceName', 'tableVersion', 'override', 'ignoreInserts', 'ignoreUpdates', 'ignoreDeletes')

Parameters:

tableName - Name of table whose schema is to be loaded (blank to prompt user)

dataSourceName - Name of ODBC data source from which to load table schema (blank to prompt user)

tableVersion - Version number of table (blank to prompt user)

override - Flag for handling conflicts such as 'Log' or 'Undo' (blank to prompt user)

ignoreInserts - True if central database should not accept inserts from remote location (blank to prompt user).

ignoreUpdates - True if central database should not accept updates from remote location (blank to prompt user).

ignoreDeletes - True if central database should not accept deletes from remote location (blank to prompt user).

Returns:

Void.

Class:

EDMcommon.cmd.AttachTableCmd

loadPicture

Loads a GIF, JPEG, or PNG image from the specified file

Syntax:

loadPicture('filename')

Parameters:

filename - Name of the GIF, JPEG, or PNG picture file to load

Returns:

String of hex digits representing the picture

Class:

EDMclient.ClientFunctions

login

Log a user into the system.

Syntax:

login('userid', 'password')

Parameters:

userid - Unique user id (blank means prompt user).
password - Password for the userid.

Returns:

Void.

Class:

EDMclient.ClientFunctions

logout

Logs the current user out of the system.

Syntax:

logout()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.ClientFunctions

lower

Changes each letter of a string to lower case

Syntax:

lower('string')

Parameters:

string - String to convert

Returns:

String converted to lower case

Class:

EDMcommon.function.CommonFunctions

maximizeContentArea

Maximizes the content area in the main EDM window.

Syntax:

maximizeContentArea()

Parameters:

None.

Returns:

Void.

Class:

EDMcommon.cmd.ToggleContentAreaSizeFunction

mod

Divides two values and returns the remainder

Syntax:

mod('firstValue', 'secondValue')

Parameters:

firstValue - Value to divide

secondValue - Value to divide by

Returns:

Remainder after division.

Class:

EDMcommon.function.CommonFunctions

month

Returns the month value of a date

Syntax:

month('date')

Parameters:

date - Date to get the month from (local-specific format)

Returns:

Month number (1-12)

Class:

EDMcommon.function.CommonFunctions

moveTransactions

Moves transactions for the specified locations from the local send directory to the specified directory.

Syntax:

moveTransactions('locationIdList', 'toDirectory')

Parameters:

locationIdList - Comma-separated list of locations whose transactions are to be moved (blank means prompt user).
toDirectory - Directory to which transactions should be moved (blank means prompt user).

Returns:

Void.

Class:

EDMclient.plugin.UserClientPlugins

msg

Displays a message box.

Syntax:

msg('text', 'title', flags)

Parameters:

text - Message text.

title - Message dialog title.

flags - Sum of button codes to display (Ok=1, Yes=2, Cancel=4, No=8).

Returns:

Value of button user selected to close message box.

Class:

EDMclient.ClientFunctions

notifyClients

Send a notification to connected clients.

Syntax:

notifyClients('title', 'message', 'delayMinutes')

Parameters:

title - Title of message to display to client (blank or ? means prompt user).

message - Message to display to client (blank or ? means prompt user).

delayMinutes - How many minutes to delay before sending message to connected clients.

Returns:

Void.

Class:

EDMcommon.cmd.NotifyClientsCmd

nullValue

Return a null value which can be used in other functions. If you want to set the value of a form control to null, use setFormValueToNull() or setValueToNull().

Syntax:

nullValue()

Parameters:

None.

Returns:

Null.

Class:

EDMcommon.function.NullValueFunction

numberToDate

Converts a number to a date

Syntax:

numberToDate('number')

Parameters:

number - Number to convert

Returns:

Date value

Class:

EDMcommon.function.CommonFunctions

numberToInt

Returns the integer part of a number string.

Syntax:

numberToInt('number')

Parameters:

number - Number string to convert

Returns:

Integer value.

Class:

EDMcommon.function.CommonFunctions

numberToString

Converts a number to a string

Syntax:

numberToString('number', 'Format')

Parameters:

number - Number to convert
Format - Format string to use

Returns:

String representation of the number

Class:

EDMcommon.function.CommonFunctions

numberToTime

Converts a number to a time

Syntax:

numberToTime('number')

Parameters:

number - Number to convert

Returns:

Time value

Class:

EDMcommon.function.CommonFunctions

numberToTimestamp

Converts a number to a timestamp

Syntax:

numberToTimestamp('number')

Parameters:

number - Number to convert

Returns:

Timestamp value

Class:

EDMcommon.function.CommonFunctions

open

Opens an object.

Syntax:

open('objectType', 'objectName')

Parameters:

objectType - Type of object (table, query, form, or function).

objectName - Name of the object.

Returns:

Void.

Class:

EDMcommon.cmd.GetOpenFormCmd

openForm

Ask server to open a form. All parameters must be complete, user will be prompted for routing information if necessary.

Syntax:

openForm('name', 'view', visible)

Parameters:

name - Name of form.

view - View to display, 'SINGLE' (form view), 'GRID' (spreadsheet view) or " (default view).

visible - True if form is visible after opening, otherwise false.

Returns:

Void.

Class:

EDMcommon.cmd.GetOpenFormCmd

openRowSetForm

Open form to edit row sets to restrict which table rows and columns users may edit.

Syntax:

openRowSetForm()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.function.OpenRowSetFormFunction

openSitePropertyValueEditor

Open the Site Property Values form.

Syntax:

openSitePropertyValueEditor()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.ClientFunctions

partitionTable

Partition a table and set a new partition expression and reorganize the rows into the correct partitions.

The audit table can now be partitioned across multiple database servers which process queries in parallel for much faster performance. Transaction records are divided among the partitions based on a partitioning expression. Queries on the audit table are sent to each partition server for parallel processing and the results are merged by EDM so that the partitioning is transparent to EDM users. The primary partition will continue to reside in the EDM central database, additional partitions can be stored on other servers.

To partition the audit table into three parts take the following steps:

- Backup your central database and the web application folder (webapps/EDM)
- Add new data sources named 'Partition2' and 'Partition3' to the config.xml with the connection information to the new partition servers.
- Restart the web server or reload the cached data so the server will recognize the new data sources.
- Add a menu option with the action: PartitionTable('CDMAudit', 'Partition2,Partition3', 'iif(FromLoc = 0, ToLoc % 3, FromLoc % 3)')
- Click on the new menu option and watch the progress of the partitioning on the status bar.

After partitioning the audit table, if you have regular backups of your system database, you should include the new partition databases in your backup schedule.

Syntax:

partitionTable('tableName', 'partitionDataSourceNames', 'partitionExpression')

Parameters:

tableName - Name of the table which is to have a new partition added.

partitionDataSourceNames - Comma-delimited list of data source names where the table partitions are to be created.

partitionExpression - Expression used to determine which partition a row is in by calculating a number between 0 and the number of partition data source names. For example, if you specify 1 partition data source name 'Part1' and want rows with even CDMLOCID's in the primary table and rows with odd CDMLOCID's in the partition then the expression would be 'CDMLOCID % 2' which will return 0 for even CDMLOCID's and 1 for odd CDMLOCID's. If the expression returns a number higher than the number of partition data sources for a row, then the row will be in the main table.

Returns:

null

Class:

EDMcommon.cmd.PartitionTableCmd

paste

Pastes an object from the clipboard.

Syntax:

paste('objectType', 'objectName')

Parameters:

objectType - Type of object (table, query, form, report or function).

objectName - Name of the object.

Returns:

Void.

Class:

EDMclient.ClientFunctions

playbackTestScript

Play a test script back.

Syntax:

playbackTestScript('name', 'serverURL', concurrentUsers, secondsBetweenStarts)

Parameters:

name - Name of the test script directory.

serverURL - URL of the server where commands are sent.

concurrentUsers - Number of concurrent instances of the test to run.

secondsBetweenStarts - Number of seconds between starting each concurrent test instance.

Returns:

Void.

Class:

EDMclient.ClientFunctions

pow

Raises a number to a power

Syntax:

pow('number', power)

Parameters:

number - Number to raise to a power

power - Power to raise the number to

Returns:

Number raised to a power

Class:

EDMcommon.function.CommonFunctions

processForcedTransactions

Receive and apply pending transactions whose effective date has arrived and whose Force flag is true for all sites.

Syntax:

processForcedTransactions()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.function.ProcessForcedTransactionsFunction

processTransactions

Receive and apply pending transactions from selected locations whose effective date has arrived

Syntax:

processTransactions('locations')

Parameters:

locations - Comma-separated list of location id numbers (blank means prompt user, * means all locations).

Returns:

Void.

Class:

EDMcommon.cmd.ProcessTransactionsCmd

promptForTransactionReport

Display dialog for users to set filters and run transaction report.

Syntax:

promptForTransactionReport()

Parameters:

None.

Returns:

Void.

Class:

EDMcommon.cmd.CreateTransactionReportCmd

publishMasterDataTableList

Publish data from the central database, with open and future transactions rolled back, to .csv files in the given directory.

Syntax:

publishMasterDataTableList('tableName', 'siteIdList', 'directory')

Parameters:

tableName - Name of table list from snapshot file. (blank means prompt user to select).

siteIdList - Comma-separated list of site IDs whose data is to be published (blank means prompt user to select).

directory - Directory where published data files are to be stored.

Returns:

Void.

Class:

EDMclient.plugin.mdm.cmd.PublishMasterDataCmd

publishMasterDataTables

Publish data from the central database, with open and future transactions rolled back, to .csv files in the given directory.

Syntax:

publishMasterDataTables('tableNameList', 'siteIdList', 'directory')

Parameters:

tableNameList - Comma-separated list of table names to publish (blank means prompt user to select, '*' means all).
siteIdList - Comma-separated list of site IDs whose data is to be published (blank means prompt user to select).
directory - Directory where published data files are to be stored.

Returns:

Void.

Class:

EDMclient.plugin.mdm.cmd.PublishMasterDataCmd

publishXCEDDataTableList

Publish data from the central database, with open and future transactions rolled back, to .csv files in the given directory.

Syntax:

publishXCEDDataTableList('tableListName', 'siteIdList', 'directory')

Parameters:

tableListName - Name of table list from snapshot file. (blank means prompt user to select).
siteIdList - Comma-separated list of site IDs whose data is to be published (blank means prompt user to select).
directory - Directory where published data files are to be stored.

Returns:

Void.

Class:

EDMclient.plugin.mdm.cmd.PublishXCEDDataCmd

publishXCEDDataTables

Publish data from the central database, with open and future transactions rolled back, to .csv files in the given directory.

Syntax:

publishXCEDDataTables('tableNameList', 'siteIdList', 'directory')

Parameters:

tableNameList - Comma-separated list of table names to publish (blank means prompt user to select, '*' means all).
siteIdList - Comma-separated list of site IDs whose data is to be published (blank means prompt user to select).
directory - Directory where published data files are to be stored.

Returns:

Void.

Class:

EDMclient.plugin.mdm.cmd.PublishXCEDDataCmd

queryAvg

Gets the average of the values of a column in a table or query using a where clause.

Syntax:

```
queryAvg( 'column', 'table', 'where')
```

Parameters:

column - Name of the column to get.

table - Name of the table containing the column.

where - Where clause to filter the rows in the table.

Returns:

Value of field if successful, otherwise empty string.

Class:

EDMcommon.cmd.QueryValueCmd

queryCount

Gets the count of the values of a column in a table or query using a where clause.

Syntax:

```
queryCount( 'column', 'table', 'where')
```

Parameters:

column - Name of the column to get.

table - Name of the table containing the column.

where - Where clause to filter the rows in the table.

Returns:

Value of field if successful, otherwise empty string.

Class:

EDMcommon.cmd.QueryValueCmd

queryFirst

Gets the first value of a column in a table or query using a where clause.

Syntax:

```
queryFirst( 'column', 'table', 'where')
```

Parameters:

column - Name of the column to get.

table - Name of the table containing the column.

where - Where clause to filter the rows in the table.

Returns:

Value of field if successful, otherwise empty string.

Class:

EDMcommon.cmd.QueryValueCmd

queryLast

Gets the last value of a column in a table or query using a where clause.

Syntax:

```
queryLast( 'column', 'table', 'where')
```

Parameters:

column - Name of the column to get.

table - Name of the table containing the column.

where - Where clause to filter the rows in the table.

Returns:

Value of field if successful, otherwise empty string.

Class:

EDMcommon.cmd.QueryValueCmd

queryMax

Gets the maximum value of a column in a table or query using a where clause.

Syntax:

```
queryMax( 'column', 'table', 'where')
```

Parameters:

column - Name of the column to get.

table - Name of the table containing the column.

where - Where clause to filter the rows in the table.

Returns:

Value of field if successful, otherwise empty string.

Class:

EDMcommon.cmd.QueryValueCmd

queryMin

Gets the minimum value of a column in a table or query using a where clause.

Syntax:

```
queryMin( 'column', 'table', 'where')
```

Parameters:

column - Name of the column to get.

table - Name of the table containing the column.

where - Where clause to filter the rows in the table.

Returns:

Value of field if successful, otherwise empty string.

Class:

EDMcommon.cmd.QueryValueCmd

querySum

Gets the sum of the values of a column in a table or query using a where clause.

Syntax:

```
querySum( 'column', 'table', 'where')
```

Parameters:

column - Name of the column to get.

table - Name of the table containing the column.

where - Where clause to filter the rows in the table.

Returns:

Value of field if successful, otherwise empty string.

Class:

EDMcommon.cmd.QueryValueCmd

queryValue

Directly execute an SQL statement and return the value of the first column in the first row. If query is not a select statement, execute the SQL statement and return a blank string. Use caution to avoid exceeding available memory.

Syntax:

```
queryValue( 'dataSourceName', 'sql')
```

Parameters:

dataSourceName - Name of the EDM data source where the query is to execute.

sql - SQL statement to execute.

Returns:

String

Class:

EDMclient.ClientFunctions

random

Returns a random number between 0 and the passed number

Syntax:

```
random( max)
```

Parameters:

max - Maximum random number that can be returned

Returns:

Random number

Class:

EDMcommon.function.CommonFunctions

receiveSiteFilesViaWeb

Receive transaction files over HTTP web connection.

Syntax:

```
receiveSiteFilesViaWeb( 'siteIdList')
```

Parameters:

siteIdList - Comma-separated list of sites id's to transfer (blank to prompt user, * for all sites).

Returns:

Void.

Class:

EDMcommon.cmd.TransferFilesCmd

receiveTransactionFilesFromAmazon

Receive transaction files from Amazon Simple Storage Service (S3).

Syntax:

```
receiveTransactionFilesFromAmazon( 'siteIdList', 'bucketName', 'prefix', 'encryptedCredentials')
```

Parameters:

siteIdList - Comma-separated list of site IDs to receive (blank to prompt user, * for all locations).

bucketName - Name of Amazon S3 bucket containing incoming and outgoing folders.

prefix - Common prefix of all transaction file names to be transferred without the site ID path segment, e.g. "outgoing".

encryptedCredentials - AWS credentials encrypted with the EncryptAmazonCredentialsCmd object.

Returns:

Void

Class:

EDMcommon.cmd.ReceiveTransactionFilesFromAmazonCmd

receiveTransactionFilesFromJMS

Receive files from the configured JMS queue into the local incoming folder.

Syntax:

```
receiveTransactionFilesFromJMS( 'siteIdList', 'queueName', 'brokerURL', 'userId', 'password')
```

Parameters:

siteIdList - Comma-separated list of site IDs to receive (blank to prompt user, * for all locations).

queueName - Name of the queue from which incoming transaction files should be received. For example: EDM.incoming.%REMOTESITEID%

brokerURL - URL used to connect to the JMS broker. For example: tcp://localhost:61616

userId - User ID used to connect to the JMS broker.

password - Password used to connect to the JMS broker.

Returns:

Void

Class:

EDMcommon.cmd.ReceiveTransactionFilesFromJMScmd

reloadCachedData

Reload cached data such as configuration, site list, etc.

Syntax:

```
reloadCachedData()
```

Parameters:

None.

Returns:

Void.

Class:

EDMcommon.cmd.ReloadCachedDataCmd

rename

Renames an object to a new name.

Syntax:

```
rename( 'objectType', 'objectName', 'newName')
```

Parameters:

objectType - Type of object (table, query, form, report or function).

objectName - Name of the object (case-insensitive).

newName - New name of the object (blank means prompt user).

Returns:

Void.

Class:

EDMclient.ClientFunctions

replaceAll

Replaces each substring of this string that matches the given regular expression with the given replacement.

Note that backslashes (\) and dollar signs (\$) in the replacement string may cause the results to be different than if it were being treated as a literal replacement string.

Syntax:

```
replaceAll( 'string', 'regularExpression', 'replacement')
```

Parameters:

string - String to which replacements are made.

regularExpression - Regular expression to which this string is to be matched.

replacement - String to be substituted for each match

Returns:

String with each match replaced with the replacement string or null if the given string is null.

Class:

EDMcommon.function.CommonFunctions

replaceTransWithRefresh

Replaces transaction files for the selected locations with a refresh table transaction using the specified snapshot.

Syntax:

replaceTransWithRefresh('locationIdList', 'snapshotName')

Parameters:

locationIdList - Comma-separated list of locations whose transactions are to be replaced with the specified snapshot.

requestDatedTableRefresh

Send request to location to send table data to replace existing data

Syntax:

```
requestDatedTableRefresh('snapshotName', 'selectedLocations', forceProcessing, 'packageName', 'packageDescription', 'rollbackType', 'rollbackDate', 'rollbackTime')
```

Parameters:

snapshotName - Name of the snapshot containing the list of tables to refresh (blank means prompt user to select).

selectedLocations - List of comma-separated location ID's of the locations from which to get the test records (blank means prompt user to select).

forceProcessing - True if transactions should be processed when only processing forced transactions.

packageName - Name of package to hold transactions (blank means prompt user to select).

packageDescription - Description of package if a new package is to be created

rollbackType - Type of transaction roll back: 'NONE', 'NOW', or 'SPECIFIED'. Prompt user if blank.

rollbackDate - If rollbackType is 'SPECIFIED', date to which transactions should be rolled back, format is mm/dd/yyyy.

rollbackTime - If rollbackType is 'SPECIFIED', time to which transactions should be rolled back, format is hh:mm:ss.SSS AM.

Returns:

Void.

Class:

EDMcommon.cmd.TableRefreshCmd

requestFile

Copy a file to other locations

Syntax:

```
requestFile('locationList', 'fromFile', 'toFile', flags, 'effectiveDate', 'packageName', 'packageDescription')
```

Parameters:

locationList - Comma-separated list of location numbers from which to request file (blank means prompt user, * means all locations).

fromFile - Full path of file to copy (if blank, prompt user to select)

toFile - Full path of destination file at other locations (if blank, prompt user to select)

flags - 1 if file should be overwritten if it exists at destination location; 2 if file should be kept synchronized after being sent.

effectiveDate - Date file should be copied to destination

packageName - Name of package to hold transactions

packageDescription - Description of package if new package.

Returns:

Void.

Class:

EDMcommon.cmd.WriteFileTransactionCmd

requestTableRefresh

Send request to location to send table data to replace existing data

Syntax:

```
requestTableRefresh('snapshotName', 'selectedLocations', 'packageName', 'packageDescription')
```

Parameters:

snapshotName - Name of the snapshot containing the list of tables to refresh (blank means prompt user to select).
selectedLocations - List of comma-separated location ID's of the locations from which to get the test records (blank means prompt user to select).
packageName - Name of package to hold transactions (blank means prompt user to select).
packageDescription - Description of package if a new package is to be created

Returns:

Void.

Class:

EDMcommon.cmd.TableRefreshCmd

resetSite

Deprecated - use resetSites instead. Send Reset transaction to reset a site to its newly installed state.

Syntax:

resetSite('siteList', 'packageName', 'packageDescription')

Parameters:

siteList - Comma-separated list of site IDs to reset.
packageName - Package name to use for transactions.
packageDescription - Package description if creating new package.

Returns:

void

Class:

EDMcommon.cmd.ResetSitesCmd

resetSites

Send transactions to reset remote sites to their newly installed state and commit the transactions.

Syntax:

resetSites('siteIdList')

Parameters:

siteIdList - Comma-separated list of site IDs to reset.

Returns:

void

Class:

EDMcommon.cmd.ResetSitesCmd

restoreContentArea

Restores the content area in the main EDM window.

Syntax:

restoreContentArea()

Parameters:

None.

Returns:

Void.

Class:

EDMcommon.cmd.ToggleContentAreaSizeFunction

restoreTestCheckpoint

Disconnect all database connections, restore checkpoint 1, and re-initialize server

Syntax:

restoreTestCheckpoint('checkPoint', 'batchFile')

Parameters:

checkPoint - Checkpoint to restore.

batchFile - Batch file that restores test checkpoint.

Returns:

Void

Class:

EDMcommon.cmd.RestoreTestCheckpointCmd

retryTest

Retry a test that failed during runTest.

Syntax:

retryTest('testListClassName', 'testClassName', 'firstTestMethodToCall', exitAfterTest)

Parameters:

testListClassName - Name of class that hold the list of test classes.

testClassName - Test class name.

firstTestMethodToCall - First test method in the class to call, null or blank means call all test methods.

exitAfterTest - True if program should exit when test is complete.

Returns:

Void.

Class:

EDMclient.function.RunTestFunction

right

Returns the rightmost characters of a string

Syntax:

right('string', Count)

Parameters:

string - String to get the characters from
Count - Number of characters to get

Returns:

Requested characters

Class:

EDMcommon.function.CommonFunctions

round

Rounds a number at the specified number of digits past the decimal

Syntax:

round('number', digits)

Parameters:

number - Number to round
digits - Number of digits past the decimal to keep

Returns:

Rounded number

Class:

EDMcommon.function.CommonFunctions

runApp

Runs an application as if from the command line. For example, to open the config.xml file in notepad when transactions are processed, set the filespec to `c:\windows\system32\notepad.exe`, the arguments to `c:\EDMweb\config.xml`, and the wait flag to `false`. To run a batch file you could set the filespec to `cmd` and the arguments to `/c c:\EDMweb\go.bat` and the wait flag to `false`.

Syntax:

runApp('filespec', 'arguments', wait)

Parameters:

filespec - Path and filename of application.
arguments - Parameters to pass to the application.
wait - Wait for application to complete before returning.

Returns:

0 if successful, otherwise error code.

Class:

EDMcommon.cmd.RunAppCmd

runJUnitTest

Run the JUnit tests in the given class.

Syntax:

runJUnitTest('jUnitClassName')

Parameters:

jUnitClassName - JUnit test class to run.

Returns:

Void.

Class:

EDMcommon.cmd.RunJUnitTestCmd

runPriceChangeByPriceGroupReport

Run price change by price group report with user-specified filters and sorting.

Syntax:

runPriceChangeByPriceGroupReport()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.plugin.iris.CreateTransactionReportForIrisPricesByPriceGroupCmd

runPriceChangeByPriceReport

Run price change by price report with user-specified filters and sorting.

Syntax:

runPriceChangeByPriceReport()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.plugin.iris.CreateTransactionReportForIrisPricesByPriceCmd

runPriceChangeReport

Run price report with user-specified filters and sorting.

Syntax:

runPriceChangeReport()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.plugin.iris.CreateTransactionReportForIrisPricesCmd

runReport

Run a report.

Syntax:

```
runReport( 'configFileName', 'outputType')
```

Parameters:

configFileName - Path and file name of file containing report settings.)

outputType - Type of report output file. Supported types are 'PDF', 'HTML', and 'XML'.

Returns:

Void.

Class:

EDMcommon.cmd.RunReportCmd

runSQL

Runs an SQL statement.

Syntax:

```
runSQL( 'sql')
```

Parameters:

sql - SQL statement to run.

Returns:

Void.

Class:

EDMclient.ClientFunctions

runTaxChangeReport

Run tax change report with user-specified filters and sorting.

Syntax:

```
runTaxChangeReport()
```

Parameters:

None.

Returns:

Void.

Class:

EDMclient.plugin.iris.CreateTransactionReportForIrisTaxChangesCmd

runTest

Invoke each method whose name starts with 'test' in the given test class.

Syntax:

```
runTest( 'testListClassName', 'testClassName', 'firstTestMethodToCall', exitAfterTest, retry)
```

Parameters:

testListClassName - Name of class that hold the list of test classes.

testClassName - Test class name.

firstTestMethodToCall - First test method in the class to call, null or blank means call all test methods.

exitAfterTest - True if program should exit when test is complete.

retry - True if test should be retried if it fails

Returns:

Void.

Class:

EDMclient.function.RunTestFunction

runTestMethod

Invoke the given test method.

Syntax:

```
runTestMethod( 'testClassName', 'testMethodToCall')
```

Parameters:

testClassName - Test class name.

testMethodToCall - Test method in the class to call.

Returns:

Void.

Class:

EDMclient.function.RunTestFunction

runTestMethods

Invoke the given test method and all methods that follow it in the given class.

Syntax:

```
runTestMethods( 'testClassName', 'testMethodToCall')
```

Parameters:

testClassName - Test class name.

testMethodToCall - Test method in the class to call.

Returns:

Void.

Class:

EDMclient.function.RunTestFunction

runTestUI

Invoke each method whose name starts with 'test' in the given test class.

Syntax:

runTestUI('testListClassName')

Parameters:

testListClassName - Name of class that holds the list of test classes. Leave blank to query for all tests.

Returns:

Void.

Class:

EDMclient.function.RunTestUIFunction

saveRecord

Saves changes to the current record on the active form.

Syntax:

saveRecord()

Parameters:

None.

Returns:

True if the save completed, or false if it was cancelled. NOTE: If the event was cancelled, records that were saved before the event was cancelled will remain saved.

Class:

EDMclient.ClientFunctions

scheduleVersionAssignments

Schedules a version of site subscriptions. Only sites having the Site Property Name and Value will be schedulable.

Syntax:

scheduleVersionAssignments('title', 'packageName', 'sitePropertyName', 'sitePropertyValue')

Parameters:

title - The title to display at the top of the window. HTML can be used to change the font color, size, face, etc.

packageName - The package to manage the schedule for.

sitePropertyName - The name of the site property to use for selecting sites to be scheduled.

sitePropertyValue - The value of the site property to use for gathering deployment information.

Returns:

Void.

Class:

EDMclient.plugin.mdm.cmd.ScheduleVersionAssignmentsFunction

select

Selects either an open object or an object in the application window.

Syntax:

select('objectType', 'objectName', inAppWindow)

Parameters:

objectType - Type of object (table, query, form, report or function).

objectName - Name of the object.

inAppWindow - Select the object in the application window.

Returns:

Void.

Class:

EDMclient.ClientFunctions

selectAllRecords

Selects all records on the active form.

Syntax:

selectAllRecords()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.ClientFunctions

selectLocation

Select one remote location from the list of remote locations maintained in a central database.

Syntax:

selectLocation('title', 'promptText')

Parameters:

title - Selection dialog title.

promptText - Selection dialog prompt text.

Returns:

Selected location number.

Class:

EDMclient.ClientFunctions

selectLocations

Prompt user to select locations from a location tree.

Syntax:

```
selectLocations( locationTree, 'tableToShowShareGroups', allowMultipleLocations, promptForEffectiveDate, 'defaultEffectiveDate',  
promptForTerminationDate, 'defaultTerminationDate', promptForForceProcessing, defaultForceProcessing)
```

Parameters:

locationTree - Tree of locations and group from which user may select (if null, get list from server).

tableToShowShareGroups - Name of table whose share groups are shown in location tree.

allowMultipleLocations - True if user may select multiple locations.

promptForEffectiveDate - True if user may enter effective date.

defaultEffectiveDate - Default effective date in local format.

promptForTerminationDate - True if user may enter termination date.

defaultTerminationDate - Default termination date in local format.

promptForForceProcessing - True if transactions should be processed when only processing forced transactions.

defaultForceProcessing - Default force processing value.

Returns:

Routing information for transactions including list of locations and effective/termination dates.

Class:

EDMclient.ClientFunctions

selectLocationsAndPackage

Prompt user to select locations from a location tree and a package for transactions.

Syntax:

```
selectLocationsAndPackage( 'tableToShowShareGroups', allowMultipleLocations, promptForEffectiveDate, 'defaultEffectiveDate',  
promptForTerminationDate, 'defaultTerminationDate', promptForForceProcessing, defaultForceProcessing)
```

Parameters:

tableToShowShareGroups - Name of table whose share groups are shown in location tree.

allowMultipleLocations - True if user may select multiple locations.

promptForEffectiveDate - True if user may enter effective date.

defaultEffectiveDate - Default effective date in local format.

promptForTerminationDate - True if user may enter termination date.

defaultTerminationDate - Default termination date in local format.

promptForForceProcessing - True if transactions should be processed when only processing forced transactions.

defaultForceProcessing - Default force processing value.

Returns:

Routing information for transactions including list of locations, effective/termination dates, and package.

Class:

EDMclient.ClientFunctions

selectPackage

Prompt user to select from a list of transaction packages or create a new one.

Syntax:

selectPackage('title')

Parameters:

title - Dialog title.

Returns:

PackageInfo object containing selected package name or new package name and description.

Class:

EDMclient.ClientFunctions

selectRecord

Selects current record on the active form.

Syntax:

selectRecord()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.ClientFunctions

selectSessionPackage

Specify the package to use for all changes until a different session package is selected.

Syntax:

selectSessionPackage('packageName', 'requireVersionedPackages')

Parameters:

packageName - Name of the package.

requireVersionedPackages - Packages shown in the list must have assigned versions.

Returns:

Void.

Class:

EDMclient.function.SelectSessionPackageFunction

sendAddSchemasTransaction

Send an transaction to update the table list and schema at selected sites.

Syntax:

```
sendAddSchemasTransaction( 'siteList', 'packageName', 'packageDescription')
```

Parameters:

siteList - Comma-separated list of site IDs to reset.
packageName - Package name to use for transactions.
packageDescription - Package description if creating new package.

Returns:

void

Class:

EDMclient.ClientFunctions

sendDatedTableRefresh

Send table data to locations to replace existing data

Syntax:

```
sendDatedTableRefresh( 'snapshotName', 'selectedLocations', forceProcessing, 'packageName', 'packageDescription', 'rollbackType',  
'rollbackDate', 'rollbackTime')
```

Parameters:

snapshotName - Name of the snapshot containing the list of tables to refresh (blank means prompt user to select).
selectedLocations - List of comma-separated location ID's of the locations from which to get the test records (blank means prompt user to select).
forceProcessing - True if transactions should be processed when only processing forced transactions.
packageName - Name of package to hold transactions (blank means prompt user to select).
packageDescription - Description of package if a new package is to be created
rollbackType - Type of transaction roll back: 'NONE', 'NOW', or 'SPECIFIED'. Prompt user if blank.
rollbackDate - If rollbackType is 'SPECIFIED', date to which transactions should be rolled back, format is mm/dd/yyyy.
rollbackTime - If rollbackType is 'SPECIFIED', time to which transactions should be rolled back, format is hh:mm:ss.SSS AM.

Returns:

Void.

Class:

EDMcommon.cmd.TableRefreshCmd

sendDeleteFileTransaction

Send transaction to delete a file.

Syntax:

```
sendDeleteFileTransaction( 'file', 'siteList', 'effectiveDate', 'packageName', 'packageDescription')
```

Parameters:

file - File name to delete (blank means prompt user to select).
siteList - Comma-separated list of site numbers (blank means prompt user, * means all sites).

effectiveDate - Date/time transaction should be applied.
packageName - Name of package to hold transactions.
packageDescription - Description of package if new package.

Returns:

Void.

Class:

EDMcommon.cmd.SendDeleteFileTransactionCmd

sendEvaluateTransaction

Send a transaction to cause the receiver to evaluate an expression.

Syntax:

sendEvaluateTransaction('expression', 'siteIdList', 'effectiveDate', 'packageName', 'packageDescription', forceProcessing)

Parameters:

expression - Expression to evaluate on receiving server (uses only server functions).
siteIdList - Comma-separated list of site numbers (blank means prompt user, * means all sites).
effectiveDate - Date/time transaction should be applied in local format or blank to be effective immediately.
packageName - Name of package to hold transactions.
packageDescription - Description of package if new package.
forceProcessing - True if transactions should be processed when only processing forced transactions.

Returns:

Void.

Class:

EDMcommon.cmd.SendEvaluateTransactionCmd

sendExecuteTransaction

Send transaction to execute an operation system command at selected sites.

Syntax:

sendExecuteTransaction('file', 'arguments', wait, 'siteList', 'effectiveDate', 'packageName', 'packageDescription')

Parameters:

file - Full path of executable file (if blank, prompt user to select)
arguments - Command line arguments.
wait - True if call should wait for process to complete before returning.
siteList - Comma-separated list of site numbers (blank means prompt user, * means all sites).
effectiveDate - Date/time transaction should be applied.
packageName - Name of package to hold transactions.
packageDescription - Description of package if new package.

Returns:

Void.

Class:

EDMcommon.cmd.SendExecuteTransactionCmd

sendFile

Send a file to other locations

Syntax:

sendFile('locationList', 'fromFile', 'toFile', flags, 'effectiveDate', 'packageName', 'packageDescription')

Parameters:

locationList - Comma-separated list of location numbers from which to request file (blank means prompt user, * means all locations).

fromFile - Full path of file to copy (if blank, prompt user to select)

toFile - Full path of destination file at other locations (if blank, prompt user to select)

flags - 1 if file should be overwritten if it exists at destination location; 2 if file should be kept synchronized after being sent.

effectiveDate - Date file should be copied to destination

packageName - Name of package to hold transactions

packageDescription - Description of package if new package.

Returns:

Void.

Class:

EDMcommon.cmd.WriteFileTransactionCmd

sendGetSchemasTransaction

Send a transaction to get the table list and schema from selected sites.

Syntax:

sendGetSchemasTransaction('siteList', 'packageName', 'packageDescription')

Parameters:

siteList - Comma-separated list of site IDs to reset.

packageName - Package name to use for transactions.

packageDescription - Package description if creating new package.

Returns:

void

Class:

EDMclient.ClientFunctions

sendRefreshToTestSite

Send table refresh transaction from a site to a lab/test site.

Syntax:

sendRefreshToTestSite('snapshotName', 'dataSiteId', 'testSiteId', 'rollbackType', 'rollbackDate', 'rollbackTime')

Parameters:

snapshotName - Name of snapshot containing list of tables to refresh (blank means prompt user to select).

dataSiteId - ID of the site whose data is to be sent (blank means prompt user to select).

testSiteId - ID of the test site where data is to be sent (blank means prompt user to select).

rollbackType - Type of transaction roll back: 'NONE', 'NOW', or 'SPECIFIED'. Prompt user if blank.

rollbackDate - If rollbackType is 'SPECIFIED', date to which transactions should be rolled back, format is mm/dd/yyyy.

rollbackTime - If rollbackType is 'SPECIFIED', time to which transactions should be rolled back, format is hh:mm:ss.SSS AM.

Returns:

Void.

Class:

EDMcommon.cmd.TableRefreshCmd

sendRemoteFiles

Send files listed in remote files table to sites.

Syntax:

sendRemoteFiles(sendUpdatedOnly, 'siteList', 'effectiveDate', 'packageName', 'packageDescription', forceProcessing, commit)

Parameters:

sendUpdatedOnly - True if only files that have been updated should be sent.

siteList - Comma-separated list of site numbers to which files should be sent (* means all, blank means prompt user to select).

effectiveDate - Date file should be copied to destination

packageName - Name of package to hold transactions (blank means prompt user to select).

packageDescription - Description of package if a new package is to be created

forceProcessing - True if transactions should be processed when only processing forced transactions.

commit - True if package containing file transactions should be committed.

Returns:

Void.

Class:

EDMclient.ClientFunctions

sendRenameFileTransaction

Send transaction to rename a file.

Syntax:

sendRenameFileTransaction('oldFile', 'newFile', 'siteList', 'effectiveDate', 'packageName', 'packageDescription')

Parameters:

oldFile - Old file name (blank means prompt user to select).

newFile - New file name (blank means prompt user to select).

siteList - Comma-separated list of site numbers (blank means prompt user, * means all sites).

effectiveDate - Date/time transaction should be applied.

packageName - Name of package to hold transactions.

packageDescription - Description of package if new package.

Returns:

Void.

Class:

EDMclient.ClientFunctions

sendSiteFilesViaWeb

Send transaction files over HTTP web connection.

Syntax:

sendSiteFilesViaWeb('siteIdList')

Parameters:

siteIdList - Comma-separated list of sites id's to transfer (blank to prompt user, * for all sites).

Returns:

Void.

Class:

EDMcommon.cmd.TransferFilesCmd

sendSynchronizeDirectoryTransaction

Send a transaction to synchronize a local directory with a directory at other sites.

Syntax:

sendSynchronizeDirectoryTransaction('siteIdList', 'sourceDirectory', 'destinationDirectory', sourceDirectoryIsOnSender, deleteMissingFiles, includeSubdirectories, 'effectiveDate', forceProcessing, 'packageName', 'packageDescription')

Parameters:

siteIdList - Comma-separated list of site IDs to which transaction will be sent (blank means prompt user, * means all sites).

sourceDirectory - Directory containing up-to-date files.

destinationDirectory - Directory that should be updated to match source directory.

sourceDirectoryIsOnSender - True if the source directory is on the local network of the system sending the transaction.

deleteMissingFiles - True if files in the destination directory should be deleted if they do not exist in the source directory.

includeSubdirectories - True if subdirectories should also be synchronized.

effectiveDate - Effective date of transaction.

forceProcessing - True if processed when only processing forced transactions.

packageName - Name of package to hold transactions.

packageDescription - Description of package if new package.

Returns:

Void.

Class:

EDMcommon.cmd.SendSyncDirectoryTranCmd

sendTableRefresh

Send table data to locations to replace existing data

Syntax:

sendTableRefresh('snapshotName', 'selectedLocations', 'packageName', 'packageDescription')

Parameters:

snapshotName - Name of the snapshot containing the list of tables to refresh (blank means prompt user to select).

selectedLocations - List of comma-separated location ID's of the locations from which to get the test records (blank means prompt user to select).

packageName - Name of package to hold transactions (blank means prompt user to select).

packageDescription - Description of package if a new package is to be created

Returns:

Void.

Class:

EDMcommon.cmd.TableRefreshCmd

sendTransactionFilesToAmazon

Send transaction files to Amazon Simple Storage Service (S3).

Syntax:

```
sendTransactionFilesToAmazon('siteIdList', 'bucketName', 'prefix', 'encryptedCredentials')
```

Parameters:

siteIdList - Comma-separated list of site IDs to send (blank to prompt user, * for all locations).

bucketName - Name of Amazon S3 bucket containing incoming and outgoing directories.

prefix - Common prefix of all transaction file names to be transferred without the site ID path segment, e.g. 'outgoing/'.

encryptedCredentials - AWS credentials encrypted with the EncryptAmazonCredentialsCmd object.

Returns:

Void.

Class:

EDMcommon.cmd.SendTransactionFilesToAmazonCmd

sendTransactionFilesToJMS

Send outgoing files for the given sites to a JMS queue.

Syntax:

```
sendTransactionFilesToJMS('siteIdList', 'queueName', 'brokerURL', 'userId', 'password')
```

Parameters:

siteIdList - Comma-separated list of site IDs to send (blank to prompt user, * for all locations).

queueName - Name of the queue where outgoing transaction files should be sent. For example: EDM.outgoing.%REMOTESITEID%

brokerURL - URL used to connect to the JMS broker. For example: tcp://localhost:61616

userId - User ID used to connect to the JMS broker.

password - Password used to connect to the JMS broker.

Returns:

Void.

Class:

EDMcommon.cmd.SendTransactionFilesToJMScmd

setClientPropertyBoolean

Sets a client property value with the given name.

Syntax:

```
setClientPropertyBoolean('name', value)
```

Parameters:

name - Name of the client property.

value - Boolean value of the client property.

Returns:

Void.

Class:

EDMclient.function.SetClientPropertyFunction

setClientPropertyInteger

Sets a client property value with the given name.

Syntax:

setClientPropertyInteger('name', value)

Parameters:

name - Name of the client property.

value - Integer value of the client property.

Returns:

Void.

Class:

EDMclient.function.SetClientPropertyFunction

setClientPropertyString

Sets a client property value with the given name.

Syntax:

setClientPropertyString('name', 'value')

Parameters:

name - Name of the client property.

value - Value of the client property.

Returns:

Void.

Class:

EDMclient.function.SetClientPropertyFunction

setConfigProperty

Set configuration property value.

Set value of the property in the configuration file with the given case-sensitive name.

Syntax:

setConfigProperty('propertyName', 'value')

Parameters:

propertyName - Name of the user property.

value - New property value.

Returns:

Old property value.

Class:

EDMcommon.cmd.SetConfigPropertyCmd

setControlProperty

Sets a property value for a control on the active form.

Syntax:

```
setControlProperty( 'controlName', 'propertyName', 'value')
```

Parameters:

controlName - Name of the control or a comma separated list of control names.

propertyName - Name of the property.

value - Value to set the property to.

Returns:

Void.

Class:

EDMclient.function.SetControlPropertyFunction

setFormControlProperty

Sets a property value for a control on a form.

Syntax:

```
setFormControlProperty( 'formName', 'controlName', 'propertyName', 'value')
```

Parameters:

formName - Name of the form.

controlName - Name of the control.

propertyName - Name of the property.

value - Value to set the property to.

Returns:

Void.

Class:

EDMclient.function.SetControlPropertyFunction

setFormRowSource

Sets the row source value for a control on the active form.

Syntax:

```
setFormRowSource( 'formName', 'controlName', 'source')
```

Parameters:

formName - Name of the form the control is on.

controlName - Name of the control.

source - Name of the table or query.

Returns:

Void.

Class:

EDMclient.ClientFunctions

setFormSiteValue

Sets site specific data for a control on the specified form.

Syntax:

```
setFormSiteValue( 'formName', 'controlNames', 'expr')
```

Parameters:

formName - Name of the form the control is on.

controlNames - Comma-delimited list of control names.

expr - An expression to evaluate to get each site's value.

Returns:

Void.

Class:

EDMclient.function.ControlValueFunctions

setFormSiteValueToNull

Sets site specific data for a control on the specified form to null.

Syntax:

```
setFormSiteValueToNull( 'formName', 'controlNames')
```

Parameters:

formName - Name of the form the control is on.

controlNames - Comma-delimited list of control names.

Returns:

Void.

Class:

EDMclient.function.ControlValueFunctions

setFormValue

Sets data for a control on a form.

Syntax:

```
setFormValue( 'formName', 'controlNames', 'value')
```

Parameters:

formName - Name of the form the control is on.

controlNames - Comma-delimited list of control names.

value - Value to set the control to.

Returns:

Void.

Class:

EDMclient.function.ControlValueFunctions

setFormValueToNull

Sets data for controls on a form to null.

Syntax:

setFormValueToNull('formName', 'controlNames')

Parameters:

formName - Name of the form the control is on.

controlNames - Comma-delimited list of control names.

Returns:

Void.

Class:

EDMclient.function.ControlValueFunctions

setLastReceivedTranFileNumberToIncoming

Set the last received transaction file number for a site to the lowest transaction file number in the incoming directory - 1.

Syntax:

setLastReceivedTranFileNumberToIncoming(siteId)

Parameters:

siteId - ID of the site whose last received transaction file number is to be set.

Returns:

Void.

Class:

EDMcommon.cmd.SetLastReceivedTranFileNumberToIncomingCmd

setLogConsoleLevel

Sets level of log messages written to console.

Syntax:

setLogConsoleLevel('levelName')

Parameters:

levelName - Level of messages to be logged (ALL,SEVERE,WARNING,CONFIG,FINE,FINER,FINEST,OFF) (blank means prompt user).

Returns:

Void.

Class:

EDMclient.ClientFunctions

setLogFileLevel

Sets level of log messages written to log file.

Syntax:

setLogFileLevel('levelName')

Parameters:

levelName - Level of messages to be logged (ALL, SEVERE, WARNING, CONFIG, FINE, FINER, FINEST, OFF) (blank means prompt user).

Returns:

Void.

Class:

EDMclient.ClientFunctions

setPropertyFileValue

Set the value of a property in a property file.

Syntax:

setPropertyFileValue('propertyName', 'propertyValue', 'propertyFileName', createFile, 'comments')

Parameters:

propertyName - Name of the property whose value is to be returned.

propertyValue - Property value.

propertyFileName - Path and name of property file.

createFile - True if property file should be created if it does not exist.

comments - Comments to add at the beginning of the property file or null for none.

Returns:

Void.

Class:

EDMcommon.cmd.SetPropertyFileValueCmd

setRecordModified

Sets the current record as modified so that it will be saved.

Syntax:

setRecordModified()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.ClientFunctions

setRowSource

Sets the row source value for a control on the active form.

Syntax:

```
setRowSource( 'controlName', 'source')
```

Parameters:

controlName - Name of the control.

source - Name of the table or query.

Returns:

Void.

Class:

EDMclient.ClientFunctions

setSiteValue

Sets site specific data for a control on the active form.

Syntax:

```
setSiteValue( 'controlNames', 'expr')
```

Parameters:

controlNames - Comma-delimited list of control names.

expr - An expression to evaluate to get each site's value.

Returns:

Void.

Class:

EDMclient.function.ControlValueFunctions

setSiteValueToNull

Sets site specific data for a control on the active form to null.

Syntax:

```
setSiteValueToNull( 'controlNames')
```

Parameters:

controlNames - Comma-delimited list of control names.

Returns:

Void.

Class:

EDMclient.function.ControlValueFunctions

setValue

Sets data for a control on the active form.

Syntax:

```
setValue( 'controlNames', 'value')
```

Parameters:

controlNames - Comma-delimited list of control names.

value - Value to set the control to.

Returns:

Void.

Class:

EDMclient.function.ControlValueFunctions

setValueToNull

Sets data for a control on the active form to null.

Syntax:

```
setValueToNull( 'controlNames')
```

Parameters:

controlNames - Comma-delimited list of control names.

Returns:

Void.

Class:

EDMclient.function.ControlValueFunctions

shareLocationData

Share data for selected tables from one location to another location

Syntax:

```
shareLocationData( 'dataLocation', 'shareLocations', 'tableList', 'packageName', 'packageDescription')
```

Parameters:

dataLocation - Id of the location to which will share its data with other locations (blank means prompt user).

shareLocations - List of comma-separated location id's of the locations which will get their data from the data location (blank means prompt user).

tableList - List of comma-separated table names of the tables to share (blank means prompt user).

packageName - Name of package to hold transactions (blank means prompt user to select).

packageDescription - Description of package if a new package is to be created

Returns:

Void.

Class:

EDMclient.ClientFunctions

showAboutDialog

Displays the About dialog with system name and copyright info.

Syntax:

showAboutDialog()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.ClientFunctions

showCopySiteSubscriptionsForm

Show the UI for selecting a source site, target sites, and configurations.

Syntax:

showCopySiteSubscriptionsForm('title', 'deployTypeId', 'panelName')

Parameters:

title - The title to display at the top of the window. HTML can be used to change the font color, size, face, etc.

deployTypeId - The ID of the deployment type, which is used to filter the sites.

panelName - The fully qualified name of the panel to use to select sites in the UI - i.e. EDMclient.ctl.SelectSitesPanel, which will be used if the parameter is left blank.

Returns:

None.

Class:

EDMclient.plugin.mdm.cmd.ShowCopySiteSubscriptionsFormFunction

showCreateSiteSubscriptionsForm

Show the UI for creating site subscriptions.

Syntax:

showCreateSiteSubscriptionsForm('title', 'panelName')

Parameters:

title - The title to display at the top of the window. HTML can be used to change the font color, size, face, etc.

panelName - The fully qualified name of the panel to use to select sites in the UI - i.e. EDMclient.ctl.SelectSitesPanel, which will be used if the parameter is left blank.

Returns:

None.

Class:

EDMclient.plugin.mdm.cmd.ShowCreateSiteSubscriptionsFormFunction

showDocument

Display a document in a browser window. Applets can only display documents on the same server they were started from.

Syntax:

showDocument('url')

Parameters:

url - URL of the document to display relative to the web application directory.

Returns:

Void

Class:

EDMclient.ClientFunctions

showErrors

Turn error messages on or off

Syntax:

showErrors(show)

Parameters:

show - T/F display error messages

Returns:

1 if messages were on before calling this function, otherwise 0

Class:

EDMclient.ClientFunctions

showLdapPropertyForm

Displays the LDAP configuration form.

Syntax:

showLdapPropertyForm()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.ClientFunctions

showSiteManager

Display site manager window.

Syntax:

```
showSiteManager()
```

Parameters:

None.

Returns:

void

Class:

EDMclient.ClientFunctions

showWarnings

Turn warning messages on or off

Syntax:

```
showWarnings( show)
```

Parameters:

show - T/F display warning messages

Returns:

1 if messages were on before calling this function, otherwise 0

Class:

EDMclient.ClientFunctions

sleep

Causes thread to sleep for the specified number of milliseconds.

Syntax:

```
sleep( millis)
```

Parameters:

millis - Number of milliseconds to sleep..

Returns:

Void.

Class:

EDMclient.ClientFunctions

space

Creates a string of blanks

Syntax:

space(count)

Parameters:

count - Number of blanks to put in the string

Returns:

String of blanks

Class:

EDMcommon.function.CommonFunctions

startRecordingTestScript

Tell server to start recording test script with the next login command.

Syntax:

startRecordingTestScript('name')

Parameters:

name - Name of the test script directory.

Returns:

Void.

Class:

EDMclient.ClientFunctions

stopRecordingTestScript

Tell server to stop recording test script.

Syntax:

stopRecordingTestScript('name')

Parameters:

name - Name of the test script directory.

Returns:

Void.

Class:

EDMclient.ClientFunctions

stringToDate

Converts a date string to a date value

Syntax:

stringToDate('date', 'format')

Parameters:

date - Date string to convert
format - Format string to use

Returns:

Date value

Class:

EDMcommon.function.CommonFunctions

stringToInt

Converts a string to an integer.

Syntax:

stringToInt('string')

Parameters:

string - String to convert

Returns:

Integer value.

Class:

EDMcommon.function.CommonFunctions

stringToNumber

Converts a string to a number

Syntax:

stringToNumber('number')

Parameters:

number - Number string to convert

Returns:

Number value

Class:

EDMcommon.function.CommonFunctions

stringToTime

Converts a time string to a time value

Syntax:

stringToTime('time', 'format')

Parameters:

time - Time string to convert
format - Format string to use

Returns:

Time value

Class:

EDMcommon.function.CommonFunctions

stringToTimestamp

Converts a timestamp string to a timestamp value

Syntax:

stringToTimestamp('timestamp', 'format')

Parameters:

timestamp - Timestamp string to convert
format - Format string to use

Returns:

Date value

Class:

EDMcommon.function.CommonFunctions

substring

Returns characters from the middle of a string

Syntax:

substring('astring', pos, count)

Parameters:

astring - String to get the characters from
pos - Index of the first character to get
count - Number of characters to get

Returns:

Requested characters

Class:

EDMcommon.function.CommonFunctions

synchronizeSitePropertyValues

Import and sync the site property values from an outside data source

Syntax:

synchronizeSitePropertyValues('DSN', 'SQL', 'tableName', 'LocationIdColName', 'LocationNameColName')

Parameters:

DSN - The string of the connection in the config.xml to be used to run the SQL. Required.

SQL - The SQL string to be executed to get the source data.

tableName - The name of the table or view that the data resides in.

LocationIdColName - The column name of the location id field. Blank defaults to "Location".

LocationNameColName - The column name of the location name field. Blank defaults new stores to "Store <id>" for a name and no updates to the existing location names will be performed.

Returns:

Void.

Class:

EDMclient.ClientFunctions

testJOptionPane

Test JOptionPane for slow closing and going behind the browser window.

Syntax:

testJOptionPane()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.ClientFunctions

time

Gets current time of day

Syntax:

time('format')

Parameters:

format - Format string describing how the date should be formatted.

Returns:

Current time formatted using format string

Class:

EDMcommon.function.CommonFunctions

timestamp

Get current date and time.

Syntax:

timestamp('format')

Parameters:

format - Format string describing how the date and time should be formatted.

Returns:

String representation of the current date and time.

Class:

EDMcommon.function.CommonFunctions

timestampToDate

Converts a timestamp to a date

Syntax:

timestampToDate('timestamp')

Parameters:

timestamp - Timestamp to convert

Returns:

Date part of the timestamp

Class:

EDMcommon.function.CommonFunctions

timestampToNumber

Converts a timestamp to a number

Syntax:

timestampToNumber('timestamp')

Parameters:

timestamp - Timestamp to convert

Returns:

Number representing the timestamp

Class:

EDMcommon.function.CommonFunctions

timestampToString

Converts a timestamp to a character string

Syntax:

timestampToString('timestamp', 'format')

Parameters:

timestamp - Date to convert

format - Format string to use

Returns:

String representation of the timestamp

Class:

EDMcommon.function.CommonFunctions

timestampToTime

Converts a timestamp to a time

Syntax:

timestampToTime('timestamp')

Parameters:

timestamp - Timestamp to convert

Returns:

Time part of the timestamp

Class:

EDMcommon.function.CommonFunctions

timeToNumber

Converts a time to a number

Syntax:

timeToNumber('time')

Parameters:

time - Time to convert

Returns:

Number representing the time

Class:

EDMcommon.function.CommonFunctions

timeToString

Converts a time to a character string

Syntax:

timeToString('time', 'format')

Parameters:

time - Time to convert

format - Format string to use

Returns:

String representation of the time

Class:

EDMcommon.function.CommonFunctions

timeToTimestamp

Converts a time to a timestamp

Syntax:

timeToTimestamp('time')

Parameters:

time - Time to convert

Returns:

Timestamp with date part set to 01/01/01

Class:

EDMcommon.function.CommonFunctions

transferLocationFilesViaFTP

Transfer files over FTP connection.

Syntax:

transferLocationFilesViaFTP('locationList')

Parameters:

locationList - Comma-separated list of Location id's to transfer (blank to prompt user, * for all locations).

Returns:

Void.

Class:

EDMcommon.cmd.TransferFilesCmd

transferLocationFilesViaRAS

Transfer files over network connection.

Syntax:

transferLocationFilesViaRAS('locationList')

Parameters:

locationList - Comma-separated list of Location id's to transfer (blank to prompt user, * for all locations).

Returns:

Void.

Class:

EDMcommon.cmd.TransferFilesCmd

transferLocationFilesViaWeb

Transfer files over HTTP web connection.

Syntax:

transferLocationFilesViaWeb('locationList')

Parameters:

locationList - Comma-separated list of Location id's to transfer (blank to prompt user, * for all locations).

Returns:

Void.

Class:

EDMcommon.cmd.TransferFilesCmd

transferSiteFilesViaFTP

Transfer transaction files over FTP connection.

Syntax:

transferSiteFilesViaFTP('siteIdList')

Parameters:

siteIdList - Comma-separated list of sites id's to transfer (blank to prompt user, * for all sites).

Returns:

Void.

Class:

EDMcommon.cmd.TransferFilesCmd

transferSiteFilesViaRAS

Transfer transaction files over network connection.

Syntax:

transferSiteFilesViaRAS('siteIdList')

Parameters:

siteIdList - Comma-separated list of sites id's to transfer (blank to prompt user, * for all sites).

Returns:

Void.

Class:

EDMcommon.cmd.TransferFilesCmd

transferSiteFilesViaWeb

Transfer transaction files over HTTP web connection.

Syntax:

```
transferSiteFilesViaWeb( 'siteIdList')
```

Parameters:

siteIdList - Comma-separated list of sites id's to transfer (blank to prompt user, * for all sites).

Returns:

Void.

Class:

EDMcommon.cmd.TransferFilesCmd

trim

Trims the blanks from both ends of a string

Syntax:

```
trim( 'string')
```

Parameters:

string - String to trim

Returns:

Trimmed string

Class:

EDMcommon.function.CommonFunctions

trimLeft

Trims the blanks from the left end of a string

Syntax:

```
trimLeft( 'string')
```

Parameters:

string - String to trim

Returns:

Trimmed string

Class:

EDMcommon.function.CommonFunctions

trimRight

Trims the blanks from the right end of a string

Syntax:

trimRight('string')

Parameters:

string - String to trim

Returns:

Trimmed string

Class:

EDMcommon.function.CommonFunctions

undo

Performs undo operation in the active window.

Syntax:

undo()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.ClientFunctions

unlockAppObjects

Prompt user to unlock selected application objects that are locked while there design is being edited.

Syntax:

unlockAppObjects()

Parameters:

None.

Returns:

Void.

Class:

EDMcommon.cmd.GetAppObjectLockListCmd

unshareLocationData

Stop sharing data for selected tables from one location to another location

Syntax:

unshareLocationData('shareLocations', 'tableList')

Parameters:

shareLocations - List of comma-separated location id's of the locations which will no longer get their data from another location (blank means prompt user).

tableList - List of comma-separated table names of the tables to stop sharing (blank means prompt user).

Returns:

Void.

Class:

EDMclient.ClientFunctions

updateAmazonDirToMatchLocalDir

Update an Amazon S3 directory to match a local directory.

Syntax:

```
updateAmazonDirToMatchLocalDir( 'serverDirectory', 'localDirectory', deleteMissingFiles, 'bucketName', 'encryptedCredentials')
```

Parameters:

serverDirectory - Directory on the server to update to match the files in the local directory.

localDirectory - Local directory to check for new, updated, and deleted files.

deleteMissingFiles - True if files in the S3 directory should be deleted if they do not exist in the local directory.

bucketName - Name of Amazon S3 bucket containing server directory.

encryptedCredentials - AWS credentials encrypted with the EncryptAmazonCredentialsCmd object.

Returns:

Void

Class:

EDMcommon.cmd.UpdateAmazonDirToMatchLocalDirCmd

updateLocalDirToMatchAmazonDir

Update a local directory to match a directory on Amazon S3.

Syntax:

```
updateLocalDirToMatchAmazonDir( 'serverDirectory', 'localDirectory', deleteMissingFiles, 'bucketName', 'encryptedCredentials')
```

Parameters:

serverDirectory - Server directory containing files and sub-directories.

localDirectory - Local directory to synchronize.

deleteMissingFiles - True if files in the local directory should be deleted if they do not exist on the server.

bucketName - Name of Amazon S3 bucket containing server directory.

encryptedCredentials - AWS credentials encrypted with the EncryptAmazonCredentialsCmd object.

Returns:

Void

Class:

EDMcommon.cmd.UpdateLocalDirToMatchAmazonDirCmd

updateLocalDirToMatchWebDir

Update a local directory to match a directory on the EDM web server.

Syntax:

```
updateLocalDirToMatchWebDir( 'serverDirectory', 'localDirectory', deleteMissingFiles)
```

Parameters:

serverDirectory - Server directory containing files and sub-directories.

localDirectory - Local directory to synchronize.

deleteMissingFiles - True if files in the local directory should be deleted if they do not exist on the server.

Returns:

Void

Class:

EDMcommon.cmd.UpdateLocalDirToMatchWebDirCmd

updateRows

Update selected rows for selected sites and generate transactions for the changes.

Syntax:

```
updateRows( 'tableName', 'columnNames', 'columnValues', 'whereClause', 'siteIdList', 'packageName', 'packageDescription', 'effectiveDate', 'terminationDate', forceProcessing)
```

Parameters:

tableName - Name of the table to be updated.

columnNames - Comma-delimited list of column names to be updated.

columnValues - New column values to be updated.

whereClause - Where clause to limit rows that are updated of the form or blank to update all rows for the selected sites. E.g. 'TaxCatNum = ' & Forms:Update Item TaxCatNum:OldTaxCatNum

siteIdList - Comma-delimited list of site IDs to update or blank for all sites.

packageName - Package name.

packageDescription - Package description.

effectiveDate - Date/time transaction becomes effective.

terminationDate - Date/time transaction terminates.

forceProcessing - True if transactions should be processed when only processing forced transactions.

Returns:

Void.

Class:

EDMcommon.cmd.UpdateRowsCmd

updateTableList

Update a table list.

Syntax:

```
updateTableList( 'oldTableListName', tableList, index)
```

Parameters:

oldTableListName - New table list to insert.

tableList - New table list.

index - 0-based index where table list should be inserted after deleting original table list, if index is < 0 or >= list size, the new table list is inserted at the same index as the original table list.

Returns:

Void.

Class:

EDMcommon.cmd.EditSnapshotTablesCmd

updateTransferSettings

Save the given transfer host, user ID, password, and company name used to connect to a central EDM server

<p>The updateTransferSettings function updates the transfer host, user ID, password, and company name on the central site record. This information is used when connecting to the central system to transfer transaction files. To have the information automatically saved when the system starts up, the function can be called in the onStartUpProperty of config.xml as shown in the following example.</p><pre>onStartup = "updateTransferSettings('localhost:8080/EDM', 'remote', 'remote', 'HQ')" </pre>

Syntax:

```
updateTransferSettings( 'hostName', 'userId', 'password', 'company')
```

Parameters:

hostName - Host name for file transfers.
userId - User id for transaction file transfers.
password - Password for transaction file transfers.
company - Company name for transaction file transfers.

Returns:

Void.

Class:

EDMcommon.cmd.UpdateTransferSettingsCmd

updateWebDirToMatchLocalDir

Update an directory on the EDM web server to match a local directory.

Syntax:

```
updateWebDirToMatchLocalDir( 'serverDirectory', 'localDirectory', deleteMissingFiles)
```

Parameters:

serverDirectory - Directory on the server to update to match the files in the local directory.
localDirectory - Local directory to check for new, updated, and deleted files.
deleteMissingFiles - True if files in the server directory should be deleted if they do not exist in the local directory.

Returns:

Void

Class:

EDMcommon.cmd.UpdateWebDirToMatchLocalDirCmd

upper

Changes each letter of a string to upper case

Syntax:

upper('string')

Parameters:

string - String to convert

Returns:

String converted to upper case

Class:

EDMcommon.function.CommonFunctions

validateTableList

Validate references between tables

Syntax:

validateTableList('tableListName', 'siteIdList')

Parameters:

tableListName - Name of the table list containing the list of tables to validate (blank means prompt user to select).

siteIdList - List of comma-separated location ID's of the locations from which to get the test records (blank means prompt user to select, * means all locations).

Returns:

Void.

Class:

EDMclient.ClientFunctions

validateTables

Validate references between tables

Syntax:

validateTables('tableNames', 'siteIdList')

Parameters:

tableNames - Comma-separated list of tables to validate (blank means prompt user to select, * means all tables).

siteIdList - List of comma-separated location ID's of the locations from which to get the test records (blank means prompt user to select, * means all locations).

Returns:

Void.

Class:

EDMcommon.cmd.ValidateTablesCmd

view

Opens an Table, Query, or Form on the client.

Syntax:

view('objectType', 'objectName')

Parameters:

objectType - Type of object (table, query, or form).

objectName - Name of the object.

Returns:

Void.

Class:

EDMcommon.cmd.OpenFormPlayerCmd

viewForm

Ask server to open a form on the client that is visible in its default view. All parameters must be complete, user will be prompted for routing information if necessary.

Syntax:

viewForm('name', 'view', 'visible')

Parameters:

name - Name of form.

view - View to display, 'SINGLE' (form view), 'GRID' (spreadsheet view) or "" (default view).

visible - True if form should be visible after opening.

Returns:

Void.

Class:

EDMcommon.cmd.OpenFormPlayerCmd

waitForServerJobs

Wait for all server jobs to complete. If not server jobs are running when the method is called, wait up to one second for server jobs to start. After waiting for a server job to finish, wait up to one second for another server job to start. Total minimum wait time is two seconds.

Syntax:

waitForServerJobs()

Parameters:

None.

Returns:

Void.

Class:

EDMclient.ClientFunctions

year

Returns the year value of a date

Syntax:

year('date')

Parameters:

date - Date to get the year from

Returns:

Four digit year number

Class:

EDMcommon.function.CommonFunctions

Server Functions

This section describes the server functions available in EDM.

Calling Functions in Expressions

Type the function name followed by its parameters. Enclose the parameters in parentheses ().

Calling a Function on a Control or Form Event

Precede the function call with an equal = sign. For example, to close the active window when the user clicks a button, type =CloseActiveWindow() in the *On push* attribute.

To call more than one function, connect the function names with operators like the plus + sign. For example, to close the active window and open the Customer form when the user clicks a button, type =CloseActiveWindow() + Open('form', 'Customer') in the *On push* attribute.

Function Names

- Function names must be followed by parentheses even when there are no parameters.
- Function names are not case sensitive.

Function Parameters

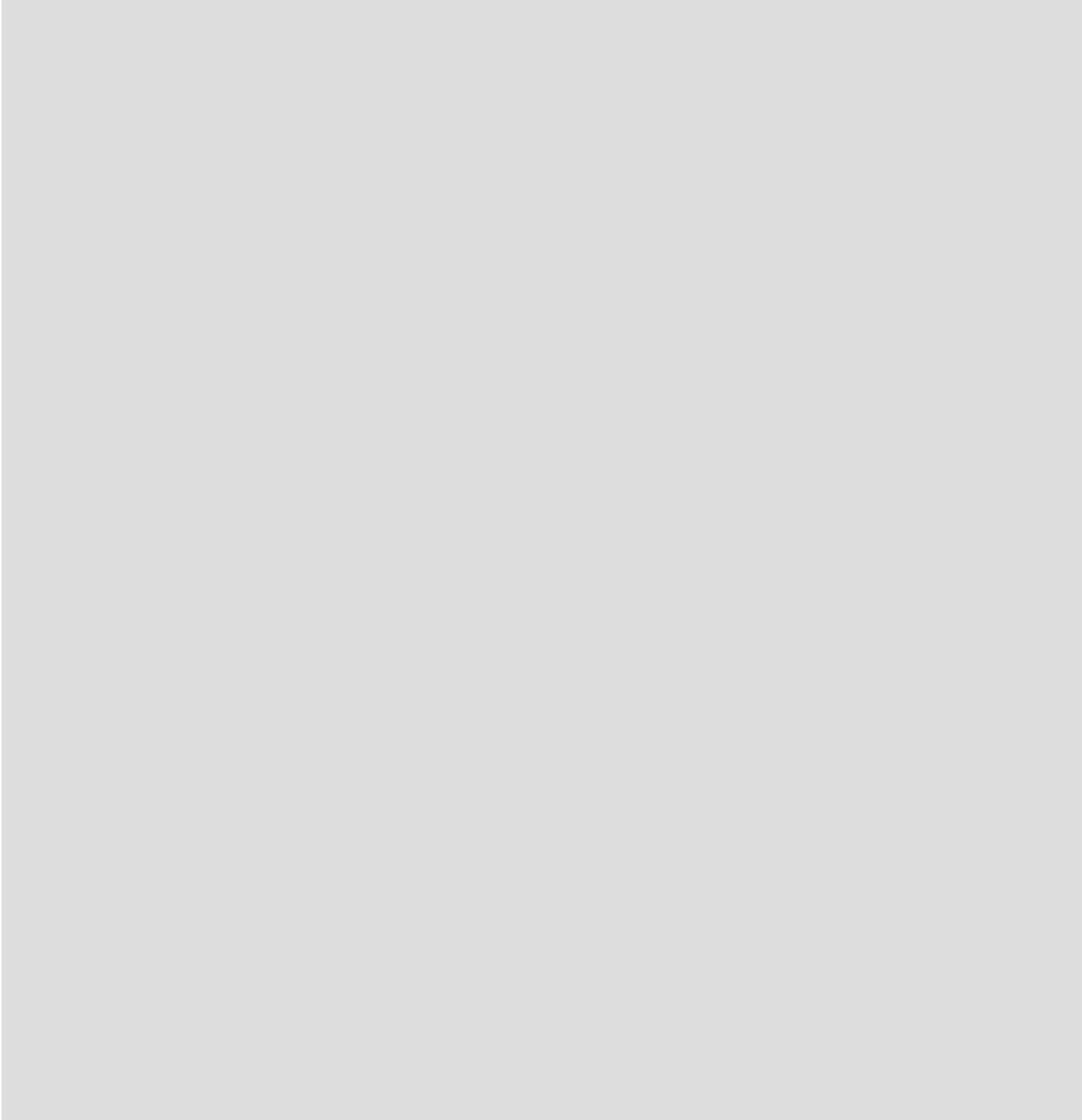
- String parameters must be enclosed in either single or double quotes.
- Numeric parameters should NOT be enclosed in quotes.
- True/False parameters may be entered as true or false without quotes, or as 1 or 0 (1=true 0=false).

Alphabetic Function Index

The following table lists all the functions in alphabetical order.

Function	What it does...
abs	Get the absolute value of the given number. If the number is null, a DataNumber with the value null is returned. If the number is blank, 0 is returned.
addAmazonCommitListener	Add listener that sends outgoing transaction files for selected sites to Amazon as soon as they are committed.
addDeleteAmazonTransactionsOnResetListener	Add listener that deletes transaction files from the Amazon incoming and outgoing directory when a reset site transaction is committed.
addJMSCommitListener	Add listener that sends outgoing transaction files for selected sites to a JMS queue as soon as they are committed.
addJMSReceiveListener	Add listener that receives transaction file messages from a JMS queue and writes them to the incoming directory.
addJMSReceiveListenerWithRetry	Add listener that receives transaction file messages from a JMS queue and writes them to the incoming directory.

addProcessReceivedTransactionsListener	Add listener that automatically processes incoming transactions as soon as they are received.
addRow	Insert a row in a table for a given list of sites that do not already have the row. For example, when entering a
addSystemEventListener	Add the listener in the given class to the list of system event listeners.
addThisSiteToCentral	Call the central web service to add this remote site to the central database.
addTransactionsProcessedListener	Add listener that evaluates the given expression whenever transaction processing completes.
applyEffectiveMasterDataTrans	Queries the audit table for committed transactions on master data tables whose effective date has arrived and calls the appropriate master data change listener for each transaction then changes the status of the transaction to Successful.
ascii	Converts the first character of a string to an ASCII value
asciiToChar	Converts an ASCII code to a character
attachMasterTables	Attach tables from an existing database so it can be edited
attachRemoteTable	Attach a table from an existing database so it can be edited
attachTable	Attach tables from an existing database so it can be edited
callEvaluateExpressionWebMethod	Call the EvaluateExpression web method.
changePackageStatus	Change status of the transactions for sites in a package to a new status.
changePassword	Change the current user's password.
commitTransactions	Commit packages of transactions so they are sent to selected locations
copyAndRefreshSiteData	Copy data for selected tables from one site to other sites and refresh the tables at the other sites on the effective date.
copyCompany	Create a new company by copying the configuration of an existing company.
copySiteData	Copy data for selected tables from one location to another location
createConfigurationVersionDataSite	Creates a data site to hold version data for configuration version. The ID of the new site will be an unused, negative site number. The name of the site will be the name of the configuration plus a hyphen plus the name of the version. Each table in the configuration type will be added to the list of managed site tables. If there is a previous version of the configuration then the data from the previous version will be copied to the new version.
createRemoteInstaller	Create an installation program that can be downloaded and used to install EDM at a remote site with pre-configured settings for the current company.
createTransactionTriggers	Create SQL scripts so triggers on application tables at remote sites can generate EDM transactions.
date	Get current date.
dateToNumber	Converts a date to a number
dateToString	Converts a date to a character string
dateToTimestamp	Converts a date to a timestamp
day	Returns the day value of a date
delete	Deletes an object.
deleteTable	Deletes a table.
deleteTableList	Delete a table list.
deleteTransactionFilesFromAmazon	Delete transaction files (*.xml and *.fil) from the Amazon Simple Storage Service (S3).
deployMasterChanges	Deploy master data to application tables in the table map.



mod	Divides two values and returns the remainder
month	Returns the month value of a date
notifyClients	Send a notification to connected clients.
nullValue	Return a null value which can be used in other functions. If you want to set the value of a form control to null, use setFormValueToNull() or setValueToNull().
numberToDate	Converts a number to a date
numberToInt	Returns the integer part of a number string.
numberToString	Converts a number to a string
numberToTime	Converts a number to a time
numberToTimestamp	Converts a number to a timestamp
partitionTable	Partition a table and set a new partition expression and reorganize the rows into the correct partitions.
pow	Raises a number to a power
processForcedTransactions	Receive and apply pending transactions whose effective date has arrived and whose Force flag is true for all sites.
processTransactions	Receive and apply pending transactions from selected sites whose effective date has arrived
publishMasterDataTableList	Publish data from the central database, with open and future transactions rolled back, to .csv files in the given directory.
publishMasterDataTables	Publish data from the central database, with open and future transactions rolled back, to .csv files in the given directory.
publishXCEDDataTableList	Publish data from the central database, with open and future transactions rolled back, to .csv files in the given directory.
publishXCEDDataTables	Publish data from the central database, with open and future transactions rolled back, to .csv files in the given directory.
purgeAuditTrail	Delete audit trail rows that are older than the given number of days.
purgePollData	Delete polled table rows that are older than the given number of days.
queryAvg	Gets the average of the values of a column in a table or query using a where clause.
queryCount	Gets the count of the values of a column in a table or query using a where clause.
queryFirst	Gets the first value of a column in a table or query using a where clause.
queryLast	Gets the last value of a column in a table or query using a where clause.
queryMax	Gets the maximum value of a column in a table or query using a where clause.
queryMin	Gets the minimum value of a column in a table or query using a where clause.
querySum	Gets the sum of the values of a column in a table or query using a where clause.
random	Returns a random number between 0 and the passed number
receiveSiteFilesViaWeb	Receive transaction files over HTTP connection.
receiveTransactionFilesFromAmazon	Receive transaction files from Amazon Simple Storage Service (S3).
receiveTransactionFilesFromJMS	Receive files from the configured JMS queue into the local incoming folder.
reloadCachedData	Reload all data cached in server memory on the server and re-initialize system.
removeAmazonCommitListener	Remove listener that sends outgoing transaction files for selected sites to Amazon as soon as they are committed.

removeTransactionsProcessedListener	Remove listener that evaluates the given expression whenever transaction processing completes.
replaceAll	Replaces each substring of this string that matches the given regular expression with the given replacement.
requestFile	Copy a file to other locations
requestTableListRefresh	Send request to location to send table data to replace existing data
requestTableRefresh	Send request to location to send table data to replace existing data
resetSites	Send transactions to reset remote sites to their newly installed state and commit the transactions.
restoreTestCheckpoint	Disconnect all database connections, restore checkpoint 1, and re-initialize server.
right	Returns the rightmost characters of a string
round	Rounds a number at the specified number of digits past the decimal
runApp	Execute the given command line. For example, to open the config.xml file in notepad execute the command <code>c:\windows\system32\notepad.exe c:\EDMweb\config.xml</code> , and the wait flag to <code>false</code> . To run a batch file you could execute the command <code>cmd /c c:\EDMweb\go.bat</code> and the wait flag to <code>false</code> .
runJUnitTest	Run the JUnit tests in the given class.
runReport	Run a report.
runSQL	Executes an insert, update, delete, or maketable SQL statement.
sendDeleteFileTransaction	Send transaction to delete a file.
sendEvaluateTransaction	Send a transaction to cause the receiver to evaluate an expression.
sendExecuteTransaction	Send transaction to execute an operation system command at selected sites.
sendFile	Send a file to other locations
sendRemoteFiles	Send files listed in remote files table to sites.
sendSiteFilesViaWeb	Send transaction files over HTTP connection.
sendSynchronizeDirectoryTransaction	Send a transaction to synchronize a local directory with a directory at other sites.
sendTableListRefresh	Send table data to locations to replace existing data
sendTableRefresh	Send table data to locations to replace existing data
sendTransactionFilesToAmazon	Send transaction files to Amazon Simple Storage Service (S3).
sendTransactionFilesToJMS	Send outgoing files for the given sites to a JMS queue.
sendTransactionFilesToJMSWithRetry	Send outgoing files for the given sites to a JMS queue. If the transfer fails due to a JMS error, the listener will retry after 15 seconds, 30 seconds, then 1 minute, 2 minutes, 4 minutes, 8 minutes, then every 15 minutes until the transfer works.
setConfigProperty	Set configuration property value.
setLastReceivedTranFileNumberToIncoming	Set the last received transaction file number for a site to the lowest transaction file number in the incoming directory - 1.
setPropertyFileValue	Set the value of a property in a property file.
space	Creates a string of blanks
stringToDate	Converts a date string to a date value
stringToInt	Converts a string to an integer.
stringToNumber	Converts a string to a number

stringToTime	Converts a time string to a time value
stringToTimestamp	Converts a timestamp string to a timestamp value
substring	Returns characters from the middle of a string
synchronizeSitePropertyValues	Import and sync the site property values from an outside data source
time	Gets current time of day
timestamp	Get current date and time.
timestampToDate	Converts a timestamp to a date
timestampToNumber	Converts a timestamp to a number
timestampToString	Converts a timestamp to a character string
timestampToTime	Converts a timestamp to a time
timeToNumber	Converts a time to a number
timeToString	Converts a time to a character string
timeToTimestamp	Converts a time to a timestamp
transferSiteFilesViaFTP	Transfer transaction files over FTP connection.
transferSiteFilesViaRAS	Transfer transaction files over network connection.
transferSiteFilesViaWeb	Transfer transaction files over HTTP connection.
trim	Trims the blanks from both ends of a string
trimLeft	Trims the blanks from the left end of a string
trimRight	Trims the blanks from the right end of a string
updateAmazonDirToMatchLocalDir	Update an Amazon S3 directory to match a local directory.
updateLocalDirToMatchAmazonDir	Update a local directory to match a directory on Amazon S3.
updateLocalDirToMatchWebDir	Update a local directory to match a directory on the EDM web server.
updateTableList	Update a table list.
updateTransferSettings	Save the given transfer host, user ID, password, and company name used to connect to a central EDM server
updateWebDirToMatchLocalDir	Update an directory on the EDM web server to match a local directory.
upper	Changes each letter of a string to upper case
validateTableList	Validate references between tables
validateTables	Validate references between tables
year	Returns the year value of a date

Functions by Category

String Functions

Function	What it does...
ascii	Converts the first character of a string to an ASCII value
asciiToChar	Converts an ASCII code to a character
getVersion	Return the current version of EDM

left	Returns the leftmost characters of a string
len	Gets the length of a string
lower	Changes each letter of a string to lower case
replaceAll	Replaces each substring of this string that matches the given regular expression with the given replacement.
right	Returns the rightmost characters of a string
space	Creates a string of blanks
substring	Returns characters from the middle of a string
trim	Trims the blanks from both ends of a string
trimLeft	Trims the blanks from the left end of a string
trimRight	Trims the blanks from the right end of a string
upper	Changes each letter of a string to upper case

Numeric Functions

Function	What it does...
abs	Get the absolute value of the given number. If the number is null, a DataNumber with the value null is returned. If the number is blank, 0 is returned.
greatest	Returns the greater of two numbers
least	Returns the lesser of two numbers
mod	Divides two values and returns the remainder
pow	Raises a number to a power
random	Returns a random number between 0 and the passed number
round	Rounds a number at the specified number of digits past the decimal

Date Functions

Function	What it does...
date	Get current date.
day	Returns the day value of a date
month	Returns the month value of a date
time	Gets current time of day
timestamp	Get current date and time.
year	Returns the year value of a date

Conversion Functions

Function	What it does...
dateToNumber	Converts a date to a number
dateToString	Converts a date to a character string
dateToTimestamp	Converts a date to a timestamp

numberToDate	Converts a number to a date
numberToInt	Returns the integer part of a number string.
numberToString	Converts a number to a string
numberToTime	Converts a number to a time
numberToTimestamp	Converts a number to a timestamp
stringToDate	Converts a date string to a date value
stringToInt	Converts a string to an integer.
stringToNumber	Converts a string to a number
stringToTime	Converts a time string to a time value
stringToTimestamp	Converts a timestamp string to a timestamp value
timestampToDate	Converts a timestamp to a date
timestampToNumber	Converts a timestamp to a number
timestampToString	Converts a timestamp to a character string
timestampToTime	Converts a timestamp to a time
timeToNumber	Converts a time to a number
timeToString	Converts a time to a character string
timeToTimestamp	Converts a time to a timestamp

Object Functions

Function	What it does...
delete	Deletes an object.
deleteTable	Deletes a table.
getAppObject	Returns application object with specified type and name or null if not found.

Record Functions

Function	What it does...
----------	-----------------

Form Functions

Function	What it does...
----------	-----------------

Lookup Functions

Function	What it does...
queryAvg	Gets the average of the values of a column in a table or query using a where clause.
queryCount	Gets the count of the values of a column in a table or query using a where clause.
queryFirst	Gets the first value of a column in a table or query using a where clause.
queryLast	Gets the last value of a column in a table or query using a where clause.
queryMax	Gets the maximum value of a column in a table or query using a where clause.

queryMin	Gets the minimum value of a column in a table or query using a where clause.
querySum	Gets the sum of the values of a column in a table or query using a where clause.

System Functions

Function	What it does...
addRow	Insert a row in a table for a given list of sites that do not already have the row. For example, when entering a
addSystemEventListener	Add the listener in the given class to the list of system event listeners.
addThisSiteToCentral	Call the central web service to add this remote site to the central database.
attachMasterTables	Attach tables from an existing database so it can be edited
attachTable	Attach tables from an existing database so it can be edited
callEvaluateExpressionWebMethod	Call the EvaluateExpression web method.
createTransactionTriggers	Create SQL scripts so triggers on application tables at remote sites can generate EDM transactions.
deleteTableList	Delete a table list.
enableProfile	Enable profiling.
evaluateExpressionAtRemoteSites	Send an expression to a list of remote sites which will evaluate the expression on the server. Only server-side functions may be included in the expression.
executeDirectSQL	Directly execute the given SQL statement on the given data source and return the results in a DataSet. If query is not a select statement, execute the SQL statement and return empty DataSet. Use caution to avoid exceeding available memory.
exportData	Exports data from a table or query to a text file.
exportSecurityPolicyToXml	Export users, roles, and other security settings to XML file.
getIniValue	Returns value for the given key from the given ini file or null if not found.
getRegistryValue	Returns the value from the Windows registry for the given key, value, and substring.
getReportConfig	Return the ReportConfig stored in the given file.
getServerTime	Returns the current system time on the server in milliseconds since 1/1/1970 UTC.
getSessions	Returns a list of the current active sessions on the server.
getTableAndQueryList	Get a sorted list of tables and queries in this application separated by semicolons.
getTableList	Get a sorted list of tables in this application separated by semicolons.
iif	Returns one of two values bases on expression
importData	Imports a text file to a table.
importFile	Imports a text file to a table.
importTransactionsFromFile	Import transactions from a CSV file.
insertTableList	Insert a new table list.
partitionTable	Partition a table and set a new partition expression and reorganize the rows into the correct partitions.
runApp	Execute the given command line. For example, to open the config.xml file in notepad execute the command <code>c:\windows\system32\notepad.exe c:\EDMweb\config.xml</code> , and the wait flag to <code>false</code> . To run a batch file you could execute the command <code>cmd /c c:\EDMweb\go.bat</code> and the wait flag to <code>false</code> .
runJUnitTest	Run the JUnit tests in the given class.
runReport	Run a report.

runSQL	Executes an insert, update, delete, or maketable SQL statement.
updateTableList	Update a table list.

Enterprise Data Manager Functions

Function	What it does...
addAmazonCommitListener	Add listener that sends outgoing transaction files for selected sites to Amazon as soon as they are committed.
addDeleteAmazonTransactionsOnResetListener	Add listener that deletes transaction files from the Amazon incoming and outgoing directory when a reset site transaction is committed.
addJMSCommitListener	Add listener that sends outgoing transaction files for selected sites to a JMS queue as soon as they are committed.
addJMSReceiveListener	Add listener that receives transaction file messages from a JMS queue and writes them to the incoming directory.
addJMSReceiveListenerWithRetry	Add listener that receives transaction file messages from a JMS queue and writes them to the incoming directory.
addProcessReceivedTransactionsListener	Add listener that automatically processes incoming transactions as soon as they are received.
addTransactionsProcessedListener	Add listener that evaluates the given expression whenever transaction processing completes.
attachRemoteTable	Attach a table from an existing database so it can be edited
changePackageStatus	Change status of the transactions for sites in a package to a new status.
changePassword	Change the current user's password.
commitTransactions	Commit packages of transactions so they are sent to selected locations
copyAndRefreshSiteData	Copy data for selected tables from one site to other sites and refresh the tables at the other sites on the effective date.
copyCompany	Create a new company by copying the configuration of an existing company.
copySiteData	Copy data for selected tables from one location to another location
createRemoteInstaller	Create an installation program that can be downloaded and used to install EDM at a remote site with pre-configured settings for the current company.
deleteTransactionFilesFromAmazon	Delete transaction files (*.xml and *.fil) from the Amazon Simple Storage Service (S3).
exportSelectedDataToFile	Export data to a file from a table
fileExists	Returns true if the given file or directory exists.
generateFunctionHelp	Generate HTML help file for a list of PluginLoader classes.
getConfigProperty	Get configuration property value.
getConfigPropertyWithDefault	Get configuration property value.
getNextUnusedId	Gets the next unused ID for the given table and column, optionally within a range.
getPropertyFileValue	Get the value of a property in a property file.
getValueListOfUsersWithRole	Get a value list of users with a given role separated by semicolons.
loadCentralData	Load schema for a central database table
loadCentralSchema	Load schema for a central database table
notifyClients	Send a notification to connected clients.

nullValue	Return a null value which can be used in other functions. If you want to set the value of a form control to null, use setFormValueToNull() or setValueToNull().
processForcedTransactions	Receive and apply pending transactions whose effective date has arrived and whose Force flag is true for all sites.
processTransactions	Receive and apply pending transactions from selected sites whose effective date has arrived
purgeAuditTrail	Delete audit trail rows that are older than the given number of days.
purgePollData	Delete polled table rows that are older than the given number of days.
receiveSiteFilesViaWeb	Receive transaction files over HTTP connection.
receiveTransactionFilesFromAmazon	Receive transaction files from Amazon Simple Storage Service (S3).
receiveTransactionFilesFromJMS	Receive files from the configured JMS queue into the local incoming folder.
reloadCachedData	Reload all data cached in server memory on the server and re-initialize system.
removeAmazonCommitListener	Remove listener that sends outgoing transaction files for selected sites to Amazon as soon as they are committed.
removeTransactionsProcessedListener	Remove listener that evaluates the given expression whenever transaction processing completes.
requestFile	Copy a file to other locations
requestTableListRefresh	Send request to location to send table data to replace existing data
requestTableRefresh	Send request to location to send table data to replace existing data
resetSites	Send transactions to reset remote sites to their newly installed state and commit the transactions.
restoreTestCheckpoint	Disconnect all database connections, restore checkpoint 1, and re-initialize server.
sendDeleteFileTransaction	Send transaction to delete a file.
sendEvaluateTransaction	Send a transaction to cause the receiver to evaluate an expression.
sendExecuteTransaction	Send transaction to execute an operation system command at selected sites.
sendFile	Send a file to other locations
sendRemoteFiles	Send files listed in remote files table to sites.
sendSiteFilesViaWeb	Send transaction files over HTTP connection.
sendSynchronizeDirectoryTransaction	Send a transaction to synchronize a local directory with a directory at other sites.
sendTableListRefresh	Send table data to locations to replace existing data
sendTableRefresh	Send table data to locations to replace existing data
sendTransactionFilesToAmazon	Send transaction files to Amazon Simple Storage Service (S3).
sendTransactionFilesToJMS	Send outgoing files for the given sites to a JMS queue.
sendTransactionFilesToJMSWithRetry	Send outgoing files for the given sites to a JMS queue. If the transfer fails due to a JMS error, the listener will retry after 15 seconds, 30 seconds, then 1 minute, 2 minutes, 4 minutes, 8 minutes, then every 15 minutes until the transfer works.
setConfigProperty	Set configuration property value.
setLastReceivedTranFileNumberToIncoming	Set the last received transaction file number for a site to the lowest transaction file number in the incoming directory - 1.
setPropertyFileValue	Set the value of a property in a property file.
synchronizeSitePropertyValues	Import and sync the site property values from an outside data source
transferSiteFilesViaFTP	Transfer transaction files over FTP connection.

transferSiteFilesViaRAS	Transfer transaction files over network connection.
transferSiteFilesViaWeb	Transfer transaction files over HTTP connection.
updateAmazonDirToMatchLocalDir	Update an Amazon S3 directory to match a local directory.
updateLocalDirToMatchAmazonDir	Update a local directory to match a directory on Amazon S3.
updateLocalDirToMatchWebDir	Update a local directory to match a directory on the EDM web server.
updateTransferSettings	Save the given transfer host, user ID, password, and company name used to connect to a central EDM server
updateWebDirToMatchLocalDir	Update an directory on the EDM web server to match a local directory.
validateTableList	Validate references between tables
validateTables	Validate references between tables

Master Data Management Functions

Function	What it does...
applyEffectiveMasterDataTrans	Queries the audit table for committed transactions on master data tables whose effective date has arrived and calls the appropriate master data change listener for each transaction then changes the status of the transaction to Successful.
createConfigurationVersionDataSite	Creates a data site to hold version data for configuration version. The ID of the new site will be an unused, negative site number. The name of the site will be the name of the configuration plus a hyphen plus the name of the version. Each table in the configuration type will be added to the list of managed site tables. If there is a previous version of the configuration then the data from the previous version will be copied to the new version.
deployMasterChanges	Deploy master data to application tables in the table map.
getMasterTableConfigurationType	Gets the Configuration Type for the given table with the given name.
getUnlockedPackageVersions	Get a ValueList of version ids and names that are not referenced by unlocked packages.
publishMasterDataTableList	Publish data from the central database, with open and future transactions rolled back, to .csv files in the given directory.
publishMasterDataTables	Publish data from the central database, with open and future transactions rolled back, to .csv files in the given directory.
publishXCEDDataTableList	Publish data from the central database, with open and future transactions rolled back, to .csv files in the given directory.
publishXCEDDataTables	Publish data from the central database, with open and future transactions rolled back, to .csv files in the given directory.

abs

Get the absolute value of the given number. If the number is null, a DataNumber with the value null is returned. If the number is blank, 0 is returned.

Syntax:

abs('number')

Parameters:

number - Number whose absolute value is to be returned.

Returns:

Absolute value of number

Class:

EDMcommon.function.CommonFunctions

addAmazonCommitListener

Add listener that sends outgoing transaction files for selected sites to Amazon as soon as they are committed.

Syntax:

```
addAmazonCommitListener('siteIdList', 'bucketName', 'prefix', 'encryptedCredentials')
```

Parameters:

siteIdList - Comma-separated list of site id numbers (* means all sites).

bucketName - Name of Amazon S3 bucket containing incoming and outgoing directories.

prefix - Common prefix of all transaction file names to be transferred without the site ID path segment, e.g. 'outgoing/'.

encryptedCredentials - AWS credentials encrypted with the EncryptAmazonCredentialsCmd object.

Returns:

Void.

Class:

EDMserver.function.AddAmazonCommitListenerFunction

addDeleteAmazonTransactionsOnResetListener

Add listener that deletes transaction files from the Amazon incoming and outgoing directory when a reset site transaction is committed.

Syntax:

```
addDeleteAmazonTransactionsOnResetListener('siteIdList', 'bucketName', 'incomingPrefix', 'outgoingPrefix', 'encryptedCredentials')
```

Parameters:

siteIdList - Comma-separated list of site IDs whose Amazon files will be deleted if they are reset. (blank to prompt user, * for all sites).

bucketName - Name of Amazon S3 bucket containing incoming and outgoing folders.

incomingPrefix - Prefix of all incoming transaction file names to be deleted without the site ID path segment, e.g. "incoming/".

outgoingPrefix - Prefix of all outgoing transaction file names to be deleted without the site ID path segment, e.g. "outgoing/".

encryptedCredentials - AWS credentials encrypted with the EncryptAmazonCredentialsCmd object.

Returns:

Void.

Class:

EDMserver.function.AddDeleteAmazonTransOnResetListenerFunction

addJMSCommitListener

Add listener that sends outgoing transaction files for selected sites to a JMS queue as soon as they are committed.

Syntax:

```
addJMSCommitListener('siteIdList', 'queueName', 'brokerURL', 'userId', 'password')
```

Parameters:

siteIdList - Comma-separated list of site id numbers (* means all sites).

queueName - Name of the queue where outgoing transaction files should be sent. For example: EDM.outgoing.%REMOTESITEID%

brokerURL - URL used to connect to the JMS broker. For example: tcp://localhost:61616

userId - User ID used to connect to the JMS broker.
password - Password used to connect to the JMS broker.

Returns:

Void.

Class:

EDMserver.function.AddJMSCommitListenerFunction

addJMSReceiveListener

Add listener that receives transaction file messages from a JMS queue and writes them to the incoming directory.

Syntax:

```
addJMSReceiveListener('queueName', 'brokerURL', 'userId', 'password')
```

Parameters:

queueName - Name of the queue where incoming transaction files are received. For example: EDM.incoming
brokerURL - URL used to connect to the JMS broker. For example: tcp://localhost:61616
userId - User ID used to connect to the JMS broker.
password - Password used to connect to the JMS broker.

Returns:

Void.

Class:

EDMserver.function.AddJMSReceiveListener

addJMSReceiveListenerWithRetry

Add listener that receives transaction file messages from a JMS queue and writes them to the incoming directory.

Syntax:

```
addJMSReceiveListenerWithRetry('queueName', 'brokerURL', 'userId', 'password', retrySeconds)
```

Parameters:

queueName - Name of the queue where incoming transaction files are received. For example: EDM.incoming
brokerURL - URL used to connect to the JMS broker. For example: tcp://localhost:61616
userId - User ID used to connect to the JMS broker.
password - Password used to connect to the JMS broker.
retrySeconds - Number of seconds we waited before previous retry if the call fails. 0 means this is the first retry.

Returns:

Void.

Class:

EDMserver.function.AddJMSReceiveListener

addProcessReceivedTransactionsListener

Add listener that automatically processes incoming transactions as soon as they are received.

Syntax:

addProcessReceivedTransactionsListener()

Parameters:

None.

Returns:

Void.

Class:

EDMserver.function.AddProcessReceivedTransactionsListenerFunction

addRow

Insert a row in a table for a given list of sites that do not already have the row. For example, when entering a

product, this function can be called in the AfterInsert event to add a row to the product type table. For example, to add rows to the TNG_dbo_ProductTypes table after the user inserts a row in the TNG_dbo_Products table, set the **afterInsert** property of the Product form to: "**=addRow('TNG_dbo_ProductTypes', getNamedControlValues('id,name,description'), getSelectedSites())"**

Syntax:

addRow('tableName', 'namedColumnValues', 'siteIdList')

Parameters:

tableName - Name of a table.

namedColumnValues - List of column names and values of the form: "name1","value1","name2","value2"...

siteIdList -

Returns:

Void.

Class:

EDMserver.cmd.AddRowCmdExecutor

addSystemEventListener

Add the listener in the given class to the list of system event listeners.

Syntax:

addSystemEventListener('listenerClassName')

Parameters:

listenerClassName - Name of listener class.

Returns:

Void.

Class:

EDMserver.cmd.AddSystemEventListenerCmdExecutor

addThisSiteToCentral

Call the central web service to add this remote site to the central database.

If the site does not exist in the central database it will be added. If it is the first site in the company, a table list will be requested. If the site data

has not been loaded then a table refresh will be requested.

Syntax:

```
addThisSiteToCentral( 'url', 'userId', 'password')
```

Parameters:

url - URL of the central server such as <http://xpress.xpient.com/EDM>.

userId - User ID used to log in to central system.

password - Password used to log into central system.

Returns:

Void.

Class:

EDMserver.cmd.AddThisSiteToCentralCmdExecutor

addTransactionsProcessedListener

Add listener that evaluates the given expression whenever transaction processing completes.

Syntax:

```
addTransactionsProcessedListener( 'expression')
```

Parameters:

expression - Expression to evaluate after transaction processing completes.

Returns:

Void.

Class:

EDMserver.function.AddTransactionsProcessedListenerFunction

applyEffectiveMasterDataTrans

Queries the audit table for committed transactions on master data tables whose effective date has arrived and calls the appropriate master data change listener for each transaction then changes the status of the transaction to Successful.

Syntax:

```
applyEffectiveMasterDataTrans()
```

Parameters:

None.

Returns:

Void.

Class:

EDMserver.plugin.mdm.cmd.ApplyEffectiveMasterDataTransCmdExecutor

ascii

Converts the first character of a string to an ASCII value

Syntax:

ascii('astring')

Parameters:

astring - String whose first character is to be converted

Returns:

ASCII value of the first character of the string

Class:

EDMcommon.function.CommonFunctions

asciiToChar

Converts an ASCII code to a character

Syntax:

asciiToChar(asciiCode)

Parameters:

asciiCode - ASCII code to convert

Returns:

Character that is assigned to the ASCII code

Class:

EDMcommon.function.CommonFunctions

attachMasterTables

Attach tables from an existing database so it can be edited

Attaches master tables by creating a central table with the CDMLOCID column added to the table and all indexes. Adds a remote table record for the central site with a version of '1'. Adds a remote table record for every remote site with a version of null so transactions are not generated to the site. Adds a remote data source version for the central site of '1'. Adds a remote data source version for every remote site of '0'. Adds the tables to a table list called 'Attached Master Tables'. Adds a data source version 0 to the convert file and a version 1 which adds all the tables to the convert file.

Syntax:

attachMasterTables('tableNames', 'dataSourceName')

Parameters:

tableNames - Comma-delimited list of table names to attach.

dataSourceName - Name of the data source in the configuration that contains the tables.

Returns:

Void.

Class:

EDMserver.plugin.mdm.cmd.AttachMasterTablesCmdExecutor

attachRemoteTable

Attach a table from an existing database so it can be edited

Syntax:

```
attachRemoteTable( 'tableName', 'dataSourceName', 'tableVersion', ignoreInserts, ignoreUpdates, ignoreDeletes)
```

Parameters:

tableName - Name of table to be attached.

dataSourceName - Name of data source from which to attach the table.

tableVersion - Version number of table to be attached.

ignoreInserts - True if central database should not accept inserts from remote location.

ignoreUpdates - True if central database should not accept updates from remote location.

ignoreDeletes - True if central database should not accept deletes from remote location.

Returns:

Void.

Class:

EDMserver.cmd.AttachTableCmdExecutor

attachTable

Attach tables from an existing database so it can be edited

Syntax:

```
attachTable( 'tableNames', 'dataSourceName')
```

Parameters:

tableNames - Comma-separated list of table names to be attached.

dataSourceName - Name of data source from which to attach the tables.

Returns:

Void.

Class:

EDMserver.cmd.AttachTableCmdExecutor

callEvaluateExpressionWebMethod

Call the EvaluateExpression web method.

Syntax:

```
callEvaluateExpressionWebMethod( 'expression', 'userId', 'encryptedPassword', 'company', 'wsdlUrl')
```

Parameters:

expression - Expression to evaluate. For example: processTransactions("**")

userId - ID of the user making the call.

encryptedPassword - Encrypted password for the user. Can be found in policy.xml.

company - Name of company where expression is to be evaluated.

wsdlUrl - URL of the WSDL for the web service.

Returns:

String

Class:

EDMserver.cmd.CallEvaluateExpressionWebMethodCmdExecutor

changePackageStatus

Change status of the transactions for sites in a package to a new status.

Syntax:

changePackageStatus('oldStatus', 'newStatus', 'packageName', 'siteIdList')

Parameters:

oldStatus - Transactions with this status type will have their status changed (Open, Committed, Sent, Received, Successful, Failed, Closed, or blank to prompt user to select).

newStatus - Transactions will be changed to this status type (Open, Committed, Sent, Received, Successful, Failed, Closed, or blank to prompt user to select).

packageName - Name of package containing transactions to change (blank means prompt user to select).

siteIdList - Comma-delimited list of site IDs whose transactions are to be changed (blank means prompt user to select).

Returns:

Void.

Class:

EDMserver.cmd.ChangePackageStatusCmdExecutor

changePassword

Change the current user's password.

Syntax:

changePassword('oldPassword', 'newPassword')

Parameters:

oldPassword - Current user's password encrypted and converted to string of hex digits.

newPassword - New password encrypted and converted to string of hex digits.

Returns:

Void.

Class:

EDMserver.cmd.ChangePasswordCmdExecutor

commitTransactions

Commit packages of transactions so they are sent to selected locations

Syntax:

commitTransactions('packageNameList', 'locationIdList')

Parameters:

packageNameList - Semicolon separated list of package names to commit. * means all sites.

locationIdList - Semicolon separated list of location numbers to commit packages. * means all packages.

Returns:

Void.

Class:

EDMserver.cmd.CommitCmdExecutor

copyAndRefreshSiteData

Copy data for selected tables from one site to other sites and refresh the tables at the other sites on the effective date.

Syntax:

copyAndRefreshSiteData('fromSiteId', 'toLocations', 'tableList', overwrite, copySharing, 'effectiveDate', forceProcessing, autoCommit)

Parameters:

fromSiteId - Site id to copy from.

toLocations - Comma-separated list of site IDs to copy to.

tableList - Name of a table list or a comma-separated list of table names to copy.

overwrite - True if destination site data should be overwritten if it exists.

copySharing - True if copied tables that are shared by the source site should be shared by the destination site.

effectiveDate - Date/time when data will be copied and refresh transaction sent out (local-specific format).

forceProcessing - True if transaction should be processed when only processing forced transactions.

autoCommit - True if refresh transaction should be committed as soon as it is generated.

Returns:

Void.

Class:

EDMserver.cmd.CopyAndRefreshSiteDataCmdExecutor

copyCompany

Create a new company by copying the configuration of an existing company.

<p>A unique company ID is calculated by removing all characters from the company name except letters, digits, and underscores and adding digits if needed to make the name unique. <p>The configuration files for the company are copied from the sub-directory of the classes directory with the same name as the template company ID. <p>The company database is named EDMHQ_ + companyId. <p>NOTE: The system data source in the config file template must not contain a connect string because it contains the database name which is not replaced. <p>The company's security information (users, roles, etc) will be loaded from the policy file in the default company files directory. <p>There must be a role called 'Company Administrator' in the policy.xml file.

Syntax:

copyCompany('companyName', 'userName', 'userId', 'password', 'passwordConfirmation', 'templateCompanyName', 'authorizationClassName', 'authorizationParameters')

Parameters:

companyName - Name of the company to create.

userName - User's name.

userId - Login user ID which will be given the 'Company Administrator' role. Null or blank means don't create a user.

password - Login password.

passwordConfirmation - Login password confirmation.

templateCompanyName - Name of the existing company whose configuration will be copied to the new company.

authorizationClassName - Name of the class used to authorize the customer to create a company based on the customer ID.

authorizationParameters - Delimited ascii string of parameters sent to authorization method to determine if the user is authorized to create a company. Each parameter must be surrounded by double quotes, use two double quote characters to represent a double quote within the parameter.

Returns:

Unique ID of new company which can be used for company-specific directory or queue names.

Class:

EDMserver.cmd.CopyCompanyCmdExecutor

copySiteData

Copy data for selected tables from one location to another location

Syntax:

copySiteData('fromSiteId', 'toLocations', 'tableList', overwrite, copySharing)

Parameters:

fromSiteId - Site id to copy from.

toLocations - Comma-separated list of site IDs to copy to.

tableList - Name of a table list or a comma-separated list of table names to copy.

overwrite - True if destination site data should be overwritten if it exists.

copySharing - True if copied tables that are shared by the source site should be shared by the destination site.

Returns:

Void.

Class:

EDMserver.cmd.CopySiteDataCmdExecutor

createConfigurationVersionDataSite

Creates a data site to hold version data for configuration version. The ID of the new site will be an unused, negative site number. The name of the site will be the name of the configuration plus a hyphen plus the name of the version. Each table in the configuration type will be added to the list of managed site tables. If there is a previous version of the configuration then the data from the previous version will be copied to the new version.

Syntax:

createConfigurationVersionDataSite(configurationTypeId, configurationType, versionId)

Parameters:

configurationTypeId - ID of the configuration type for which the version data site is being created.

configurationType - ID of the configuration for which the version data site is being created.

versionId - ID of the version for which the version data site is being created.

Returns:

Site number of the newly created version data site.

Class:

EDMserver.plugin.mdm.cmd.CreateConfigVersionDataSiteCmdExecutor

createRemoteInstaller

Create an installation program that can be downloaded and used to install EDM at a remote site with pre-configured settings for the current company.

<p>The function copies CompileInfo.txt from webapps/EDM/company/companyId/RemoteInstaller to webapps/EDM/WEB-INF/RemoteInstaller then executes webapps/EDM/WEB-INF/RemoteInstaller/build.bat. The CompileInfo.txt file in each company directory contains company-specific information needed to create the remote installation program.</p><p>See the Xpient wiki page "Automated Setup of New Remote Sites" in the "EDM Documentation" section for more information.</p>

Syntax:

createRemoteInstaller()

Parameters:

None.

Returns:

URL of page where installer can be downloaded.

Class:

EDMserver.cmd.CreateRemoteInstallerCmdExecutor

createTransactionTriggers

Create SQL scripts so triggers on application tables at remote sites can generate EDM transactions.

Creates a SQL script called EDM_Transaction_Functions.sql that creates functions in the EDM database that can be called by triggers in the remote application database to generate transactions. Also creates a script called App_Transaction_Triggers.sql to create triggers on the tables in the given table list that generate transactions in the EDM audit table on insert, update, and delete. Currently this function only supports SQL Server databases.

Syntax:

```
createTransactionTriggers( 'tableListName')
```

Parameters:

tableListName - Name of the table list containing the names of the tables to which triggers should be added.

Returns:

Void.

Class:

EDMserver.cmd.CreateTransactionTriggersCmdExecutor

date

Get current date.

Syntax:

```
date( 'format')
```

Parameters:

format - Format string describing how the date should be formatted.

Returns:

String containing current date.

Class:

EDMcommon.function.CommonFunctions

dateToNumber

Converts a date to a number

Syntax:

dateToNumber('date')

Parameters:

date - Date to convert

Returns:

Number representing the date

Class:

EDMcommon.function.CommonFunctions

dateToString

Converts a date to a character string

Syntax:

dateToString('date', 'format')

Parameters:

date - Date to convert

format - Format string to use

Returns:

String representation of the date

Class:

EDMcommon.function.CommonFunctions

dateToTimestamp

Converts a date to a timestamp

Syntax:

dateToTimestamp('date')

Parameters:

date - Date to convert

Returns:

Timestamp representing the date with the time set to 00:00:00

Class:

EDMcommon.function.CommonFunctions

day

Returns the day value of a date

Syntax:

day('date')

Parameters:

date - Date to get the day from

Returns:

Day number (1-31)

Class:

EDMcommon.function.CommonFunctions

delete

Deletes an object.

Syntax:

delete('objectType', 'objectName')

Parameters:

objectType - Type of object (table, query, form, report or function).

objectName - Name of the object.

Returns:

Void.

Class:

EDMserver.ServerFunctions

deleteTable

Deletes a table.

Syntax:

deleteTable('tableName', deletePhysicalTable)

Parameters:

tableName - Name of the table.

deletePhysicalTable - True if system should delete the physical table from the database as well.

Returns:

Void.

Class:

EDMserver.ServerFunctions

deleteTableList

Delete a table list.

Syntax:

deleteTableList('tableListName')

Parameters:

tableListName - Name of table list to delete.

Returns:

Deleted table list or null if not found.

Class:

EDMserver.cmd.EditSnapshotTablesCmdExecutor

deleteTransactionFilesFromAmazon

Delete transaction files (*.xml and *.fil) from the Amazon Simple Storage Service (S3).

All files with names of the form **prefix/siteid/.xml and prefix/siteid/.fil** will be deleted from S3. For example, if the S3 bucket contains incoming/1/5.xml and outgoing/1/5.xml and the prefix is 'outgoing' then the outgoing/1/5.xml file will be deleted. The system will add '/' to the end prefix if it is not there and will convert any '\' characters in the prefix to '/' so AWS management console sees path segments as folders. The system will also convert the prefix to all lower case to make it case-insensitive.

Syntax:

```
deleteTransactionFilesFromAmazon( 'siteIdList', 'bucketName', 'prefix', 'encryptedCredentials')
```

Parameters:

siteIdList - Comma-separated list of site IDs to delete (blank to prompt user, * for all locations).

bucketName - Name of Amazon S3 bucket containing incoming and outgoing folders.

prefix - Common prefix of all transaction file names to be transferred without the site ID path segment, e.g. "outgoing/".

encryptedCredentials - AWS credentials encrypted with the EncryptAmazonCredentialsCmd object.

Returns:

Void

Class:

EDMserver.cmd.DeleteTransactionFilesFromAmazonCmdExecutor

deployMasterChanges

Deploy master data to application tables in the table map.

Syntax:

```
deployMasterChanges( 'deployTypeId', setAppliedToNull, 'versionId', 'appIdList')
```

Parameters:

deployTypeId - ID of the type of deployment being made which determines which site property and value is used to identify sites included in the deployment and the data sources and schemas used for the application tables.

setAppliedToNull - True if the applied date of the subscription records for sites included in the deployment should be set to null to force deployment of all tables.

versionId - ID of the version to be deployed, null or blank means deploy data for all subscribed versions.

appIdList - Comma-delimited list of app ID's to deploy or "*" to deploy to all apps.

Returns:

Number of sites that were deployed.

Class:

EDMserver.plugin.mdm.cmd.DeployMasterChangesCmdExecutor

enableProfile

Enable profiling.

Syntax:

enableProfile()

Parameters:

None.

Returns:

True if profiling was previously enable.

Class:

EDMserver.cmd.EnableProfileCmdExecutor

evaluateExpressionAtRemoteSites

Send an expression to a list of remote sites which will evaluate the expression on the server. Only server-side functions may be included in the expression.

Syntax:

evaluateExpressionAtRemoteSites('expression', 'siteIdList')

Parameters:

expression - Expression to evaluate. For example: processTransactions("**")
siteIdList - Comma-delimited list of remote site IDs that are to evaluate the expression.

Returns:

Void.

Class:

EDMserver.function.EvaluateExpressionAtRemoteSitesFunction

executeDirectSQL

Directly execute the given SQL statement on the given data source and return the results in a DataSet. If query is not a select statement, execute the SQL statement and return empty DataSet. Use caution to avoid exceeding available memory.

Syntax:

executeDirectSQL('dataSourceName', 'sql')

Parameters:

dataSourceName - Name of the EDM data source where the query is to execute.
sql - SQL statement to execute.

Returns:

DataSet

Class:

EDMserver.cmd.ExecuteQueryCmdExecutor

exportData

Exports data from a table or query to a text file.

Syntax:

```
exportData( 'filename', addNameRow, isDelimited, 'delimiter', 'columnLengths', 'dateFormat', 'source')
```

Parameters:

filename - Name of the text file in which to store the data.
addNameRow - T/F should the first row of the text file contain column names?.
isDelimited - T/F should the text file be delimited (as opposed to fixed length fields)?.
delimiter - Character that separates the fields.
columnLengths - Length of each field separated by semicolons ';' for fixed length fields.
dateFormat - Format string that describes how dates should appear in the text file.
source - Name of the table or query to export.

Returns:

Void.

Class:

EDMserver.ServerFunctions

exportSecurityPolicyToXml

Export users, roles, and other security settings to XML file.

Syntax:

```
exportSecurityPolicyToXml( 'file')
```

Parameters:

file - File path and name.

Returns:

Void.

Class:

EDMserver.cmd.ExportSecurityPolicyToXmlCmdExecutor


exportSelectedDataToFile

Export data to a file from a table

Syntax:

```
exportSelectedDataToFile( 'siteList', useShareGroupNumbers, 'filePathAndName', addNameRow, isDelimited, 'delimiter', 'columnLengths', 'dateFormat', 'source')
```

Parameters:

siteList - Either a star  or a comma delimited list of sites to export
useShareGroupNumbers - Boolean on whether or not to use the share group numbers instead of the real location ids.
filePathAndName - File path and name of file to be exported to.
addNameRow - Boolean to show or hide the column names as the first row.
isDelimited - Boolean – whether or not the data should be delimited. This will override the column widths if both are set.
delimiter - The string character – either a comma or semicolon – that is used to delimit columns.
columnLengths - A comma delimited list of the widths of the columns for a fixed length export.
dateFormat - A string of the format to use for dates. Defaults to GDT.
source - The table name or query name to be exported.

Returns:

Void.

Class:

EDMserver.ServerFunctions

fileExists

Returns true if the given file or directory exists.

Syntax:

fileExists('filePath')

Parameters:

filePath - Path and name of the file to check.

Returns:

Boolean.

Class:

EDMserver.function.FileExistsFunction

generateFunctionHelp

Generate HTML help file for a list of PluginLoader classes.

Syntax:

generateFunctionHelp('pluginLoaderClasses', 'destinationHtmlFile', isClassListForClient)

Parameters:

pluginLoaderClasses - Comma-delimited list of PluginLoader class names whose functions should be included in the help file.

destinationHtmlFile - Path and name of HTML help file to generate.

isClassListForClient - True if the functions are in client side classes, else False if they are in server side classes.

Returns:

Void

Class:

EDMserver.cmd.GenerateFunctionHelpCmdExecutor

getAppObject

Returns application object with specified type and name or null if not found.

Syntax:

getAppObject('objectType', 'objectName')

Parameters:

objectType - Type of application object (table, query, form, report).

objectName - Name of application object.

Returns:

Application object or null if not found.

Class:

EDMserver.cmd.GetAppObjectCmdExecutor

getConfigProperty

Get configuration property value.

Get value of the property in the configuration file with the given case-sensitive name or null if not found.

Syntax:

```
getConfigProperty( 'propertyName')
```

Parameters:

propertyName - Name of the property.

Returns:

Property value or null if not found.

Class:

EDMserver.cmd.GetConfigPropertyCmdExecutor

getConfigPropertyWithDefault

Get configuration property value.

Get value of the property in the configuration file with the given case-sensitive name or the given default value if not found.

Syntax:

```
getConfigPropertyWithDefault( 'propertyName', 'defaultValue')
```

Parameters:

propertyName - Name of the property.

defaultValue - Default value to return if property not found.

Returns:

Property value or the given default value if not found.

Class:

EDMserver.cmd.GetConfigPropertyCmdExecutor

getIniValue

Returns value for the given key from the given ini file or null if not found.

Syntax:

```
getIniValue( 'iniFileName', 'key')
```

Parameters:

iniFileName - Path and name of ini file. Path may be absolute or relative.

key - Key whose value is to be returned.

Returns:

Ini value or null if not found.

Class:

EDMserver.cmd.GetIniFileValueCmdExecutor

getMasterTableConfigurationType

Gets the Configuration Type for the given table with the given name.

Syntax:

getMasterTableConfigurationType('tableName')

Parameters:

tableName - The name of the table to check.

Returns:

The ConfigurationType for the given table with the given name.

Class:

EDMserver.plugin.mdm.cmd.GetMasterTableConfigurationTypeCmdExecutor

getNextUnusedId

Gets the next unused ID for the given table and column, optionally within a range.

Syntax:

getNextUnusedId('tableName', 'columnName', 'minValue', 'maxValue')

Parameters:

tableName - The name of the table containing the column.

columnName - The name of the column to return the next ID value for.

minValue - The minimum value to return. If not supplied, will use 1.

maxValue - The maximum value to return. If not supplied, will use Integer.MAX_VALUE.

Returns:

Integer.

Class:

EDMserver.cmd.GetNextUnusedIdCmdExecutor

getPropertyFileValue

Get the value of a property in a property file.

Syntax:

getPropertyFileValue('propertyName', 'propertyFileName')

Parameters:

propertyName - Name of the property whose value is to be returned.

propertyFileName - Path and name of property file.

Returns:

Property value

Class:

EDMserver.cmd.GetPropertyFileValueCmdExecutor

getRegistryValue

Returns the value from the Windows registry for the given key, value, and substring.

Syntax:

getRegistryValue('key', 'value', 'substring')

Parameters:

key - Registry key to use in the HKEY_LOCAL_MACHINE section of the registry.

value - Value in the given key to retrieve.

substring - Portion of registry value to return when value is of the form 'substring1=value1;substring2=value2;...'. Ignored if null or blank.

Returns:

Registry value.

Class:

EDMserver.ServerFunctions

getReportConfig

Return the ReportConfig stored in the given file.

Syntax:

getReportConfig('settingFile')

Parameters:

settingFile - File containing the report configuration.

Returns:

ReportConfig.

Class:

EDMserver.cmd.GetReportConfigCmdExecutor

getServerTime

Returns the current system time on the server in milliseconds since 1/1/1970 UTC.

Syntax:

getServerTime()

Parameters:

None.

Returns:

Integer.

Class:

EDMserver.cmd.GetServerTimeCmdExecutor

getSessions

Returns a list of the current active sessions on the server.

Syntax:

getSessions()

Parameters:

None.

Returns:

String.

Class:

EDMserver.cmd.GetSessionsCmdExecutor

getTableAndQueryList

Get a sorted list of tables and queries in this application separated by semicolons.

Syntax:

getTableAndQueryList()

Parameters:

None.

Returns:

Sorted list of tables and queries separated by semicolons.

Class:

EDMserver.ServerFunctions

getTableList

Get a sorted list of tables in this application separated by semicolons.

Syntax:

getTableList()

Parameters:

None.

Returns:

Sorted list of tables and queries separated by semicolons.

Class:

EDMserver.ServerFunctions

getUnlockedPackageVersions

Get a ValueList of version ids and names that are not referenced by unlocked packages.

Syntax:

```
getUnlockedPackageVersions( 'packageId')
```

Parameters:

packageId - The ID of the package that will be included in the list, regardless whether or not it is unlocked. For example, the package currently being edited.

Returns:

String.

Class:

EDMserver.plugin.mdm.function.GetUnlockedPackageVersionsFunction

getValueListOfUsersWithRole

Get a value list of users with a given role separated by semicolons.

Syntax:

```
getValueListOfUsersWithRole( 'roleName')
```

Parameters:

roleName - Name of role.

Returns:

String.

Class:

EDMserver.cmd.GetValueListOfUsersWithRoleCmdExecutor

getVersion

Return the current version of EDM

Syntax:

```
getVersion()
```

Parameters:

None.

Returns:

The current version of EDM

Class:

EDMcommon.function.CommonFunctions

greatest

Returns the greater of two numbers

Syntax:

```
greatest( 'firstNumber', 'secondNumber')
```

Parameters:

firstNumber - First number to compare

secondNumber - Second number to compare

Returns:

Greater of two numbers

Class:

EDMcommon.function.CommonFunctions

iif

Returns one of two values bases on expression

Syntax:

```
iif( 'expression', 'trueValue', 'falseValue')
```

Parameters:

expression - Expression to evaluate

trueValue - Value to return if the expression is TRUE

falseValue - Value to return if the expression is FALSE

Returns:

First value if expression is true, otherwise second value

Class:

EDMcommon.function.CommonFunctions

importData

Imports a text file to a table.

Syntax:

```
importData( 'filename', HasNameRow, isDelimited, 'delimiter', 'columnLengths', 'dateFormat', 'tableName')
```

Parameters:

filename - Name of the text file to import.

HasNameRow - T/F does the first row of the text file contain column names?.

isDelimited - T/F is the text file delimited (as opposed to fixed length fields)?.

delimiter - Character that separates the fields.

columnLengths - Length of each field separated by semicolons ';' for fixed length fields.

dateFormat - Format string that describes how dates appear in the text file.

tableName - Name of the table to which the data is to be added.

Returns:

Void.

Class:

EDMserver.ServerFunctions

importFile

Imports a text file to a table.

Syntax:

importFile('filename')

Parameters:

filename - Name of the text file to import.

Returns:

Void.

Class:

EDMserver.ServerFunctions

importTransactionsFromFile

Import transactions from a CSV file.

Syntax:

importTransactionsFromFile('filename')

Parameters:

filename - Name of the CSV file on the server containing transaction information.

Returns:

Number of transactions imported.

Class:

EDMserver.cmd.ImportTransactionsFromFileCmdExecutor

insertTableList

Insert a new table list.

Syntax:

insertTableList(tableList, index)

Parameters:

tableList - New table list to insert.

index - 0-based index where table list should be inserted among other table lists, if index is < 0 or >= list size, the new table list is added to the end of the list.

Returns:

Void.

Class:

least

Returns the lesser of two numbers

Syntax:

least('First number', 'Second number')

Parameters:

First number - First number to compare

Second number - Second number to compare

Returns:

Lesser of two numbers

Class:

EDMcommon.function.CommonFunctions

left

Returns the leftmost characters of a string

Syntax:

left('String', Count)

Parameters:

String - String to get the characters from

Count - Number of characters to get

Returns:

Requested characters

Class:

EDMcommon.function.CommonFunctions

len

Gets the length of a string

Syntax:

len('String')

Parameters:

String - String you want the length of

Returns:

Length of the string

Class:

EDMcommon.function.CommonFunctions

loadCentralData

Load schema for a central database table

Syntax:

```
loadCentralData( 'tableNames', locationNumber, 'dataSourceName')
```

Parameters:

tableNames - Comma-separated list of table names whose data is to be loaded.

locationNumber - Location number of the location whose data is being loaded.

dataSourceName - Name of data source from which to load table data.

Returns:

Void.

Class:

EDMserver.ServerFunctions

loadCentralSchema

Load schema for a central database table

Syntax:

```
loadCentralSchema( 'tableName', 'dataSourceName', 'tableVersion')
```

Parameters:

tableName - Name of table whose schema is to be loaded

dataSourceName - Name of data source from which to load table schema

tableVersion - Version number of table

Returns:

Void.

Class:

EDMserver.cmd.AttachTableCmdExecutor

lower

Changes each letter of a string to lower case

Syntax:

```
lower( 'string')
```

Parameters:

string - String to convert

Returns:

String converted to lower case

Class:

EDMcommon.function.CommonFunctions

mod

Divides two values and returns the remainder

Syntax:

```
mod( 'firstValue', 'secondValue')
```

Parameters:

firstValue - Value to divide

secondValue - Value to divide by

Returns:

Remainder after division.

Class:

EDMcommon.function.CommonFunctions

month

Returns the month value of a date

Syntax:

```
month( 'date')
```

Parameters:

date - Date to get the month from (local-specific format)

Returns:

Month number (1-12)

Class:

EDMcommon.function.CommonFunctions

notifyClients

Send a notification to connected clients.

Syntax:

```
notifyClients( 'title', 'message', 'delayMinutes')
```

Parameters:

title - Title of message to display to client (blank or ? means prompt user).

message - Message to display to client (blank or ? means prompt user).

delayMinutes - How many minutes to delay before sending message to connected clients.

Returns:

Void.

Class:

EDMserver.cmd.NotifyClientsCmdExecutor

nullValue

Return a null value which can be used in other functions. If you want to set the value of a form control to null, use setFormValueToNull() or setValueToNull().

Syntax:

```
nullValue()
```

Parameters:

None.

Returns:

Null.

Class:

EDMcommon.function.NullValueFunction

numberToDate

Converts a number to a date

Syntax:

```
numberToDate( 'number')
```

Parameters:

number - Number to convert

Returns:

Date value

Class:

EDMcommon.function.CommonFunctions

numberToInt

Returns the integer part of a number string.

Syntax:

```
numberToInt( 'number')
```

Parameters:

number - Number string to convert

Returns:

Integer value.

Class:

EDMcommon.function.CommonFunctions

numberToString

Converts a number to a string

Syntax:

```
numberToString( 'number', 'Format')
```

Parameters:

number - Number to convert
Format - Format string to use

Returns:

String representation of the number

Class:

EDMcommon.function.CommonFunctions

numberToTime

Converts a number to a time

Syntax:

```
numberToTime( 'number')
```

Parameters:

number - Number to convert

Returns:

Time value

Class:

EDMcommon.function.CommonFunctions

numberToTimestamp

Converts a number to a timestamp

Syntax:

```
numberToTimestamp( 'number')
```

Parameters:

number - Number to convert

Returns:

Timestamp value

Class:

EDMcommon.function.CommonFunctions

partitionTable

Partition a table and set a new partition expression and reorganize the rows into the correct partitions.

<p>The audit table can now be partitioned across multiple database servers which process queries in parallel for much faster performance.

Transaction records are divided among the partitions based on a partitioning expression. Queries on the audit table are sent to each partition server for parallel processing and the results are merged by EDM so that the partitioning is transparent to EDM users. The primary partition will continue to reside in the EDM central database, additional partitions can be stored on other servers. To partition the audit table into three parts take the following steps:

- Backup your central database and the web application folder (webapps/EDM)
- Add new data sources named 'Partition2' and 'Partition3' to the config.xml with the connection information to the new partition servers.
- Restart the web server or reload the cached data so the server will recognize the new data sources.
- Add a menu option with the action: PartitionTable('CDMAudit', 'Partition2,Partition3', 'if(FromLoc = 0, ToLoc % 3, FromLoc % 3)')
- Click on the new menu option and watch the progress of the partitioning on the status bar.

After partitioning the audit table, if you have regular backups of your system database, you should include the new partition databases in your backup schedule.

Syntax:

partitionTable('tableName', 'partitionDataSourceNames', 'partitionExpression')

Parameters:

tableName - Name of the table which is to have a new partition added.

partitionDataSourceNames - Comma-delimited list of data source names where the table partitions are to be created.

partitionExpression - Expression used to determine which partition a row is in by calculating a number between 0 and the number of partition data source names. For example, if you specify 1 partition data source name 'Part1' and want rows with even CDMLOCID's in the primary table and rows with odd CDMLOCID's in the partition then the expression would be 'CDMLOCID % 2' which will return 0 for even CDMLOCID's and 1 for odd CDMLOCID's. If the expression returns a number higher than the number of partition data sources for a row, then the row will be in the main table.

Returns:

null

Class:

EDMserver.cmd.PartitionTableCmdExecutor

pow

Raises a number to a power

Syntax:

pow('number', power)

Parameters:

number - Number to raise to a power

power - Power to raise the number to

Returns:

Number raised to a power

Class:

EDMcommon.function.CommonFunctions

processForcedTransactions

Receive and apply pending transactions whose effective date has arrived and whose Force flag is true for all sites.

Syntax:

processForcedTransactions()

Parameters:

None.

Returns:

Void.

Class:

EDMserver.function.ProcessForcedTransactionsServerFunction

processTransactions

Receive and apply pending transactions from selected sites whose effective date has arrived

Syntax:

processTransactions('siteIdList')

Parameters:

siteIdList - Comma-separated list of site id numbers (* means all sites).

Returns:

Void.

Class:

EDMserver.function.ProcessTransactionsServerFunction

publishMasterDataTableList

Publish data from the central database, with open and future transactions rolled back, to .csv files in the given directory.

Syntax:

publishMasterDataTableList('tableName', 'siteIdList', 'directory')

Parameters:

tableName - Name of table list from snapshot file. (blank means prompt user to select).

siteIdList - Comma-separated list of site IDs whose data is to be published (blank means prompt user to select).

directory - Directory where published data files are to be stored.

Returns:

Void.

Class:

EDMserver.plugin.mdm.cmd.PublishMasterDataCmdExecutor

publishMasterDataTables

Publish data from the central database, with open and future transactions rolled back, to .csv files in the given directory.

Syntax:

publishMasterDataTables('tableNameList', 'siteIdList', 'directory')

Parameters:

tableNameList - Comma-separated list of table names to publish (blank means prompt user to select, '*' means all).

siteIdList - Comma-separated list of site IDs whose data is to be published (blank means prompt user to select).

directory - Directory where published data files are to be stored.

Returns:

Void.

Class:

EDMserver.plugin.mdm.cmd.PublishMasterDataCmdExecutor

publishXCEDDataTableList

Publish data from the central database, with open and future transactions rolled back, to .csv files in the given directory.

Syntax:

```
publishXCEDDataTableList( 'tableName', 'siteIdList', 'directory')
```

Parameters:

tableName - Name of table list from snapshot file.

siteIdList - Comma-separated list of site IDs whose data is to be published.

directory - Directory where published data files are to be stored.

Returns:

Void.

Class:

EDMserver.plugin.mdm.cmd.PublishXCEDDataCmdExecutor

publishXCEDDataTables

Publish data from the central database, with open and future transactions rolled back, to .csv files in the given directory.

Syntax:

```
publishXCEDDataTables( 'tableNameList', 'siteIdList', 'directory')
```

Parameters:

tableNameList - Comma-separated list of table names to publish ("*" means all).

siteIdList - Comma-separated list of site IDs whose data is to be published.

directory - Directory where published data files are to be stored.

Returns:

Void.

Class:

EDMserver.plugin.mdm.cmd.PublishXCEDDataCmdExecutor

purgeAuditTrail

Delete audit trail rows that are older than the given number of days.

Syntax:

```
purgeAuditTrail( daysToKeep)
```

Parameters:

daysToKeep - Number of days of history to keep.

Returns:

Void.

Class:

EDMserver.ServerFunctions

purgePollData

Delete polled table rows that are older than the given number of days.

Syntax:

purgePollData(daysToKeep)

Parameters:

daysToKeep - Number of days of history to keep.

Returns:

Void.

Class:

EDMserver.ServerFunctions

queryAvg

Gets the average of the values of a column in a table or query using a where clause.

Syntax:

queryAvg('column', 'table', 'where')

Parameters:

column - Name of the column to get.

table - Name of the table containing the column.

where - Where clause to filter the rows in the table.

Returns:

Value of field if successful, otherwise empty string.

Class:

EDMserver.cmd.QueryValueCmdExecutor

queryCount

Gets the count of the values of a column in a table or query using a where clause.

Syntax:

queryCount('column', 'table', 'where')

Parameters:

column - Name of the column to get.

table - Name of the table containing the column.

where - Where clause to filter the rows in the table.

Returns:

Value of field if successful, otherwise empty string.

Class:

EDMserver.cmd.QueryValueCmdExecutor

queryFirst

Gets the first value of a column in a table or query using a where clause.

Syntax:

queryFirst('column', 'table', 'where')

Parameters:

column - Name of the column to get.

table - Name of the table containing the column.

where - Where clause to filter the rows in the table.

Returns:

Value of field if successful, otherwise empty string.

Class:

EDMserver.cmd.QueryValueCmdExecutor

queryLast

Gets the last value of a column in a table or query using a where clause.

Syntax:

queryLast('column', 'table', 'where')

Parameters:

column - Name of the column to get.

table - Name of the table containing the column.

where - Where clause to filter the rows in the table.

Returns:

Value of field if successful, otherwise empty string.

Class:

EDMserver.cmd.QueryValueCmdExecutor

queryMax

Gets the maximum value of a column in a table or query using a where clause.

Syntax:

queryMax('column', 'table', 'where')

Parameters:

column - Name of the column to get.

table - Name of the table containing the column.

where - Where clause to filter the rows in the table.

Returns:

Value of field if successful, otherwise empty string.

Class:

EDMserver.cmd.QueryValueCmdExecutor

queryMin

Gets the minimum value of a column in a table or query using a where clause.

Syntax:

```
queryMin( 'column', 'table', 'where')
```

Parameters:

column - Name of the column to get.

table - Name of the table containing the column.

where - Where clause to filter the rows in the table.

Returns:

Value of field if successful, otherwise empty string.

Class:

EDMserver.cmd.QueryValueCmdExecutor

querySum

Gets the sum of the values of a column in a table or query using a where clause.

Syntax:

```
querySum( 'column', 'table', 'where')
```

Parameters:

column - Name of the column to get.

table - Name of the table containing the column.

where - Where clause to filter the rows in the table.

Returns:

Value of field if successful, otherwise empty string.

Class:

EDMserver.cmd.QueryValueCmdExecutor

random

Returns a random number between 0 and the passed number

Syntax:

```
random( max)
```

Parameters:

max - Maximum random number that can be returned

Returns:

Random number

Class:

EDMcommon.function.CommonFunctions

receiveSiteFilesViaWeb

Receive transaction files over HTTP connection.

Syntax:

receiveSiteFilesViaWeb('siteIdList')

Parameters:

siteIdList - Comma-separated list of Location id's to transfer, * means all sites).

Returns:

Void.

Class:

EDMserver.cmd.TransferFilesCmdExecutor

receiveTransactionFilesFromAmazon

Receive transaction files from Amazon Simple Storage Service (S3).

Syntax:

receiveTransactionFilesFromAmazon('siteIdList', 'bucketName', 'prefix', 'encryptedCredentials')

Parameters:

siteIdList - Comma-separated list of site IDs to receive (blank to prompt user, * for all locations).

bucketName - Name of Amazon S3 bucket containing incoming and outgoing folders.

prefix - Common prefix of all transaction file names to be transferred without the site ID path segment, e.g. "outgoing/".

encryptedCredentials - AWS credentials encrypted with the EncryptAmazonCredentialsCmd object.

Returns:

Void

Class:

EDMserver.cmd.ReceiveTransactionFilesFromAmazonCmdExecutor

receiveTransactionFilesFromJMS

Receive files from the configured JMS queue into the local incoming folder.

Syntax:

receiveTransactionFilesFromJMS('siteIdList', 'queueName', 'brokerURL', 'userId', 'password')

Parameters:

siteIdList - Comma-separated list of site IDs to receive, '*' for all locations.

queueName - Name of the queue from which incoming transaction files should be received. For example: EDM.incoming.%REMOTESITEID%

brokerURL - URL used to connect to the JMS broker. For example: tcp://localhost:61616

userid - User ID used to connect to the JMS broker.
password - Password used to connect to the JMS broker.

Returns:

Void

Class:

EDMserver.cmd.ReceiveTransactionFilesFromJMScmdExecutor

reloadCachedData

Reload all data cached in server memory on the server and re-initialize system.

Syntax:

reloadCachedData()

Parameters:

None.

Returns:

Void.

Class:

EDMserver.cmd.ReloadCachedDataCmdExecutor

removeAmazonCommitListener

Remove listener that sends outgoing transaction files for selected sites to Amazon as soon as they are committed.

Syntax:

removeAmazonCommitListener('siteIdList', 'bucketName', 'prefix', 'encryptedCredentials')

Parameters:

siteIdList - Comma-separated list of site id numbers (* means all sites).
bucketName - Name of Amazon S3 bucket containing incoming and outgoing directories.
prefix - Common prefix of all transaction file names to be transferred without the site ID path segment, e.g. 'outgoing/'.
encryptedCredentials - AWS credentials encrypted with the EncryptAmazonCredentialsCmd object.

Returns:

Boolean - True if listener was removed.

Class:

EDMserver.function.AddAmazonCommitListenerFunction

removeTransactionsProcessedListener

Remove listener that evaluates the given expression whenever transaction processing completes.

Syntax:

removeTransactionsProcessedListener('expression')

Parameters:

expression - Expression to evaluate after transaction processing completes.

Returns:

Void.

Class:

EDMserver.function.AddTransactionsProcessedListenerFunction

replaceAll

Replaces each substring of this string that matches the given regular expression with the given replacement.

Note that backslashes (\) and dollar signs (\$) in the replacement string may cause the results to be different than if it were being treated as a literal replacement string.

Syntax:

replaceAll('string', 'regularExpression', 'replacement')

Parameters:

string - String to which replacements are made.

regularExpression - Regular expression to which this string is to be matched.

replacement - String to be substituted for each match

Returns:

String with each match replaced with the replacement string or null if the given string is null.

Class:

EDMcommon.function.CommonFunctions

requestFile

Copy a file to other locations

Syntax:

requestFile('locationList', 'fromFile', 'toFile', overwrite, synchronize, 'effectiveDate', 'packageName', 'packageDescription', forceProcessing)

Parameters:

locationList - Comma-separated list of location numbers from which to request file (blank means prompt user, * means all locations).

fromFile - Full path of file to copy (if blank, prompt user to select)

toFile - Full path of destination file at other locations (if blank, prompt user to select)

overwrite - True if file should be overwritten if it exists at destination location.

synchronize - True if file should be kept synchronized after being sent.

effectiveDate - Date file should be copied to destination

packageName - Name of package to hold transactions

packageDescription - Description of package if new package.

forceProcessing - True if transaction should be processed when only processing forced transactions.

Returns:

Void.

Class:

EDMserver.cmd.WriteFileTransactionCmdExecutor

requestTableListRefresh

Send request to location to send table data to replace existing data

Syntax:

requestTableListRefresh('tableList', 'siteIdList', forceProcessing, 'packageName', 'packageDescription', 'rollBackType', 'rollBackDate', 'rollBackTime')

Parameters:

tableList - Comma-delimited list of table names.

siteIdList - Comma-delimited list of site IDs whose table data is being requested.

forceProcessing - True if transactions should be processed when only processing forced transactions.

packageName - Name of package to hold transactions (blank means prompt user to select).

packageDescription - Description of package if a new package is to be created

rollBackType - Type of transaction roll back: 'NONE', 'NOW', or 'SPECIFIED'.

rollBackDate - If rollbackType is 'SPECIFIED', date to which transactions should be rolled back, format is mm/dd/yyyy.

rollBackTime - If rollbackType is 'SPECIFIED', time to which transactions should be rolled back, format is hh:mm:ss.SSS AM.

Returns:

Void.

Class:

EDMserver.cmd.TableListRefreshCmdExecutor

requestTableRefresh

Send request to location to send table data to replace existing data

Syntax:

requestTableRefresh('snapshotName', 'selectedLocations', forceProcessing, 'packageName', 'packageDescription')

Parameters:

snapshotName - Name of the snapshot containing the list of tables to refresh (blank means prompt user to select).

selectedLocations - List of comma-separated location ID's of the locations from which to get the test records (blank means prompt user to select).

forceProcessing - True if transactions should be processed when only processing forced transactions.

packageName - Name of package to hold transactions (blank means prompt user to select).

packageDescription - Description of package if a new package is to be created

Returns:

Void.

Class:

EDMserver.cmd.TableRefreshCmdExecutor

resetSites

Send transactions to reset remote sites to their newly installed state and commit the transactions.

Syntax:

resetSites('siteList')

Parameters:

siteList - Comma-separated list of site IDs to reset.

Returns:

void

Class:

EDMserver.cmd.ResetSitesCmdExecutor

restoreTestCheckpoint

Disconnect all database connections, restore checkpoint 1, and re-initialize server.

Syntax:

restoreTestCheckpoint('checkPoint', 'batchFile')

Parameters:

checkPoint - Checkpoint to restore.

batchFile - Batch file that restores test checkpoint.

Returns:

Void

Class:

EDMserver.cmd.RestoreTestCheckpointCmdExecutor

right

Returns the rightmost characters of a string

Syntax:

right('string', Count)

Parameters:

string - String to get the characters from

Count - Number of characters to get

Returns:

Requested characters

Class:

EDMcommon.function.CommonFunctions

round

Rounds a number at the specified number of digits past the decimal

Syntax:

round('number', digits)

Parameters:

number - Number to round

digits - Number of digits past the decimal to keep

Returns:

Rounded number

Class:

EDMcommon.function.CommonFunctions

runApp

Execute the given command line. For example, to open the config.xml file in notepad execute the command `c:\windows\system32\notepad.exe c:\EDMweb\config.xml`, and the wait flag to `false`. To run a batch file you could execute the command `cmd /c c:\EDMweb\go.bat` and the wait flag to `false`.

Syntax:

```
runApp( 'commandLine', wait)
```

Parameters:

commandLine - Command line to execute.

wait - Wait for application to complete before returning.

Returns:

0 if successful, otherwise error code.

Class:

EDMserver.cmd.RunAppCmdExecutor

runJUnitTest

Run the JUnit tests in the given class.

Syntax:

```
runJUnitTest( 'jUnitClassName')
```

Parameters:

jUnitClassName - JUnit test class to run.

Returns:

Void.

Class:

EDMserver.cmd.RunJUnitTestCmdExecutor

runReport

Run a report.

Syntax:

```
runReport( 'outputFileName', 'dataSourceName', 'dataSetClass', 'jrxmlFileName', parameters, 'outputType')
```

Parameters:

outputFileName - The name of the output report file..

dataSourceName - The data source name to use when gathering data for the report.

dataSetClass - Class that implements the ReportDataSetGenerator to collect data for a report. Null means use the data source instead..

jrxmlFileName - The name of the Jasper jrxml file.

parameters - Map of parameter names and values.

outputType - Report output type 'PDF', 'HTML', or 'XML'.

Returns:

String containing the fully qualified report file name.

Class:

EDMserver.cmd.RunReportCmdExecutor

runSQL

Executes an insert, update, delete, or maketable SQL statement.

Syntax:

runSQL('sql')

Parameters:

sql - Insert, update, delete, or maketable SQL statement to execute.

Returns:

Void.

Class:

EDMserver.ServerFunctions

sendDeleteFileTransaction

Send transaction to delete a file.

Syntax:

sendDeleteFileTransaction('file', 'siteList', 'effectiveDate', 'packageName', 'packageDescription')

Parameters:

file - File name to delete (blank means prompt user to select).

siteList - Comma-separated list of site numbers (blank means prompt user, * means all sites).

effectiveDate - Date/time transaction should be applied.

packageName - Name of package to hold transactions.

packageDescription - Description of package if new package.

Returns:

Void.

Class:

EDMserver.cmd.SendDeleteFileTransactionCmdExecutor

sendEvaluateTransaction

Send a transaction to cause the receiver to evaluate an expression.

Syntax:

sendEvaluateTransaction('expression', 'siteIdList', 'effectiveDate', 'packageName', 'packageDescription', forceProcessing)

Parameters:

expression - Expression to evaluate on receiving server (uses only server functions).

siteIdList - Comma-separated list of site numbers (blank means prompt user, * means all sites).

effectiveDate - Date/time transaction should be applied in local format or blank to be effective immediately.
packageName - Name of package to hold transactions.
packageDescription - Description of package if new package.
forceProcessing - True if transactions should be processed when only processing forced transactions.

Returns:

CommitPromptInfo

Class:

EDMserver.cmd.SendEvaluateTransactionCmdExecutor

sendExecuteTransaction

Send transaction to execute an operation system command at selected sites.

Syntax:

sendExecuteTransaction('file', 'arguments', wait, 'siteList', 'effectiveDate', 'packageName', 'packageDescription', forceProcessing)

Parameters:

file - Full path of executable file (if blank, prompt user to select)
arguments - Command line arguments.
wait - True if call should wait for process to complete before returning.
siteList - Comma-separated list of site numbers (blank means prompt user, * means all sites).
effectiveDate - Date/time transaction should be applied.
packageName - Name of package to hold transactions.
packageDescription - Description of package if new package.
forceProcessing - True if transactions should be processed when only processing forced transactions.

Returns:

Void.

Class:

EDMserver.cmd.SendExecuteTransactionCmdExecutor

sendFile

Send a file to other locations

Syntax:

sendFile('locationList', 'fromFile', 'toFile', overwrite, synchronize, 'effectiveDate', 'packageName', 'packageDescription', forceProcessing)

Parameters:

locationList - Comma-separated list of location numbers from which to request file (blank means prompt user, * means all locations).
fromFile - Full path of file to copy (if blank, prompt user to select)
toFile - Full path of destination file at other locations (if blank, prompt user to select)
overwrite - True if file should be overwritten if it exists at destination location.
synchronize - True if file should be kept synchronized after being sent.
effectiveDate - Date file should be copied to destination
packageName - Name of package to hold transactions
packageDescription - Description of package if new package.
forceProcessing - True if transaction should be processed when only processing forced transactions.

Returns:

Void.

Class:

sendRemoteFiles

Send files listed in remote files table to sites.

Syntax:

```
sendRemoteFiles( sendUpdatedOnly, 'siteIdList', 'effectiveDate', 'packageName', 'packageDescription', forceProcessing, commit)
```

Parameters:

sendUpdatedOnly - True if only files that have been updated should be sent.
siteIdList - Comma-separated list of site numbers to which files should be sent (* means all, blank means prompt user to select).
effectiveDate - Date file should be copied to destination
packageName - Name of package to hold transactions (blank means prompt user to select).
packageDescription - Description of package if a new package is to be created
forceProcessing - True if transactions should be processed when only processing forced transactions.
commit - True if package containing file transactions should be committed.

Returns:

Void.

Class:

EDMserver.ServerFunctions

sendSiteFilesViaWeb

Send transaction files over HTTP connection.

Syntax:

```
sendSiteFilesViaWeb( 'siteIdList')
```

Parameters:

siteIdList - Comma-separated list of Location id's to transfer, * means all sites).

Returns:

Void.

Class:

EDMserver.cmd.TransferFilesCmdExecutor

sendSynchronizeDirectoryTransaction

Send a transaction to synchronize a local directory with a directory at other sites.

Syntax:

```
sendSynchronizeDirectoryTransaction( 'siteIdList', 'sourceDirectory', 'destinationDirectory', sourceDirectoryIsOnSender, deleteMissingFiles, includeSubdirectories, 'effectiveDate', forceProcessing, 'packageName', 'packageDescription')
```

Parameters:

siteIdList - Comma-separated list of site IDs to which transaction will be sent (blank means prompt user, * means all sites).
sourceDirectory - Directory containing up-to-date files.
destinationDirectory - Directory that should be updated to match source directory.
sourceDirectoryIsOnSender - True if the source directory is on the local network of the system sending the transaction.
deleteMissingFiles - True if files in the destination directory should be deleted if they do not exist in the source directory.

includeSubdirectories - True if subdirectories should also be synchronized.
effectiveDate - Effective date of transaction.
forceProcessing - True if processed when only processing forced transactions.
packageName - Name of package to hold transactions.
packageDescription - Description of package if new package.

Returns:

Void.

Class:

EDMserver.cmd.SendSyncDirectoryTranCmdExecutor

sendTableListRefresh

Send table data to locations to replace existing data

Syntax:

sendTableListRefresh('tableList', 'siteIdList', destinationSiteId, forceProcessing, 'packageName', 'packageDescription', 'rollBackType', 'rollBackDate', 'rollBackTime', autoCommit)

Parameters:

tableList - Comma-delimited list of table names.
siteIdList - Comma-delimited list of site IDs whose table data is being requested.
destinationSiteId - Null if sending site data to same site, otherwise the site id of the test/lab site where the real site's data is to be sent.
forceProcessing - True if transactions should be processed when only processing forced transactions.
packageName - Name of package to hold transactions (blank means prompt user to select).
packageDescription - Description of package if a new package is to be created
rollBackType - Type of transaction roll back: 'NONE', 'NOW', or 'SPECIFIED'. Prompt user if blank.
rollBackDate - If rollBackType is 'SPECIFIED', date to which transactions should be rolled back, format is mm/dd/yyyy.
rollBackTime - If rollBackType is 'SPECIFIED', time to which transactions should be rolled back, format is hh:mm:ss.SSS AM.
autoCommit - True if refresh transaction should be committed as soon as it is generated.

Returns:

Void.

Class:

EDMserver.cmd.TableListRefreshCmdExecutor

sendTableRefresh

Send table data to locations to replace existing data

Syntax:

sendTableRefresh('snapshotName', 'selectedLocations', forceProcessing, 'packageName', 'packageDescription')

Parameters:

snapshotName - Name of the snapshot containing the list of tables to refresh (blank means prompt user to select).
selectedLocations - List of comma-separated location ID's of the locations from which to get the test records (blank means prompt user to select).
forceProcessing - True if transactions should be processed when only processing forced transactions.
packageName - Name of package to hold transactions (blank means prompt user to select).
packageDescription - Description of package if a new package is to be created

Returns:

Void.

Class:

EDMserver.cmd.TableRefreshCmdExecutor

sendTransactionFilesToAmazon

Send transaction files to Amazon Simple Storage Service (S3).

Syntax:

```
sendTransactionFilesToAmazon( 'siteIdList', 'bucketName', 'prefix', 'encryptedCredentials')
```

Parameters:

siteIdList - Comma-separated list of site IDs to send (* means all sites).

bucketName - Name of Amazon S3 bucket containing incoming and outgoing directories.

prefix - Common prefix of all transaction file names to be transferred without the site ID path segment, e.g. 'outgoing/'.

encryptedCredentials - AWS credentials encrypted with the EncryptAmazonCredentialsCmd object.

Returns:

Void.

Class:

EDMserver.cmd.SendTransactionFilesToAmazonCmdExecutor

sendTransactionFilesToJMS

Send outgoing files for the given sites to a JMS queue.

Syntax:

```
sendTransactionFilesToJMS( 'siteIdList', 'queueName', 'brokerURL', 'userId', 'password')
```

Parameters:

siteIdList - Comma-separated list of site IDs to send (* for all locations).

queueName - Name of the queue where outgoing transaction files should be sent. For example: EDM.outgoing.%REMOTESITEID%

brokerURL - URL used to connect to the JMS broker. For example: tcp://localhost:61616

userId - User ID used to connect to the JMS broker.

password - Password used to connect to the JMS broker.

Returns:

Void.

Class:

EDMserver.cmd.SendTransactionFilesToJMSCmdExecutor

sendTransactionFilesToJMSWithRetry

Send outgoing files for the given sites to a JMS queue. If the transfer fails due to a JMS error, the listener will retry after 15 seconds, 30 seconds, then 1 minute, 2 minutes, 4 minutes, 8 minutes, then every 15 minutes until the transfer works.

Syntax:

```
sendTransactionFilesToJMSWithRetry( 'siteIdList', 'queueName', 'brokerURL', 'userId', 'password', retrySeconds)
```

Parameters:

siteIdList - Comma-separated list of site IDs to send (* for all locations).

queueName - Name of the queue where outgoing transaction files should be sent. For example: EDM.outgoing.%REMOTESITEID%

brokerURL - URL used to connect to the JMS broker. For example: tcp://localhost:61616

userId - User ID used to connect to the JMS broker.

password - Password used to connect to the JMS broker.

retrySeconds - Number of seconds we waited before previous retry if the call fails. 0 means this is the first retry.

Returns:

Void.

Class:

EDMserver.cmd.SendTransactionFilesToJMScmdExecutor

setConfigProperty

Set configuration property value.

Set value of the property in the configuration file with the given case-sensitive name.

Syntax:

setConfigProperty('propertyName', 'value')

Parameters:

propertyName - Name of the user property.

value - New property value.

Returns:

Old property value.

Class:

EDMserver.cmd.SetConfigPropertyCmdExecutor

setLastReceivedTranFileNumberToIncoming

Set the last received transaction file number for a site to the lowest transaction file number in the incoming directory - 1.

Syntax:

setLastReceivedTranFileNumberToIncoming(siteId)

Parameters:

siteId - ID of the site whose last received transaction file number is to be set.

Returns:

Void.

Class:

EDMserver.cmd.SetLastReceivedTranFileNumberToIncomingCmdExecutor

setPropertyFileValue

Set the value of a property in a property file.

Syntax:

setPropertyFileValue('propertyName', 'propertyValue', 'propertyFileName', createFile, 'comments')

Parameters:

propertyName - Name of the property whose value is to be returned.
propertyValue - Property value.
propertyFileName - Path and name of property file.
createFile - True if property file should be created if it does not exist.
comments - Comments to add at the beginning of the property file or null for none.

Returns:

Void.

Class:

EDMserver.cmd.SetPropertyFileValueCmdExecutor

space

Creates a string of blanks

Syntax:

space(count)

Parameters:

count - Number of blanks to put in the string

Returns:

String of blanks

Class:

EDMcommon.function.CommonFunctions

stringToDate

Converts a date string to a date value

Syntax:

stringToDate('date', 'format')

Parameters:

date - Date string to convert
format - Format string to use

Returns:

Date value

Class:

EDMcommon.function.CommonFunctions

stringToInt

Converts a string to an integer.

Syntax:

stringToInt('string')

Parameters:

string - String to convert

Returns:

Integer value.

Class:

EDMcommon.function.CommonFunctions

stringToNumber

Converts a string to a number

Syntax:

stringToNumber('number')

Parameters:

number - Number string to convert

Returns:

Number value

Class:

EDMcommon.function.CommonFunctions

stringToTime

Converts a time string to a time value

Syntax:

stringToTime('time', 'format')

Parameters:

time - Time string to convert

format - Format string to use

Returns:

Time value

Class:

EDMcommon.function.CommonFunctions

stringToTimestamp

Converts a timestamp string to a timestamp value

Syntax:

stringToTimestamp('timestamp', 'format')

Parameters:

timestamp - Timestamp string to convert

format - Format string to use

Returns:

Date value

Class:

EDMcommon.function.CommonFunctions

substring

Returns characters from the middle of a string

Syntax:

substring('astring', pos, count)

Parameters:

astring - String to get the characters from

pos - Index of the first character to get

count - Number of characters to get

Returns:

Requested characters

Class:

EDMcommon.function.CommonFunctions

synchronizeSitePropertyValues

Import and sync the site property values from an outside data source

Syntax:

synchronizeSitePropertyValues('DSN', 'SQL', 'tableName', 'LocationIdColName', 'LocationNameColName')

Parameters:

DSN - The string of the connection in the config.xml to be used to run the SQL. Required.

SQL - The SQL string to be executed to get the source data.

tableName - The name of the table the SQL is executed against.

LocationIdColName - The column name of the location id field. Blank defaults to "Location".

LocationNameColName - The column name of the location name field. Blank defaults new stores to "Store <id>" for a name and no updates to the existing location names will be performed.

Returns:

Void.

Class:

EDMserver.ServerFunctions

time

Gets current time of day

Syntax:

time('format')

Parameters:

format - Format string describing how the date should be formatted.

Returns:

Current time formatted using format string

Class:

EDMcommon.function.CommonFunctions

timestamp

Get current date and time.

Syntax:

timestamp('format')

Parameters:

format - Format string describing how the date and time should be formatted.

Returns:

String representation of the current date and time.

Class:

EDMcommon.function.CommonFunctions

timestampToDate

Converts a timestamp to a date

Syntax:

timestampToDate('timestamp')

Parameters:

timestamp - Timestamp to convert

Returns:

Date part of the timestamp

Class:

EDMcommon.function.CommonFunctions

timestampToNumber

Converts a timestamp to a number

Syntax:

timestampToNumber('timestamp')

Parameters:

timestamp - Timestamp to convert

Returns:

Number representing the timestamp

Class:

EDMcommon.function.CommonFunctions

timestampToString

Converts a timestamp to a character string

Syntax:

timestampToString('timestamp', 'format')

Parameters:

timestamp - Date to convert

format - Format string to use

Returns:

String representation of the timestamp

Class:

EDMcommon.function.CommonFunctions

timestampToTime

Converts a timestamp to a time

Syntax:

timestampToTime('timestamp')

Parameters:

timestamp - Timestamp to convert

Returns:

Time part of the timestamp

Class:

EDMcommon.function.CommonFunctions

timeToNumber

Converts a time to a number

Syntax:

timeToNumber('time')

Parameters:

time - Time to convert

Returns:

Number representing the time

Class:

EDMcommon.function.CommonFunctions

timeToString

Converts a time to a character string

Syntax:

timeToString('time', 'format')

Parameters:

time - Time to convert

format - Format string to use

Returns:

String representation of the time

Class:

EDMcommon.function.CommonFunctions

timeToTimestamp

Converts a time to a timestamp

Syntax:

timeToTimestamp('time')

Parameters:

time - Time to convert

Returns:

Timestamp with date part set to 01/01/01

Class:

EDMcommon.function.CommonFunctions

transferSiteFilesViaFTP

Transfer transaction files over FTP connection.

Syntax:

transferSiteFilesViaFTP('siteIdList')

Parameters:

siteIdList - Comma-separated list of Location id's to transfer, * means all sites).

Returns:

Void.

Class:

EDMserver.cmd.TransferFilesCmdExecutor

transferSiteFilesViaRAS

Transfer transaction files over network connection.

Syntax:

transferSiteFilesViaRAS('siteIdList')

Parameters:

siteIdList - Comma-separated list of Location id's to transfer, * means all sites).

Returns:

Void.

Class:

EDMserver.cmd.TransferFilesCmdExecutor

transferSiteFilesViaWeb

Transfer transaction files over HTTP connection.

Syntax:

transferSiteFilesViaWeb('siteIdList')

Parameters:

siteIdList - Comma-separated list of Location id's to transfer, * means all sites).

Returns:

Void.

Class:

EDMserver.cmd.TransferFilesCmdExecutor

trim

Trims the blanks from both ends of a string

Syntax:

trim('string')

Parameters:

string - String to trim

Returns:

Trimmed string

Class:

EDMcommon.function.CommonFunctions

trimLeft

Trims the blanks from the left end of a string

Syntax:

```
trimLeft( 'string')
```

Parameters:

string - String to trim

Returns:

Trimmed string

Class:

EDMcommon.function.CommonFunctions

trimRight

Trims the blanks from the right end of a string

Syntax:

```
trimRight( 'string')
```

Parameters:

string - String to trim

Returns:

Trimmed string

Class:

EDMcommon.function.CommonFunctions

updateAmazonDirToMatchLocalDir

Update an Amazon S3 directory to match a local directory.

Syntax:

```
updateAmazonDirToMatchLocalDir( 'serverDirectory', 'localDirectory', deleteMissingFiles, 'bucketName', 'encryptedCredentials')
```

Parameters:

serverDirectory - Directory on the server to update to match the files in the local directory.

localDirectory - Local directory to check for new, updated, and deleted files.

deleteMissingFiles - True if files in the S3 directory should be deleted if they do not exist in the local directory.

bucketName - Name of Amazon S3 bucket containing server directory.

encryptedCredentials - AWS credentials encrypted with the EncryptAmazonCredentialsCmd object.

Returns:

Void

Class:

EDMserver.cmd.UpdateAmazonDirToMatchLocalDirCmdExecutor

updateLocalDirToMatchAmazonDir

Update a local directory to match a directory on Amazon S3.

Syntax:

```
updateLocalDirToMatchAmazonDir( 'serverDirectory', 'localDirectory', deleteMissingFiles, 'bucketName', 'encryptedCredentials')
```

Parameters:

serverDirectory - Server directory containing files and sub-directories.

localDirectory - Local directory to synchronize.

deleteMissingFiles - True if files in the local directory should be deleted if they do not exist on the server.

bucketName - Name of Amazon S3 bucket containing server directory.

encryptedCredentials - AWS credentials encrypted with the EncryptAmazonCredentialsCmd object.

Returns:

Void

Class:

EDMserver.cmd.UpdateLocalDirToMatchAmazonDirCmdExecutor

updateLocalDirToMatchWebDir

Update a local directory to match a directory on the EDM web server.

Syntax:

```
updateLocalDirToMatchWebDir( 'serverDirectory', 'localDirectory', deleteMissingFiles)
```

Parameters:

serverDirectory - Server directory containing files and sub-directories.

localDirectory - Local directory to synchronize.

deleteMissingFiles - True if files in the local directory should be deleted if they do not exist on the server.

Returns:

Void

Class:

EDMserver.cmd.UpdateLocalDirToMatchWebDirCmdExecutor

updateTableList

Update a table list.

Syntax:

```
updateTableList( 'oldTableName', tableList, index)
```

Parameters:

oldTableName - New table list to insert.

tableList - New table list.

index - 0-based index where table list should be inserted after deleting original table list, if index is < 0 or >= list size, the new table list is inserted at the same index as the original table list.

Returns:

Void.

Class:

EDMserver.cmd.EditSnapshotTablesCmdExecutor

updateTransferSettings

Save the given transfer host, user ID, password, and company name used to connect to a central EDM server

<p>The updateTransferSettings function updates the transfer host, user ID, password, and company name on the central site record. This information is used when connecting to the central system to transfer transaction files. To have the information automatically saved when the system starts up, the function can be called in the onStartupProperty of config.xml as shown in the following example.</p><pre>onStartup = "updateTransferSettings('localhost:8080/EDM', 'remote', 'remote', 'HQ')"</pre>

Syntax:

updateTransferSettings('hostName', 'userId', 'password', 'company')

Parameters:

hostName - Host name for file transfers.
userId - User id for transaction file transfers.
password - Password for transaction file transfers.
company - Company name for transaction file transfers.

Returns:

Void.

Class:

EDMserver.cmd.UpdateTransferSettingsCmdExecutor

updateWebDirToMatchLocalDir

Update an directory on the EDM web server to match a local directory.

Syntax:

updateWebDirToMatchLocalDir('serverDirectory', 'localDirectory', deleteMissingFiles)

Parameters:

serverDirectory - Directory on the server to update to match the files in the local directory.
localDirectory - Local directory to check for new, updated, and deleted files.
deleteMissingFiles - True if files in the server directory should be deleted if they do not exist in the local directory.

Returns:

Void

Class:

EDMserver.cmd.UpdateWebDirToMatchLocalDirCmdExecutor

upper

Changes each letter of a string to upper case

Syntax:

upper('string')

Parameters:

string - String to convert

Returns:

String converted to upper case

Class:

EDMcommon.function.CommonFunctions

validateTableList

Validate references between tables

Syntax:

validateTableList('tableListName', 'siteIdList')

Parameters:

tableListName - Name of the table list containing the list of tables to validate.

siteIdList - List of comma-separated location ID's of the locations from which to get the test records (* means all locations).

Returns:

Server file reference.

Class:

EDMserver.cmd.ValidateTableListCmdExecutor

validateTables

Validate references between tables

Syntax:

validateTables('tableNames', 'siteIdList')

Parameters:

tableNames - Comma-separated list of tables to validate (* means all tables).

siteIdList - Comma-separated list of site IDs to validate (* means all sites).

Returns:

Server file reference.

Class:

EDMserver.cmd.ValidateTablesCmdExecutor

year

Returns the year value of a date

Syntax:

year('date')

Parameters:

date - Date to get the year from

Returns:

Four digit year number

Class:

EDMcommon.function.CommonFunctions

Format Strings

Format strings are used to:

- Control how data is displayed on spreadsheets and forms
- Define fields in a table
- Define controls on forms

Examples

To display the telephone number 8005551212 as (800) 555-1212, use the format string (###) ###-####

To format the date 6/30/96 as Friday, June 30, 1996, use the format string Dddd, Mmmm d, yyyy

Topics

[Format Strings for Text](#)

[Format Strings for Numbers](#)

[Format Strings for Dates and Times](#)

Format Strings for Text

You can format field or control text to appear as asterisks (*) instead of the actual data, which is useful when the user is entering a password or sensitive account information.

Formatting a Field in a Table

Select *Password* for the *Format* attribute -OR- type an asterisk (*).

Formatting a Control on a Form

1. Select the control.
2. Type an asterisk (*) for the *Format* attribute in the attribute window.

Format Strings for Numbers

You can format field and form control numeric values with symbols including dollar signs, thousand separators, scientific notations and percent symbols.

Numeric format strings may have one or two sections separated by a semicolon (;).

If the format string has one section, the same format is used for both positive and negative values. A negative sign (-) is automatically inserted for negative values.

If the format string has two sections, the first section provides the formatting for positive numbers and the second section provides the formatting for negative numbers.

The following table includes a format string with one section and a format string with two sections.

Format string	Value	Formatted value
0.00	100.5	100.50
	-145.1	-145.10
0.00;(0.00)	100.5	100.50
	-145.1	(145.10)

Symbols Allowed in a Numeric Format String

Symbol	Description															
\$	Output the currency string. The currency string is specified by the country setting for your system.															
.	Output the decimal point character. The decimal point character is specified in the country settings for your system.															
,	Output the thousands separator character. The thousands separator character is specified in the country settings for your system.															
#	<p>Output a digit. If there is no digit to output in the position, output nothing.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Format String</th> <th>Displayed Value</th> </tr> </thead> <tbody> <tr> <td>12.3</td> <td>### ##</td> <td>12.3</td> </tr> <tr> <td>125.224</td> <td>### ##</td> <td>125.22</td> </tr> <tr> <td>0</td> <td>### ##</td> <td>.</td> </tr> <tr> <td>1500</td> <td>### ##</td> <td>1500.</td> </tr> </tbody> </table> <p>If the value has more digits to the left of the decimal than there are symbols in the format string, the format string is automatically extended to the left. However, if the value has more digits to the right of the decimal, the value is rounded to the last digit.</p>	Value	Format String	Displayed Value	12.3	### ##	12.3	125.224	### ##	125.22	0	### ##	.	1500	### ##	1500.
Value	Format String	Displayed Value														
12.3	### ##	12.3														
125.224	### ##	125.22														
0	### ##	.														
1500	### ##	1500.														
0	<p>Output a digit. If there is no digit to output in the position, output a zero.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Format String</th> <th>Displayed Value</th> </tr> </thead> <tbody> <tr> <td>12.3</td> <td>000.00</td> <td>012.30</td> </tr> <tr> <td>125.224</td> <td>000.00</td> <td>125.22</td> </tr> <tr> <td>0</td> <td>000.00</td> <td>000.00</td> </tr> <tr> <td>1500</td> <td>000.00</td> <td>1500.00</td> </tr> </tbody> </table>	Value	Format String	Displayed Value	12.3	000.00	012.30	125.224	000.00	125.22	0	000.00	000.00	1500	000.00	1500.00
Value	Format String	Displayed Value														
12.3	000.00	012.30														
125.224	000.00	125.22														
0	000.00	000.00														
1500	000.00	1500.00														
?	<p>Output a digit. If there is no digit to output in the position, output a space character.</p> <p>If the value has more digits to the left of the decimal than there are symbols in the format string, the format string is automatically extended to the left. However, if the value has more digits to the right of the decimal, the value is rounded to the last digit.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Format String</th> <th>Displayed Value</th> </tr> </thead> <tbody> <tr> <td>12.3</td> <td>??? ??</td> <td>"12.3"</td> </tr> <tr> <td>125.224</td> <td>??? ??</td> <td>"125.22"</td> </tr> <tr> <td>0</td> <td>??? ??</td> <td>". "</td> </tr> <tr> <td>1500</td> <td>??? ??</td> <td>"1500. "</td> </tr> </tbody> </table>	Value	Format String	Displayed Value	12.3	??? ??	"12.3"	125.224	??? ??	"125.22"	0	??? ??	". "	1500	??? ??	"1500. "
Value	Format String	Displayed Value														
12.3	??? ??	"12.3"														
125.224	??? ??	"125.22"														
0	??? ??	". "														
1500	??? ??	"1500. "														
%	<p>Output the value as a percentage. The value is multiplied by 100 and the percent symbol (%) is output.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Format String</th> <th>Displayed Value</th> </tr> </thead> <tbody> <tr> <td>0.15</td> <td>#0%</td> <td>15%</td> </tr> <tr> <td>0.04</td> <td>#0%</td> <td>4%</td> </tr> <tr> <td>0</td> <td>#0%</td> <td>0%</td> </tr> <tr> <td>1</td> <td>#0%</td> <td>100%</td> </tr> </tbody> </table>	Value	Format String	Displayed Value	0.15	#0%	15%	0.04	#0%	4%	0	#0%	0%	1	#0%	100%
Value	Format String	Displayed Value														
0.15	#0%	15%														
0.04	#0%	4%														
0	#0%	0%														
1	#0%	100%														

e+ e-	<p>Output using scientific notation. e+ outputs the sign of the exponent if it is negative. e- outputs the sign of the exponent if it is negative or positive.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Format String</th> <th>Displayed Value</th> </tr> </thead> <tbody> <tr> <td>12500</td> <td>0.00e+#0</td> <td>1.25e04</td> </tr> <tr> <td>0.005</td> <td>0.00e+#0</td> <td>5.0e-03</td> </tr> <tr> <td>12500</td> <td>0.00e-#0</td> <td>1.25e+04</td> </tr> <tr> <td>0.005</td> <td>0.00e-#0</td> <td>5.0e-03</td> </tr> </tbody> </table> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <p>You can also use e+ or e- in the format string. This causes the "e" to be uppercase in the formatted value.</p> </div>	Value	Format String	Displayed Value	12500	0.00e+#0	1.25e04	0.005	0.00e+#0	5.0e-03	12500	0.00e-#0	1.25e+04	0.005	0.00e-#0	5.0e-03
Value	Format String	Displayed Value														
12500	0.00e+#0	1.25e04														
0.005	0.00e+#0	5.0e-03														
12500	0.00e-#0	1.25e+04														
0.005	0.00e-#0	5.0e-03														
-()space	<p>Output plus or minus signs, parentheses or blank spaces. These characters are often used to distinguish positive and negative values.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Format String</th> <th>Displayed Value</th> </tr> </thead> <tbody> <tr> <td>12.3</td> <td>+0.00;-0.00</td> <td>+12.30</td> </tr> <tr> <td>-1.1</td> <td>+0.00;-0.00</td> <td>-1.10</td> </tr> <tr> <td>12.3</td> <td>0. 0 0</td> <td>12. 3 0</td> </tr> </tbody> </table> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <p>These are the only characters that can be included in numeric format strings to be output directly. To output other characters or strings, use the "\ " symbol or enclose the characters in quotation marks.</p> </div>	Value	Format String	Displayed Value	12.3	+0.00;-0.00	+12.30	-1.1	+0.00;-0.00	-1.10	12.3	0. 0 0	12. 3 0			
Value	Format String	Displayed Value														
12.3	+0.00;-0.00	+12.30														
-1.1	+0.00;-0.00	-1.10														
12.3	0. 0 0	12. 3 0														
\	Output the character following the backslash. For example, if the format string is "0.00 \t\o\n\s", the value 1.25 is formatted as "1.25 tons".															
"string"	Output the string. The quotation marks are not output. For example, if the format string is "0.00 "tons"", the value 1.25 is formatted as "1.25 tons".															
'string'	Output the string. The quotation marks are not output. For example, if the format string is "0.00 'tons'", the value 1.25 is formatted as "1.25 tons".															
GN	<p>General format for numbers. This is the format used if no format string is given.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Format String</th> <th>Displayed Value</th> </tr> </thead> <tbody> <tr> <td>12.3</td> <td>GN</td> <td>12.3</td> </tr> <tr> <td>125.224</td> <td>GN</td> <td>125.224</td> </tr> <tr> <td>0</td> <td>GN</td> <td>0</td> </tr> <tr> <td>-1500</td> <td>GN</td> <td>-1500</td> </tr> </tbody> </table> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <p>If you use GN, the only other symbols you can use in the format string are those enclosed in brackets; for example [US].</p> </div>	Value	Format String	Displayed Value	12.3	GN	12.3	125.224	GN	125.224	0	GN	0	-1500	GN	-1500
Value	Format String	Displayed Value														
12.3	GN	12.3														
125.224	GN	125.224														
0	GN	0														
-1500	GN	-1500														
GF	<p>General fixed format for numbers. This is the same as the GN format above except that the number of decimal digits conforms to the country settings defined for your system.</p> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <p>If you use GF, the only other symbols you can use in the format string are those enclosed in brackets; for example [US].</p> </div>															
GC	<p>General currency format for numbers. The currency format is specified by the country settings for your system.</p> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <p>If you use GC, the only other symbols you can use in the format string are those enclosed in brackets; for example [US].</p> </div>															

[S/n]	Scale the number before it is output. The number is divided by 'n' before it is formatted, where 'n' is a power of 10 (10, 100, 1000, etc). For example, if the format string is "#0.00[S/1000]", 12340 is formatted as "12.34".
[US]	The United States defaults are substituted for the country settings defined for your system. For example, if the format string is "\$#,##0.00[US]", 1234.56 is formatted as "\$1,234.56".
*	Password format. Asterisks (*) will be displayed instead of the actual data.

Format Strings for Dates and Times

You can format field or control date/time values as follows:

- Specify the elements of the date or time to output
- Specify the sequence of the elements to output
- Specify numeric or alphabetic output for the date

When editing the date or time on forms or spreadsheets, the format string controls the string that is displayed.

For example, if the format string contains date and time symbols, e.g. 'GDT', then when you edit the field the data is presented in the form 'M/d/yy h:mm:ss a' (country dependent).

If the format string contains only date symbols, e.g. 'GD', then when you edit the field you only see the date part 'M/d/yy' and the time is set to 12:00AM.

If the format string contains only time symbols, e.g. 'GT', then when you edit the field, you only see the time part 'h:mm:ss a' and the date is set to 1899-12-30.

Symbols Allowed in Date/Time Format Strings

Date and time formats are specified by *date and time pattern* strings. Unquoted letters A to Z (and a to z) are interpreted as pattern letters representing the components of a date or time string. Text can be quoted using single quotes (') to avoid interpretation. All other characters are not interpreted, but are copied to the output string during formatting or matched against the input string during parsing.

The following pattern letters are defined (all other characters from A to Z (and a to z) are reserved):

Letter	Date or Time Component	Presentation	Examples
G	Era designator	Text	AD
y	Year	Year	1996; 96
M	Month in year	Month	July; Jul; 07
w	Week in year	Number	27
W	Week in month	Number	2
D	Day in year	Number	189
d	Day in month	Number	10
F	Day of week in month	Number	2
E	Day in week	Text	Tuesday; Tue
a	AM/PM marker	Text	PM
H	Hour in day (0-23)	Number	0
k	Hour in day (1-24)	Number	24
K	Hour in AM/PM (0-11)	Number	0
h	Hour in AM/PM (1-12)	Number	12
m	Minute in hour	Number	30
s	Second in minute	Number	55
S	Millisecond	Number	978
z	Time zone	General time zone	Pacific Standard Time; PST; GMT-08:00
Z	Time zone	RFC 822 time zone	-0800

Pattern letters are usually repeated, as their number determines the exact presentation.

<p>Text</p>	<p>For formatting, if the number of pattern letters is 4 or more, then the full form is used. Otherwise, a short or abbreviated form is used if available.</p> <p>For parsing, both forms are accepted regardless of the number of pattern letters.</p>
<p>Number</p>	<p>For formatting, the number of pattern letters is the minimum number of digits. Shorter numbers are padded with zeros to meet the minimum.</p> <p>For parsing, the number of pattern letters is ignored unless it's needed to separate two adjacent fields.</p>
<p>Year</p>	<p>For formatting, if the number of pattern letters is 2, then the year is truncated to 2 digits. Otherwise, it is interpreted as a number.</p> <p>For parsing, if the number of pattern letters is more than 2, then the year is interpreted literally, regardless of the number of digits. So using the pattern "MM/dd/yyyy", "01/11/12" parses to Jan 11, 12 A.D. For parsing with the abbreviated year pattern ("y" or "yy"), <i>SimpleDateFormat</i> must interpret the abbreviated year relative to some century. It adjusts dates to be within 80 years before and 20 years after the time the <i>SimpleDateFormat</i> instance is created.</p> <p>For example, using a pattern of "MM/dd/yy" with a <i>SimpleDateFormat</i> instance created on Jan 1, 1997, the string "01/11/12" would be interpreted as Jan 11, 2012, while the string "05/04/64" would be interpreted as May 4, 1964.</p> <p>During parsing, only strings consisting of exactly two digits, as defined by <i>Character.isDigit(char)</i>, will be parsed into the default century. Any other numeric string, such as a one digit string, a three or more digit string, or a two digit string that isn't all digits (for example, "-1"), is interpreted literally. So "01/02/3" or "01/02/003" are parsed, using the same pattern, as Jan 2, 3 AD. Likewise, "01/02/-3" is parsed as Jan 2, 4 BC.</p>
<p>Month</p>	<p>If the number of pattern letters is 3 or more, the month is interpreted as text. Otherwise, it is interpreted as a number.</p>
<p>General time zone</p>	<p>Time zones are interpreted as text if they have names. For time zones representing a GMT offset value, the following syntax is used:</p> <p><i>GMTOffsetTimeZone:</i> <i>GMT Sign Hours : Minutes</i></p> <p><i>Sign:</i> one of + -</p> <p><i>Hours:</i> <i>Digit</i> <i>Digit Digit</i></p> <p><i>Minutes:</i> <i>Digit Digit</i> <i>Digit:</i> one of 0 1 2 3 4 5 6 7 8 9</p> <p><i>Hours</i> must be between 0 and 23. <i>Minutes</i> must be between 00 and 59. The format is locale independent. <i>TwoDigitHours</i> must be between 00 and 23.</p> <p>For parsing, general time zones are also accepted.</p>

Examples

The following examples show how date and time patterns are interpreted in the U.S. locale. The given date and time are 2001-07-04 12:08:56 local time in the U.S. Pacific Time time zone.

Date and Time Pattern	Result

"yyyy.MM.dd G 'at' HH:mm:ss z"	2001.07.04 AD at 12:08:56 PDT
"EEE, MMM d, ''yy"	Wed, Jul 4, '01
"h:mm a"	12:08 PM
"hh 'o'clock' a, zzzz"	12 o'clock PM, Pacific Daylight Time
"K:mm a, z"	0:08 PM, PDT
"yyyyy.MMMMM.dd GGG hh:mm aaa"	02001.July.04 AD 12:08 PM
"EEE, d MMM yyyy HH:mm:ss Z"	Wed, 4 Jul 2001 12:08:56 -0700
"yyMMddHHmmssZ"	010704120856-0700
"yyyy-MM-dd'T'HH:mm:ss.SSSZ"	2001-07-04T12:08:56.235-0700

The following format strings may be used to refer to dates, times or date/time values in the default format of the current locale.

Symbol	Description
GD	General date format. This format is used if no format string is given. The short date format defined by the country settings for your system is used.
GT	General format for time. The time format defined by the country settings for your system is used.
GDT	General format for dates with times. The time format defined by the country settings for your system is appended to the short date format.

File Transfer

The pages in this section describe how to configure the system to transfer transaction files between central and remote sites.

Topics

[Web Transfer](#)

[Amazon S3 Transfer](#)

[FTP Transfer](#)

[Batch Process for File Transfer at Central Locations](#)

[Batch Process for File Transfer at Remote Locations](#)

[Changing the Process Transactions Menu Options to also Transfer Files](#)

[Adding File Transfer to the Menu](#)

[Managing Files](#)

Web Transfer

If the central server is visible to the remote sites via the Internet or a virtual private network, then users can configure the system to transfer transaction files by communicating directly with the central server using the HTTP protocol. This file transfer method is fast and it is easy to set up and manage the connection with existing firewalls.

Normally, remote sites are not run on a Web server, but are installed as standalone programs, so the remote sites have to initiate communication with the central site where the Web application is always running. It is possible for the central site to initiate communication, but this configuration is rare due to the requirement for a Web server at each remote site.

Configuring Remote Sites

To configure a remote site to communicate with the central server:

1. Start the remote system by running *go.bat* from the installation folder (usually *C:\EDMWeb*).
2. Log Out the remote user, and Log In as a user with permission to edit the location table.
3. Select the *Sites* module.
4. Select *Edit Locations and Groups* from the *Setup* menu to open the location form.
5. Select the *Location ID* field. Press *Ctrl-F* on your keyboard to open the *Find* dialog. Type 0, and then click *OK* to find the location record for the central location (usually location 0).
6. Set the *Transfer host* field to *servername:8080/EDM*, where *servername* is the name (or IP address) of the central server and *:8080* is the port number used by the server.

The `:8080` can be omitted if the central server uses the default port 80.

7. Set the *Transfer user id* field to *remote*.
8. Set the *Transfer password* field to *remote*.
9. Set the *Transfer company* field to *HQ* (or the name you provided for the central site).

Transferring Files

The system includes a batch file named *transfer_web.bat* that transfers the transaction files, processes the transactions, and then transfers the results. Once the transfer fields have been filled in for the central location record, you can transfer and process transactions by simply running *transfer_web.bat* from the installation folder (usually *C:\EDMWeb*).

If the transfer is not successful, review the *log.txt* file in the installation folder.

Transferring Directories

The system can synchronize entire directories between the central server and the remote sites.

Synchronizing a Local Directory

The following describes how to synchronize a local directory with a directory on the server.

The `updateLocalDirToMatchWebDir` function is used to synchronize a directory of image or document files at remote sites with a directory on the server.

To have remote sites synchronize a local folder with a server folder when the contents of the server folder have changed, set up a menu item with an action like this:

```
sendEvaluateTransaction( 'updateLocalDirToMatchWebDir( "files/images", "c:\EDMweb\files\images", true )', '1', "", '0-Default Package', "", true )
```

To have remote sites synchronize a local folder with a server folder every day, call the function in *transfer_web.bat* before the *'Exit()'* function like this:

```
... updateLocalDirToMatchWebDir( 'files/images', 'c:\EDMweb\files\images', true ) + Exit()
```

See `updateLocalDirToMatchWebDir` function documentation for details.

Synchronizing a Directory on the Server

The following describes how to synchronize a directory on the server with a local directory.

The `updateWebDirToMatchLocalDir` function is used to synchronize a directory of image or document files on the server with a directory at a remote site.

To have remote sites synchronize a server folder with a local folder when the contents of the local folder have changed, set up a menu item with an action like this:

```
sendEvaluateTransaction( 'updateWebDirToMatchLocalDir( "files/images", "c:\EDMweb\files\images", true )', '1', "", '0-Default Package', "", true )
```

To have remote sites synchronize a server folder with a local folder every day, call the function in *transfer_web.bat* before the *'Exit()'* function like this:

```
... updateWebDirToMatchLocalDir( 'files/images', 'c:\EDMweb\files\images', true ) + Exit()
```

See `updateWebDirToMatchLocalDir` function documentation for details.

Amazon S3 Transfer

Using the *Amazon Simple Storage Service (S3)*, EDM can leverage the power of cloud computing to transfer transaction files or entire directories between the central site and remote sites. S3 provides users access to the same highly-scalable, reliable, secure, fast and inexpensive infrastructure that Amazon uses to run its own global network of Web sites.

Setting up your Amazon account

To use Amazon transaction file transfer, users will need their own Amazon Web Services (AWS) account with access to S3.

The `encryptAmazonCredentials` function is used to encrypt a property file containing the Amazon Web Services (AWS) access key and secret key. The encrypted credentials can be passed to the functions that send and receive transaction files from Amazon S3 so that they can log into AWS. The encrypted property file can be sent to remote sites so that they can also connect to AWS to send and receive transaction files without making the AWS access key and secret key visible to personnel at the site.

Creating an Encrypted AWS Credentials File

To create an encrypted AWS credentials file:

1. Create a file called `aws.properties` with the following lines using your valid AWS account keys:
`accessKey = myaccesskey`
`secretKey = mysecretkey`
2. Use the *Upload File* link (located at the bottom of the web page below the EDM window) to upload the file to the server. To upload the file to the `webapps\EDMFiles` directory, leave the server folder blank.
3. Add an EDM menu item to encrypt the file with the name *Encrypt Amazon Credentials* and the action `encryptAmazonCredentials('Files\aws.properties')`.
4. Select the new menu item.
5. To render the encrypted `aws.properties` file inaccessible to browser users, move the file from the Web server's `webapps\EDMFiles` directory to the `web-inf\classes` directory.
6. Copy the `aws.properties` file to the `C:\EDMWeb` folder at remote sites that will use S3.

See `encryptAmazonCredentials` function documentation for details.

Transferring Central Transaction Files to and from Amazon S3

At central sites, the administrator can configure `config.xml` to schedule a recurring task that sends outgoing transaction files to S3 and receives incoming files from S3.

In the following example, `config.xml` is configured to send and receive files every 15 minutes.

```
<schedule
  frequency = "TIMED"
  time = "1/1/2001 12:15 AM"
  task = "receiveTransactionFilesFromAmazon( '*',
    'mybucket',
    'incoming',
    getPropertyFileValue( 'credentials', 'web-inf\classes\aws.properties' )
  )
  + sendTransactionFilesToAmazon( '*',
    'mybucket',
    'outgoing',
    getPropertyFileValue( 'credentials', 'web-inf\classes\aws.properties' )
  )"
/>
```

See `sendTransactionFilesToAmazon` and `receiveTransactionFilesFromAmazon` function documentation for details.

Transferring Remote Transaction Files to and from Amazon S3

To use Amazon to transfer transaction files at remote sites:

1. Copy the encrypted AWS credentials properties file to the `C:\EDMWeb` directory.
2. Start EDM with a startup expression like the one shown below from `transfer_amazon.bat`.

```
call "C:\EDMWeb\go" "-expr=ShowWarnings(0) + ShowErrors(0) +
receiveTransactionFilesFromAmazon( '*', 'mybucket', 'outgoing',
getPropertyFileValue( 'credentials', 'aws.properties' ) ) +
ProcessTransactions('*') + sendTransactionFilesToAmazon( '*',
'mybucket', 'incoming', getPropertyFileValue( 'credentials',
'aws.properties' ) ) + Exit()" "-nomenu"
```

To use the Amazon file transfer at remote sites, the remote user will need Execute permission for the *getPropertyFileValue*, *receiveTransactionFilesFromAmazon*, and *sendTransactionFilesToAmazon* functions in the *policy.xml* file. These permissions are provided in the default policy file that is installed with EDM.

Updating an S3 Directory to Match a Directory on the Central Server

The *updateAmazonDirToMatchLocalDir* function synchronizes an Amazon S3 directory with a local directory, so that remote sites can retrieve the files in the directory with the *updateLocalDirToMatchAmazonDir* function.

For example, if the central EDM Web server has a directory with all the image files used by the POS system at remote sites, a directory on Amazon S3 can be updated to match the server directory.

To synchronize the S3 directory *Files/Images* with the local directory *C:\Images*, create an EDM menu item with an action like this:

```
updateLocalDirToMatchAmazonDir( 'Files/Images', 'C:\Images', true,
'mybucket', getPropertyFileValue( 'credentials', 'aws.properties' ) )
```

See *updateAmazonDirToMatchLocalDir* function documentation for details.

Updating a Directory at a Remote site to Match a Directory on S3

The *updateLocalDirToMatchAmazonDir* function synchronizes a local directory with a directory on Amazon S3 to easily update large numbers of files at remote sites, such as image or document files.

For example, if remote sites have a directory with all the image files used by the POS system, you can create a directory on S3 that contains the latest image files and the remote sites can compare the directories periodically (or on demand) and download any new or updated files, or delete any obsolete files.

To synchronize the local directory *C:\Images* with the S3 directory *Files/Images* everyday during end-of-day transaction processing, add the following expression to *transfer_amazon.bat*:

```
updateLocalDirToMatchAmazonDir( 'Files/Images', 'C:\Images', true,
'mybucket', getPropertyFileValue( 'credentials', 'aws.properties' ) )
```

To only synchronize the remote sites with S3 when the S3 directory is updated, use an EDM menu item after updating the S3 directory to send an *evaluate* transaction to the desired sites. When the sites process the transaction, they will call the function to synchronize the local directory with the directory on S3.

```
sendEvaluateTransaction( 'updateLocalDirToMatchAmazonDir( 'Files/Images',
'C:\Images', true, 'mybucket', getPropertyFileValue( 'credentials',
'aws.properties' ) )', '', '', '', '', true )
```

See *updateLocalDirToMatchAmazonDir* function documentation for details.

FTP Transfer

If the central and remote sites are not connected with a virtual private network, then some other mechanism must be used to transfer transaction files. The system can use the Internet's File Transfer Protocol (FTP) to transfer transaction files between locations and an FTP server. Typically, the remote sites initiate the transfer because the sites process transactions automatically as part of their End-of-Day process.

Handling FTP Transfers

There are three ways to handle the FTP transfers:

Central FTP Server	The central system stores incoming and outgoing files in the home directory of the FTP server located on the local area network, or even on the central system itself. Remote sites use FTP to transfer files to the FTP server on the central network. The FTP server must be visible to the remote sites either via the Internet or a virtual private network. This is the simplest and most efficient configuration because it eliminates the need to use FTP to transfer files from the central <i>Outgoing</i> and <i>Incoming</i> directories to the FTP server.
Internet FTP Server	Both the central and remote systems use FTP to transfer files to an FTP server on the Internet. This requires both the central and remote sites to have Internet access.
Remote FTP Servers	The central system uses FTP to transfer files to an FTP server running at each remote site. Remote systems store their incoming and outgoing files in the home directory of the FTP server located on the local area network, or even on the remote system itself. The remote FTP servers must be visible to the central system either via the Internet or a virtual private network.

The recommended way to set up the system is to run the FTP system on a computer that the central system can directly access via the network, or on the central system computer if another computer is not available.

See Also

[Setting Up an FTP Server](#)

Setting Up an FTP Server

In order to use FTP to transfer transaction files, the central and remote sites must have:

- Access to an FTP server
- The FTP host name
- The user ID and password to access the FTP server

It may be convenient to create a special user for the system that directs the user to a home directory created to hold the incoming and outgoing transaction files.

For example, the FTP server administrator could create a user called *TRANS* whose home directory is the *TRANS* directory, which would contain *Incoming* and *Outgoing* sub-directories. Both the *Incoming* and *Outgoing* directories would contain a sub-directory for each remote site, where the name of the subdirectory is the location ID of the remote site.

The following is a sample directory layout for the FTP server using *TRANS* as the home directory.

```
\TRANS
  \Incoming
    \1
    \2
  \Outgoing
    \1
    \2
```

Central FTP Server

The central system stores incoming and outgoing files in the home directory of the FTP server located on the local area network, or even on the central system itself. Remote sites use FTP to transfer files to the FTP server on the central network. The FTP server must be visible to the remote sites either via the Internet or a virtual private network.

This is the simplest and most efficient configuration because it eliminates the need to use FTP to transfer files from the central *Outgoing* and *Incoming* directories to the FTP server.

System Requirements

Central system	Must map a drive letter to the home directory of the FTP server, which is the default directory after logging on the FTP server
Remote system	Must access the FTP server either via the Internet or a virtual private network

Configuring the Central System

To configure the central system to use the FTP home directory:

1. Map a drive letter (such as X:) to the FTP home directory.
2. Edit the [Config.xml](#) file.
3. Set the *Local Send Directory* field to *X:\TRANS\Outgoing%REMOTELLOCATIONID%* (where X is the drive letter mapped to the home directory on the FTP server).
4. Set the *Local Receive Directory* field to *X:\TRANS\Incoming%REMOTELLOCATIONID%* (where X is the drive letter mapped to the home directory on the FTP server).

The *Remote Send Directory* and *Remote Receive Directory* are not used in this configuration.

5. If there are any files in the central system's local *Incoming* or *Outgoing* directories, move them to the directories on the FTP server.

Configuring the Remote System

To configure the remote system:

1. Start the system and click *View Application*.
2. Type the password (initially set to *PSI*).
3. Click the *Forms* tab. Double-click *CDMConfiguration*.
4. Set the *Remote Send Directory* field to *TRANS\Outgoing%LOCALLOCATIONID%*
5. Set the *Remote Receive Directory* field to *TRANS\Incoming%LOCALLOCATIONID%*
6. Close the form.
7. Double-click the *CDMLocationsAndGroups* form.
8. If the central location record (usually location ID 0) is not displayed, move to the location record for the central location.
9. Set the *FTP Host* (i.e. *ftp.mycompany.com*), *FTP user ID* and *FTP password* fields to the appropriate values. The *FTP host* field value may be the host name optionally followed by a colon and a port number (if the FTP server does not use the default port 21). For example, if you are connecting to an FTP server that uses port 999, type the host name followed by a colon and the port number: *ftp.mycompany.com:999*
10. Close the form and the application window.
11. Create a batch file that transfers transaction files, processes the transactions and transfers the response files. See [Batch Process for File Transfer at Remote Locations](#) for an example.

Testing the FTP Transfer

1. On the central system, generate a transaction for the remote site, such as a *Request Table Refresh*.
2. On the remote system, double-click the transfer batch file in *Windows Explorer*.
3. On the central system, verify that the files in the *Outgoing* directory for the remote location are gone and that there are now transactions in the *Incoming* directory for the remote site.
4. On the central system, process the transactions. Verify the transactions were processed successfully.

Internet FTP Server

Both the central and remote systems use FTP to transfer files to an FTP server on the Internet. This requires both the central and remote sites to access the FTP server either over the Internet or using a virtual private network.

Configuring The Central System

1. Edit the [Config.xml](#) file.
2. Set the *Local Send Directory* field to *Outgoing%REMOTELLOCATIONID%*
3. Set the *Local Receive Directory* field to *Incoming%REMOTELLOCATIONID%*
4. Set the *Remote Send Directory* field to *TRANS\Incoming%REMOTELLOCATIONID%*
5. Set the *Remote Receive Directory* field to *TRANS\Outgoing%REMOTELLOCATIONID%*
6. Close the configuration form.

7. Click *Edit Locations and Groups* on the *Locations* tab of the central main menu.
8. Move to the location record for the central location (usually location ID 0).
9. Set the *FTP Host* (ie. *ftp.mycompany.com*), *FTP user ID* and *FTP password* fields to the appropriate values. The *FTP host* field value may be the host name optionally followed by a colon and a port number (if the FTP server does not use the default port 21). For example, if you are connecting to an FTP server that uses port 999, type the host name followed by a colon and the port number: *ftp.mycompany.com:999*
10. Close the location form.
11. Create a batch file to transfer and process transaction files from the central location. The batch file can be scheduled to automatically run or executed manually via a button on the main menu. See [Batch Process for File Transfer at Central Locations](#) for a sample batch file. See also [Changing the Process Transactions Menu Options to Also Transfer Files](#).

Configuring the Remote System

1. Start the system and click *View Application*.
2. Type the password (initially set to *PSI*).
3. Click the *Forms* tab. Double-click *CDMConfiguration*.
4. Set the *Remote Send Directory* field to *TRANS\Outgoing%LOCALLOCATIONID%*
5. Set the *Remote Receive Directory* field to *TRANS\Incoming%LOCALLOCATIONID%*
6. Close the form.
7. Double-click the *CDMLocationsAndGroups* form.
8. If the central location record (usually location ID 0) is not displayed, move to the location record for the central location.
9. Set the *FTP Host* (ie. *ftp.mycompany.com*), *FTP user ID* and *FTP password* fields to the appropriate values. The *FTP host* field value may be the host name optionally followed by a colon and a port number (if the FTP server does not use the default port 21). For example, if you are connecting to an FTP server that uses port 999, type the host name followed by a colon and the port number: *ftp.mycompany.com:999*
10. Close the form and the application window.
11. Create a batch file to transfer transaction files, process the transactions, and transfer the response files. See [Batch Process for File Transfer at Remote Locations](#) for a sample batch file.

Testing the FTP Transfer

1. On the central system, generate a transaction for the remote location, such as *Request Table Refresh*.
2. On the central system, transfer the transaction files to/from the FTP server either by clicking the *Process Transactions* button on the menu (if it has been changed to also transfer files), or by double-clicking the transfer batch file in Windows Explorer.
3. On the remote system, transfer the transaction files to/from the FTP server by double-clicking the transfer batch file in Windows Explorer.
4. On the central system, transfer and process transaction files by clicking the *Process Transactions* button on the menu (if it has been changed to also transfer files), or by double-clicking the transfer batch file in Windows Explorer.
5. Verify the transactions were processed successfully.

Remote FTP Server

In the remote FTP server configuration, an FTP server runs at each remote site and the central system connects to the FTP server to transfer transaction files. To use this configuration, the FTP server at each remote site must be visible to the central system either via the Internet or a virtual private network.

This configuration will take more time to process since it has to connect to a different FTP server for each remote site.

System Requirements

Central system	Must access the FTP server either via the Internet or a virtual private network
Remote system	Must map a drive letter to the home directory of the FTP server, which is the default directory after logging on the FTP server

Configuring the Central System

1. Edit the *Config.xml* file.
2. Set the *Local Send Directory* field to *Outgoing%REMOTELLOCATIONID%*
3. Set the *Local Receive Directory* field to *Incoming%REMOTELLOCATIONID%*
4. Set the *Remote Send Directory* field to *TRANS\Outgoing%LOCALLOCATIONID%*
5. Set the *Remote Receive Directory* field to *TRANS\Incoming%LOCALLOCATIONID%*
6. Close the configuration form.
7. Click *Edit Locations and Groups* on the *Locations* tab of the central main menu.
8. **PERFORM THE FOLLOWING STEP FOR EACH REMOTE LOCATION**
Set the *FTP Host* (may be the IP address), *FTP user ID*, and *FTP password* fields to appropriate values for the remote location. The *FTP host* field value may be the host name optionally followed by a colon and a port number (if the FTP server does not use the default port

21).

For example, if you are connecting to an FTP server that uses port 999, type the host name followed by a colon and the port number, e.g. 123.456.789.012:999

9. Close the location form.
10. To configure the central system to not use a central FTP server, leave the *FTP Host* field blank in the central location record.
11. Create a batch file to transfer and process transaction files from the central location. The batch file can be scheduled to automatically run or executed manually via a button on the main menu. See [Batch Process for File Transfer at Central Locations](#) for a sample batch file. See also [Changing the Process Transactions Menu Options to also Transfer Files](#).

Configuring the Remote System

In the remote FTP server configuration, the remote system reads and writes transaction files directly from the home directory of the FTP server that is running on either the remote system itself, or somewhere on the local area network at the remote location.

To configure the remote system to use the FTP home directory:

1. Map a drive letter (such as X:) to the FTP home directory.
2. Start the system and click *View Application*.
3. Type the password (initially set to *PSI*).
4. Click the *Forms* tab. Double-click *CDMConfiguration*.
5. Set the *Local Send Directory* field to *X:\TRANS\Outgoing%REMOTELOCATIONID%* (where X: is the drive letter mapped to the FTP home directory).
6. Set the *Local Receive Directory* field to *X:\TRANS\Incoming%REMOTELOCATIONID%* (where X: is the drive letter mapped to the FTP home directory).
7. Close the form.

Testing the FTP Transfer

1. On the central system, generate a transaction for the remote location, such as *Request Table Refresh*.
2. On the central system, transfer the transaction files to/from the FTP server either by clicking the *Process Transactions* button on the menu (if it has been changed to also transfer files), or by double-clicking the transfer batch file in Windows Explorer.
3. On the remote system, click the *Process Transactions* button on the main menu.
4. On the central system, transfer and process transaction files by clicking the *Process Transactions* button on the menu (if it has been changed to also transfer files), or by double-clicking the transfer batch file in Windows Explorer.
5. Verify the transactions were processed successfully.

Batch Process for File Transfer at Central Locations

At central locations, a batch file similar to the following transfers files, processes the transactions, and then transfers the response files.

```
@echo off
echo Processing transactions files at headquarters...
set JAVA_HOME=C:\Program Files\Java\jre1.5.0_02\bin
set DIR=C:\jakarta-tomcat-5.5.7\webapps\EDM
set CLASSPATH=C:\jakarta-tomcat-5.5.7\shared\lib\EDMjni.jar
"%JAVA_HOME%\java.exe" "-Duser.dir=%DIR%" \-jar
%DIR%\WEB-INF\lib\system.jar "-cfg=%DIR%\WEB-INF\classes\config.xml"
"-user=remote" "-pwd=remote"
"-expr=ShowWarnings(0)+ShowErrors(0)+transferLocationFilesViaFtp('')+ProcessTransactions('')+transferLocationFilesViaFtp('')+Exit()"
```

Notice that the parameter to the *TransferLocationFilesViaFTP()* function is " instead of a location number. This will transfer files to all locations that have a valid FTP host set in their location record.

Make sure the directory names in the *SET* lines are correct.

To configure the system to abort on errors, change the parameter to the *ShowWarnings* and *ShowErrors* functions from 0 to 1.

Batch Process for File Transfer at Remote Locations

At remote sites, the system provides batch files called *transfer_web.bat* and *transfer_ftp.bat* that transfer files, process the transactions, and then transfer the response files. In order for the batch file to operate correctly, the location record for the central location must have a valid transfer host, user ID and password.

If you want the system to abort on errors, change the parameter to the *ShowWarnings* and *ShowErrors* functions from 0 to 1.

Changing the Process Transactions Menu Options to also Transfer Files

For manual control over transferring and processing transaction files, change the *Process All Transactions* and *Process Selected Transactions* options (available from the *Processing* menu of the *Transactions* module) to transfer transaction files both before and after processing the transaction files.

The files are transferred before processing to retrieve any new transactions. The files are transferred after processing to send back status transactions to inform the sender of the results of the transaction processing.

To add FTP file transfer to the transaction processing menu options:

1. Use a text editor to edit the *menu_app_version.xml* file, which is located in the folder containing the system configuration files.
2. Find the menu with the *name="Transactions"*.
3. Find the item with the *name="Process All Transactions"* and change the *action* attribute to:

```
TransferLocationFilesViaFTP(' '*')+ProcessTransactions(' '*')+TransferLocationFilesViaFTP(' '*)
```

4. Find the item with the *name="Process Selected Transactions"* and change the *action* attribute to:

```
TransferLocationFilesViaFTP('')+ProcessTransactions('')+TransferLocationFilesViaFTP('')
```

5. Close and save the menu file.
6. If the system is running on a Web server, open the *Windows Control Panel* and go to *Administrative Tools - Services* to restart the Apache Tomcat service.

To use the web transfer function instead of the FTP transfer function, replace *TransferLocationFilesViaFTP* with *TransferLocationFilesViaWeb* in the instructions above.

Adding File Transfer to the Menu

1. Use a text editor to edit the *menu_app_version.xml* file, which is located in the folder containing the system configuration files.
2. Find the menu with the *name="Transactions"*.
3. Add an item line like this:

```
<item name="Transfer Files"
action="transferLocationFilesViaFtp(' '*')"/>
```

4. Close and save the menu file.
5. If the system is running on a Web server, open the *Windows Control Panel* and go to *Administrative Tools - Services* to restart the Apache Tomcat service.

To use the web transfer function instead of the FTP transfer function, replace *TransferLocationFilesViaFTP* with *TransferLocationFilesViaWeb* in the instructions above.

Managing Files

As with database records, the system can manage files for multiple remote sites. The *Send File* and *Request File* transactions (available from the *Remote PC* menu of the *Transactions* module) are used to send and request files, respectively. The *Synchronize* option for these transactions sends updated versions of the files to the sites when the *SendRemoteFiles* function is executed.

The following steps describe how to manage the file *C:\POS\Print.frm* for all remote sites.

1. Select the *Transactions* module.
2. Select *Request File* from the *Remote PC* menu.
3. Select the desired locations.
4. In the *From File:* field, type *C:\POS\Print.frm*.
5. In the *To File:* field, type *Files\Print%REMOTESITEID%.frm_*.
6. To overwrite the existing copy of the file at central, select the *Overwrite* checkbox.
7. Select the *Synchronize* checkbox so the system will remember which files go to each site.
8. Transfer and process transactions with all sites.

In this example, the folder *C:\jakarta-tomcat-5.5.7\webapps\EDMFiles* is created on the central server. This folder contains a file from each site with the respective site number specified in the filename.

You can include the following variables in the remote file and folder names to send and request files from multiple stores with one command.

Variable	What it is replaced with
%THISSITEID%	The ID of this site
%THISSITENAME%	The name of this site
%THISCUSTOMID%	The custom ID of this site
%REMOTESITEID%	The remote site ID
%REMOTESITENAME%	The remote site name
%REMOTECUSTOMID%	The remote site custom ID
%CENTRALSITEID%	The central site ID
%CENTRALSITENAME%	The central site name
%CENTRALCUSTOMID%	The central site custom ID

Sending Files to Sites

Continuing with our example above, to update the *Print.frm* file for site 101:

1. Edit *C:\jakarta-tomcat-5.5.7\webapps\EDMFiles\Print_101.frm*.
2. Select *Transactions – Send Updated Remote Files* on the EDM central main menu. To send the file whether it is updated or not, select *Transactions – Send All Remote Files*.

SendRemoteFiles reads the records in the *CDMRemoteFiles* table to determine which files to send to which sites. If the file is updated (or if the *sendUpdatedOnly* flag is false), the file is sent to the site via the *SendFile* transaction.

Automatically Sending Updated Files

To schedule the *SendRemoteFiles* function to execute periodically, add the following line to the *Config.xml* file. Define the desired values for the *frequency* and *time* parameters.

```
<schedule frequency = "DAILY" time = "4:15 AM" task = "SendRemoteFiles(
true, '*', '', '-Remote File Package', '', true )" />
```

The *SendRemoteFiles* function parameters include:

Parameter	Description
sendUpdatedOnly	True to send only updated files
siteList	Comma-delimited list of site numbers where you are sending files * = all Leave blank to prompt user to select the sites.
effectiveDate	The date to copy the file to the destination

packageName	The name of the package to contain the transactions Leave blank to prompt user to select.
packageDescription	The description of the package if a new package is to be created.
commit	True to commit the package that contains the transactions

Sharing a File Across Multiple Sites

Send the file to multiple sites with the *Synchronize* option enabled.

For example, if sites 101 and 102 use the same *Print.frm* file, create a *Print_Group_A.frm* file and send it to both sites with the *Synchronize* option enabled. The system remembers that file goes to both sites when it is updated and the *SendRemoteFiles* function is called.

If the system is already set up to send *Print_101.frm* to site 101, the system will send both files. You can either delete the file that should no longer be sent -OR- delete the respective record in the *CDMRemoteFiles* table.

Polling Data

The transactional data (sales, inventory usage, cash management) that is generated at remote sites can be retrieved at central using EDM's powerful and flexible polling capabilities. The following describes the different methods for polling data.

Methods for Polling Data

Method	Description	Advantages	Disadvantages
Detect Changes with Snapshots	Compares the current data in the table to the last snapshot of the data and sends the difference to central as transactions. This method is appropriate for large tables with relatively few changes, such as an employee table or tables in which any record may have been changed or deleted.	<ul style="list-style-type: none"> • Easy to set up • Changes to any record are detected 	Processing may take a long time for a large number of changes
Refresh Entire Table	Sends an entire table to central where it is compared against existing data. This method is appropriate for tables with few records or tables in which any record may have been changed or deleted.	<ul style="list-style-type: none"> • Easy to set up • Changes to any record are detected 	Processing may take a long time for large tables
Refresh Partial Table	Sends selected records and fields of a table to central where data is compared against existing data. This method is appropriate for tables that have many added or changed records and the records can be selected. For example, a POS system generates many new order records during the day and stamps each record with a business date. A partial table refresh only sends the records for the last day instead of every record in the table.	When only certain records in a table can be added or changed, only those records are sent to central.	<ul style="list-style-type: none"> • A query must be created to select the records • The system cannot detect deleted records

See Also

[Polling Aloha POS \(TM\) Data](#)

Detect Changes with Snapshots

If any records in a table may be inserted/updated/deleted, but relatively few changes are made to the table, then changes to the table should be detected by comparing a current snapshot of the table with a previous snapshot. The detected changes are sent as insert, update or delete transactions to the central location.

Adding a Snapshot Table List

See [Adding a Snapshot](#).

Comparing the Snapshots Daily

Execute the following command from the installation directory. The *snapshotName* should be the name of the new snapshot you created. See *compare_menus.bat* in the installation folder for an example.

```
"C:\EDMWeb\go"  
"-expr=ShowWarnings(0)+ShowErrors(0)+GenerateSnapshot('_snapshotName_')+Exit()" "-nomenu"
```

Refresh Entire Table

If many records in a relatively small table may be inserted/updated/deleted, then the entire table should be sent to central for comparison against existing data.

Adding a Snapshot Table List

See [Adding a Snapshot](#).

Comparing the Snapshots Daily

Execute the following command from the system installation directory. The *snapshotName* should be the name of the new snapshot you created. The *centralLocationId* should be the location ID of the central location.

```
"C:\EDMWeb\go"  
"-expr=ShowWarnings(0)+ShowErrors(0)+SendTableRefresh('_snapshotName_', '_centralLocationId_', '0-Default Package')+Exit()"
```

Refresh Partial Table

The system can be configured to only poll records from a table that are added or updated. For example, a POS system generates many new order records during the day and stamps each record with a business date. A partial table refresh only sends the records for the last day instead of every record in the table.

Creating the Polling Queries

You need to create polling queries that select the records and fields to be polled. Polling queries must follow these rules:

- The table to poll is the first table in the query's *FROM* clause.
- The table to poll must have a primary or unique key.
- Any columns from the table to poll may be included.
- The column names in the query must match the column names in the table.
- The polling query must include all the primary or unique key fields in the table in the sequence that they are specified in the key.
- The polling query must sort each of the key fields in ascending order.
- No calculated fields or fields from other tables may be included in the result set.

Sample Queries

Below are some sample queries that select certain records and fields from an order table and an order item table. Note that only records within seven days of the current business date are selected and that not all the fields from the table are selected.

Query to get selected fields from all the order records for the last 7 days

```
SELECT BusinessDate, RegisterNum, OrderNum, ItemCount, SubTotal, Tax  
FROM tblOrder  
WHERE (BusinessDate > NumberToDate (DateToNumber (CharToDate (QueryFirst ('BusinessDate', 'tblBusinessDate', ''), 'GD')) - 7))  
ORDER BY BusinessDate, RegisterNum, OrderNum
```

Query to get selected fields from all the order item records for the last 7 days

```
SELECT BusinessDate, RegisterNum, OrderNum, SequenceNum, LongDesc, ItemNum, Quantity, Weight, Price  
FROM tblOrderItem  
WHERE (BusinessDate > NumberToDate (DateToNumber (CharToDate (QueryFirst ('BusinessDate', 'tblBusinessDate', ''), 'GD')) - 7))  
ORDER BY BusinessDate, RegisterNum, OrderNum, SequenceNum
```


Creating the Polling Snapshot

Once the polling queries are created, create a snapshot that lists the queries to include in the polling process. See [Adding a Snapshot](#) for detailed instructions.

Sending the Polling Data to Central

To query the polling data and create the transaction to send to the central location, execute the following command from the EDM installation directory. The *snapshotName* should be the name of the new snapshot you created. The *centralLocationId* should be the location ID of the central location.

```
"C:\EDMWeb\go"  
"-expr=ShowWarnings(0)+ShowErrors(0)+SendTableRefresh('_snapshotName_', '_centralLocationId_', '0-Default Package')+Exit()"
```

The command may be called as part of the nightly communication, end-of-day, or in a manually executed batch file. It is recommended that the command be called before the communications session is started.

Polling Aloha POS (TM) Data

The system provides a complete polling solution for Aloha POS (TM) users. Transactional data is collected from the daily folders, compressed, transmitted to headquarters, and loaded into central tables using a fast import process. To control the amount of data that is transferred, you can specify the maximum number of days to include in one polling package.

The central database has the same structures as the Aloha transaction tables plus a site number and a polling date column, so querying the data or creating corporate reports is easy.

See Also

[Remote Setup](#)
[Central Setup](#)
[Managing Polling](#)

Remote Setup

The following describes the functions related to [Polling Aloha POS \(TM\) Data](#): *setupAlohaPolling* and *sendAlohaPollData*.

setupAlohaPolling

After installing the remote system, call the *setupAlohaPolling* function to create the polling configuration tables and set the default polling values, including:

- The folder that contains the daily grind folders
- The maximum number of days to include in any one polling package
- The type of central database that will be importing the data

To call the *setupAlohaPolling* function from a command line:

```
"C:\EDMWeb\go" "-expr=setupAlohaPolling( 'C:\Aloha', 10, 'SQLSERVER', 'Poll  
Tables' )+Exit()" "-nomenu"
```

sendAlohaPollData

The remote site must also be configured to send the polling data periodically by calling the *sendAlohaPollData* function. Normally, data is sent one time daily, so this function can be called during end of day processing.

To call the *sendAlohaPollData* function from a command line:

```
"C:\EDMWeb\go" "-expr=sendAlohaPollData()+Exit()" "-nomenu"
```

Central Setup

This page describes setting up polling at the central site when [polling Aloha POS \(TM\) data](#).

Prerequisites

The following assumes you have:

- Installed the system at headquarters
- Installed the system at a remote site on a lab system
- Setup polling on the remote system

Setting up Polling at the Central Site

1. If it does not already exist, add a site record for the remote site where polling is setup.
2. Select the *Transactions* module.
3. Select *Request Table List* from the *Remote PC* menu.
4. Select the remote site, and then click *OK*. Click *OK* again to select the default package. Click *Yes* to commit the transaction.
5. Transfer the transactions to the remote site and process the transactions.
6. Transfer the transactions from the remote site to central and process the transactions.
7. Open the the *Site Manager* and click *Setup Site Polling*.
8. Set the following values:

Unique polling configuration ID	Type 1 for Aloha polling
Maximum polling days to return at once	Type the maximum number of days to include in a polling package. For example, if you set this value to 7 and there are 30 days of history at the site, then the system will send the 7 most recent, unsent days of data each day until all 30 days are sent. Type 0 to disable polling.
Central database type	Specify the type of central database that will import the data, eg. SQLSERVER or ORACLE
Folder that contains dated grind folders	Type the full path to the folder that contains the date folders (e.g. 20130829) that are created when the daily grind is run
Polled Tables	Add the tables to poll and include the prefix <i>AlohaPoll_</i> (two underscores) before each table name For example, to poll the <i>Gnditem</i> and <i>Gndsale</i> tables, add <i>AlohaPollGnditem</i> and <i>AlohaPoll_Gndsale</i> .

9. Close the form and commit the transactions.

The updated polling configuration data is sent to the remote site and the polling data is sent when the *sendAlohaPollData* function is executed. The system sends a polling package if any new or updated polling data is found. A polling package is always sent as a single zipped file even though it may contain data for multiple tables and days.

Managing Polling

The following relates to [Polling Aloha POS \(TM\) Data](#).

Viewing Polling Status

To view the polling status for all sites:

1. Select the *Sites* module.
2. Select the *Site Manager* from the *Setup* menu.

The following table describes the polling status color codes

Color	Polling Status
Green	The last poll date is yesterday or today

Yellow	The last poll date is earlier than yesterday
Red	An error occurred when polling data
Light gray	Polling is disabled

Automatic Retry of Failed Polling

If the central system fails to process a polling transaction, or if the system receives data for a newer date when data for an older date is missing, a transaction is sent back to the site to request a resend of the failed or missing poll data. Only the table and date that actually failed (or is missing) are requested.

Manually Requesting Poll Data Resend

If polling data is lost or damaged in the central database, users can request a resend from the sites.

1. Select the *Setup* module.
2. Expand the *Application* menu.
3. Expand the *Tables* menu.
4. Select *PollInfoPOLLSTAMP*.
5. Delete the rows for the data to resend.

For example, to have site 1234 resend the data for the *Gnditem* table for August 19, 2013, delete the following row, then close the table and commit the transactions:

```
CDMLOCID POLLID TABLENAME POLLDATE STAMP 1234 1 AlohaPollGnditem 8/31/2013 12:00 AM <NULL>
```

Changing Polling Configuration

1. Select the *Sites* module.
2. Select *Site Manager* from the *Setup* menu.
3. Click *Setup Site Polling*.
4. Select the desired sites.
5. Edit the following values.

Unique polling configuration ID	Type 1 for Aloha polling
Maximum polling days to return at once	Type the maximum number of days to include in a polling package. For example, if you set this value to 7 and there are 30 days of history at the site, then the system will send the 7 most recent, unsent days of data each day until all 30 days are sent. Type 0 to disable polling.
Central database type	Specify the type of central database that will import the data, eg. SQLSERVER or ORACLE
Folder that contains dated grind folders	Type the full path to the folder that contains the date folders (e.g. 20130829) that are created when the daily grind is run
Polled Tables	Add the tables to poll and include the prefix <i>AlohaPoll__</i> (two underscores) before each table name For example, to poll the <i>Gnditem</i> and <i>Gndsale</i> tables, add <i>AlohaPollGnditem</i> and <i>AlohaPoll_Gndsale</i> .

Security

Security for the system is accomplished at two different levels.

Internet Security	Includes the security of the Web server and traffic between the browsers and server
Application Security	Includes authenticating and authorizing users to determine the functions a user may perform

See Also

[Permission Types](#)
[Permissions](#)

Roles

Assigning Permissions to Roles

Editing Security Information

File Security

Preventing Embedded Executable Scripts

Internet Security

When the system is visible via the Internet, the following actions need to be taken by the system administrator to ensure the system is secure from unauthorized users and attackers:

- Set up a firewall where only the required ports are visible. For example, if only your Web server should be visible on the Internet, disable all ports except port 80.
- Use Secure Socket Layer (SSL) to encrypt all traffic between browsers and the server (see [Using Secure Socket Layer \(SSL\) to Protect Data](#))
- Change all default EDM and database passwords
- Turn off all unused services, such as FTP and email servers
- Remove sample applications from the Web server
- Review the latest published security vulnerabilities and apply the latest security patches for the operating system, applications and databases

Application Security

For data security, each user logs in with a unique password and each generated transaction is logged in the audit trail with the associated user ID. Regardless of the user who initiated the process, the system uses Administrator permissions during installation and transaction processing in order to update all tables.

Default Users

When the system is first installed, the following users are predefined.

Important Note

Change the passwords for all three default users immediately after installation.

User ID	Password	Description
Admin	Admin	<ul style="list-style-type: none">• Administrator user• All permissions granted• Can edit users and roles
Central	Central	<ul style="list-style-type: none">• Central system user• Can use all forms and menu options• Can edit data in any table• Cannot modify forms, queries, users or roles
Remote	Remote	<ul style="list-style-type: none">• Remote system user• Can process and view transactions• Cannot use forms or edit data

Permission Types

The security system allows you to control access to the following types of information:

Permission Type	Description
Company	A central database of site data
Table	A table containing data
Data	Data in a table
Query	A query that can be used to select, sort or update data in a table
Form	A window that can be displayed to make it easy to edit table data
Function	A system function such as <i>ProcessTransactions</i>

Menu	A branch of the menu that appears in the left window of the main menu
MenuItem	A menu item that appears in the right window of the main menu
Sites	A list of sites accessible to a user
Rowset	A named subset of rows and columns within a table

Permissions

The following describes the permissions that can be assigned to each [Permission Type](#):

Permission	Description
View	Allow user to view the information
Add	Allow user to add new information of the same type
Edit	Allow user to update the information
Delete	Allow user to delete the information
Execute	Allow user to execute the object, such as a function

Roles

Each user is assigned a role, which is a set of permissions that every user with that role shares.

Administrator

All permissions granted.

Central User

A *Central* user is usually set up with the following permissions (where * in the *Object Name* field indicates any name).

Permission Type	Object Name	Permissions
Company	*	View
Table	*	View/Add/Edit/Delete
Query	*	View/Add
Form	*	View
Function	*	View/Execute
Menu	*	View
MenuItem	*	View/Execute
Data	*	View/Add/Edit/Delete
Sites	*	View/Add/Edit/Delete

Remote User

A *Remote* user is usually only allowed to process and view transactions (where * in the *Object Name* field indicates any name). Because the user ID and password for *Remote* users is easily discovered by personnel at remote sites, the role needs to have very limited capabilities so that malicious users do not damage the system. The default security policy only grants permission to required functions, such as processing transactions and viewing the audit trail. Users logged in using the *Remote* role may not call functions that would update data or send transactions to central unless the *Remote* role is modified by the system administrator.

Permission Type	Object Name	Permissions
Company	*	View
Table	*	
Query	*	
Form	CDMAudit	View
Form	CDMAuditFilter	View
Function	ProcessTransactions	View/Execute
Menu	Transactions	View
MenuItem	Process All Transactions	View/Execute
MenuItem	View Transactions	View/Execute
Data	CDMAudit	View
Data	CDMPackage	View/Add
Sites	*	View

Assigning Permissions to Roles

When assigning permissions to a role, use the following list to help you understand what each permission means for each type of information. Setting *Name* to * indicates "all items".

Permission Type	Name	View	Add	Edit	Delete	Execute
Company	Name of company	Access the company				
Table	Name of table	Open table as a grid	Add table to application	Update table design	Delete table from application	
Query	Name of query	Open select query as grid or execute action query	Add query to application	Update and save query design	Delete query from application	
Form	Name of form	Open the form	Add form to application	Update and save form design	Delete form from application	
Function	Name of function					Execute the function
Menu	Name of menu	View menu on main client window	Add new menu	Update menu properties	Delete menu	
MenuItem	Name of menu item	View/select menu item on main menu	Add a new menu item	Update a menu item	Delete a menu item	Execute the menu item action
Data	Name of table	View data in a table	Add data to a table	Update data in a table	Delete data from a table	
Sites	List of sites (e.g. 1,2,5-9)	Select sites on central forms and functions	Add sites to site table	Update sites in site table	Delete sites from site table	
Rowset	Name of Rowset	View row/column data in a table	Add a row	Update data in a row/column	Delete a row	

Editing Security Information

The following security configuration options are accessible from the *Users* menu of the *Setup* module.

Change Password	Enables users to change their password at any time
Edit Users	Enables users to add/change/delete users from the system. This option is only available to the <i>Administrator</i> role.
Edit Roles	Enables users to add/delete roles and add/change/delete role permissions. This option is only available to the <i>Administrator</i> role. The <i>Administrator</i> role cannot be modified or deleted.

File Security

To protect the central system when running on a Web server, only files in the Web application folder and its subfolders (except *WEB-INF*) may be sent or received by the server. Only central Web server files are protected. Standalone central systems and remote systems that don't usually run Web servers don't need the same level of security. Remote systems with Web servers need to be updatable by transactions, so they must allow access to all folders. Inaccessible folders are checked when creating and applying transactions.

Preventing Embedded Executable Scripts

To prevent malicious users from embedding executable scripts in system-generated HTML, inserted characters (such as <,>,(,),#,&) are replaced with the equivalent HTML entity string, such as:

```
&lt; &gt;
```

For example, when a difference report is generated and an item name has a < character in it, the HTML in the report will replace the < character with:

```
&lt;
```

Supported Databases

The following lists the databases that EDM supports.

dBASE IV Using the DataDirect ODBC Driver

This page describes using EDM with dBASE IV using the DataDirect ODBC driver. To go to the Supported Databases page, click [here](#).

The system can create, read and write dBASE IV (.dbf) files using the DataDirect ODBC driver. To configure the system, take the following steps:

1. Set the *type* property in the *systemDataSource* and/or *dataSource* sections in the *config.xml* file to *DBASEIVDD*.
2. Set the *database* property in the *systemDataSource* and/or *dataSource* sections in the *config.xml* file to the full path of the folder containing the .DBF files.

dBASE IV Using the Microsoft ODBC Driver

This page describes using EDM with dBASE IV using the Microsoft ODBC driver. To go to the Supported Databases page, click [here](#).

The system can create, read and write dBASE IV (.dbf) files using the Microsoft ODBC driver. To configure the system, take the following steps:

1. Set the *type* property in the *systemDataSource* and/or *dataSource* sections in the *config.xml* file to *DBASEIV*.
2. Set the *database* property in the *systemDataSource* and/or *dataSource* sections in the *config.xml* file to the full path of the folder containing the .DBF files.

Microsoft Access 97

This page describes using EDM with Microsoft Access 97. To go to the Supported Databases page, click [here](#).

The system can create, read and write Microsoft Access 97 databases. To configure the system, take the following steps:

1. Set the *type* property in the *systemDataSource* and/or *dataSource* sections in the *config.xml* file to *ACCESS*.
2. Set the *database* property in the *systemDataSource* and/or *dataSource* sections in the *config.xml* file to the full path of the .MDB file.

Microsoft Access 2000

This page describes using EDM with Microsoft Access 2000. To go to the Supported Databases page, click [here](#).

The system can create, read and write Microsoft Access 2000 databases. To support the Access 2000 database format, take the following steps:

1. Create the *Program Files\Common Files\Microsoft Shared\DAO* folder.

2. Copy *DAO360.DLL* from the *Access2K* directory of the installation CD to the *Program Files\Common Files\Microsoft Shared\DAO* folder.
3. Copy *JNIODBC.DLL* from the *Access2K* directory of the installation CD to the installation directory.
4. Set the *type* property in the *systemDataSource* and/or *dataSource* sections in the *config.xml* file to *ACCESS*.
5. Set the *database* property in the *systemDataSource* and/or *dataSource* sections in the *config.xml* file to the full path of the *.MDB* file.

Microsoft SQL Server 2000 Using JDBC Driver

This page describes using EDM with Microsoft SQL Server 2000 using JDBC Driver. To go to the Supported Databases page, click [here](#).

The system can create, read and write Microsoft SQL Server 2000 databases with full Unicode support including multi-byte Asian languages. To configure the system, take the following steps:

1. Set the **Your Locale** field in the **Regional Options** of the **Control Panel** to the correct locale so the system loads the correct fonts.
2. Set the *type* property in the *systemDataSource* and/or *dataSource* sections in the *config.xml* file to *SQLSERVER_JDBC*.
3. Set the *database* property in the *systemDataSource* and/or *dataSource* sections in the *config.xml* file to the name of the SQL Server database.
4. Set the *server* property in the *systemDataSource* and/or *dataSource* sections in the *config.xml* file to name of the server or *localhost* if the server is running on the same machine as the application.
5. Set the *user* property in the *systemDataSource* and/or *dataSource* sections in the *config.xml* file to the user ID used to access the database.
6. Set the *password* property in the *systemDataSource* and/or *dataSource* sections in the *config.xml* file to the password for the user ID used to access the database.

Installing the Latest JDBC Driver from Microsoft

1. Download the driver files from Microsoft at: <http://msdn.microsoft.com/en-us/sqlserver/aa937724>
2. Install the files (*mssqlserver.jar*, *msbase.jar*, and *msutil.jar*) in the *Jre/Lib/Ext* subdirectory of the installation directory.

Microsoft SQL Server 2005 Using JDBC Driver

This page describes using EDM with Microsoft SQL Server 2005 using JDBC Driver. To go to the Supported Databases page, click [here](#).

The system can create, read and write Microsoft SQL Server 2005 databases with full Unicode support including multi-byte Asian languages.

Setting Up SQL Server 2005

To set up SQL Server 2005 take the following steps:

1. Install SQL Server 2005 and be sure to set up the system to use SQL Server authorization.
2. Install the SQL Server Management Studio.
3. Open the SQL Server Configuration Manager.
4. Select SQL Server 2005 Network Configuration > Protocols for SQLEXPRESS.
5. Right-click the TCP/IP node, and select Enable.
6. Close the SQL Server Configuration Manager.
7. Restart the SQL Server service.
8. Open the SQL Server Configuration Manager.
9. Select SQL Server 2005 Network Configuration > Protocols for SQLEXPRESS.
10. Right-click the TCP/IP node, and select Properties.
11. Select the IP Addresses tab in the TCP/IP dialog.
12. In the IP ALL section, note the port value specified in TCP Dynamic Ports.

Configuring EDM to Use SQL Server 2005

To configure the system to use SQL Server 2005, take the following steps:

1. Set the **Your Locale** field in the **Regional Options** of the **Control Panel** to the correct locale so the system loads the correct fonts.
2. Set the *type* property in the *systemDataSource* and/or *dataSource* sections in the *config.xml* file to *SQLSERVER_JDBC*.
3. Set the *database* property in the *systemDataSource* and/or *dataSource* sections in the *config.xml* file to the name of the SQL Server database.
4. Set the *server* property in the *systemDataSource* and/or *dataSource* sections in the *config.xml* file to name of the server followed by a colon (:) and the TCP Dynamic Port number or *localhost:portnumber* if the server is running on the same machine as the application. For example, *MyServer:49189* or *localhost:49189*.
5. Set the *user* property in the *systemDataSource* and/or *dataSource* sections in the *config.xml* file to the user ID used to access the database.
6. Set the *password* property in the *systemDataSource* and/or *dataSource* sections in the *config.xml* file to the password for the user ID used to access the database.

Installing the Latest JDBC Driver from Microsoft

1. Download the driver files from Microsoft at: <http://msdn.microsoft.com/en-us/sqlserver/aa937724>
2. Install the file `sqljdbc.jar` in the "lib" folder of the installation directory.
3. To use the improved memory functions of the latest driver for 2005 when using a connection string, add `;"responseBuffering=adaptive"` to the end of the connection string.

Using Integrated Security

The JDBC driver for SQL Server 2005 and later enables the user to use integrated security so that the user id and password for the database do not have to appear in the `config.xml` file.

Users who want to use integrated security can add a "connect" attribute to the "systemDataSource" and "dataSource" elements like this:

```
<systemDataSource
  name = "EDMWeb"
  version = "1"
  type = "SQLSERVER_JDBC"
  server = "localhost"
  database = "EDMWeb"
  connect =
"jdbc:sqlserver://localhost;databaseName=EDMWeb;integratedSecurity=true" />
```

In addition to specifying "integratedSecurity=true" in the connection string, to use Windows security instead of a user name and password from `config.xml`, the following files must be in the correct folders:

- **sqljdbc.jar** must be in the lib folder of the system installation folder. The file is included in the config/common folder of the installation folder on the installation CD image.
- **sqljdbc_auth.dll** must be on the system path such as in the Windows/system32 folder. The file is included in the config/common folder of the installation folder on the installation CD image. For 64-bit systems, put the `sqljdbc_auth.dll` from the `config/common/sqljdbc_auth_64bit` directory to the `Windows/SysWOW64` directory.
- If the error "java.lang.SecurityException: sealing violation: package oracle.jdbc.driver is sealed" is in the log, then deleting the Oracle JDBC driver (`ojdbc6.jar`) from the "lib" folder will resolve the issue.

Microsoft SQL Server Using ODBC Driver

This page describes using EDM with Microsoft SQL Server Using ODBC Driver. To go to the Supported Databases page, click [here](#).

The system can create, read and write Microsoft SQL Server databases. To configure the system, take the following steps:

1. Set the `type` property in the `systemDataSource` and/or `dataSource` sections in the `config.xml` file to `SQLSERVER`.
2. Set the `database` property in the `systemDataSource` and/or `dataSource` sections in the `config.xml` file to the name of the SQL Server database.
3. Set the `server` property in the `systemDataSource` and/or `dataSource` sections in the `config.xml` file to name of the server or (*local*) if the server is running on the same machine as the application.

MSDE Using JDBC Driver

This page describes using EDM with MSDE using JDBC Driver. To go to the Supported Databases page, click [here](#).

The system can create, read and write MSDE databases with full Unicode support including multi-byte Asian languages. To configure the system, take the following steps:

1. Set the **Your Locale** field in the **Regional Options** of the **Control Panel** to the correct locale so the system loads the correct fonts.
2. Be sure SQL Server was installed with the `DISABLENETWORKPROTOCOLS=0` and the `SECURITYMODE=SQL` so that the system can connect to the database using SQL Server security instead of Windows security. The installation command should be similar to the following, though you should have a different password for the sa user and the instancename may be any unique name for the instance:
`setup sapwd="mypassword" instancename="EDM" DISABLENETWORKPROTOCOLS=0 SECURITYMODE=SQL`
3. Set the `type` property in the `systemDataSource` and/or `dataSource` sections in the `config.xml` file to `SQLSERVER_JDBC`.
4. Set the `database` property in the `systemDataSource` and/or `dataSource` sections in the `config.xml` file to the name of the SQL Server database.

5. Set the *server* property in the *systemDataSource* and/or *dataSource* sections in the *config.xml* file to name of the server followed by the MSDE instance name or *localhost* followed by the MSDE instance name if the server is running on the same machine as the application. For example, *server = "MyServer\MyInstance"* or *server = "localhost\MyInstance"*.
6. Set the *user* property in the *systemDataSource* and/or *dataSource* sections in the *config.xml* file to the user ID used to access the database.
7. Set the *password* property in the *systemDataSource* and/or *dataSource* sections in the *config.xml* file to the password for the user ID used to access the database.

Installing the Latest JDBC Driver from Microsoft

1. Download the driver files from Microsoft at: <http://msdn.microsoft.com/en-us/sqlserver/aa937724>
2. Install the files (*mssqlserver.jar*, *msbase.jar*, and *msutil.jar*) in the *Jre/Lib/Ext* subdirectory of the installation directory.

Oracle

This page describes using EDM with Oracle. To go to the Supported Databases page, click [here](#).

Enterprise Data Manager supports the Oracle database for use as the central database. The user must create the database before installing Enterprise Data Manager.

Setting Up the Central Database

To use Oracle as the central database, take the following steps:

1. Create an Oracle database (usually named EDMWebHQ).
2. Create a user on the new database with DBA privileges.
3. Copy the contents of the installation CD to a temp directory on the hard drive.
4. Use Notepad to edit **data\config\config_central_app_version.xml** in the temp directory and change the **systemDataSource** section so that it looks like the following:

```
name = "EDMWebHQ"
type = "ORACLE"
database = "databasename"
server = "servername"
user = "username"
password = "password"
```

5. Close and save the file.
6. Run the **setup_central_app_version.bat** file in the temp installation directory to start the installation.

Assigning Central Database Users

If desired, you can create another role for normal users to select, insert, update, and delete records in the database by taking the following steps:

1. Log into the EDMWebHQ database using the dba user id and password.
2. Create a role called RMUSER.
3. Grant privileges to RMUSER for each table in the database.

To grant privileges on the system tables, you could use the following:

```
grant select, insert, update, delete on CDMAUDIT to rmuser;
grant select, insert, update, delete on CDMLOCATION to rmuser;
grant select, insert, update, delete on CDMLOCATIONGROUP to rmuser;
grant select, insert, update, delete on CDMREMOTETABLES to rmuser;
grant select, insert, update, delete on CDMREMOTEFILES to rmuser;
grant select, insert, update, delete on CDMVALUE to rmuser;
grant select, insert, update, delete on CDMPACKAGE to rmuser;
```

To grant privileges on application tables, use a grant statement like the following:

```
grant select, insert, update, delete on tablename to rmuser;
```

Assign data entry personnel to the RMUSER role.

Using Oracle as the Central Database

To use Oracle as the central database using the Oracle JDBC connection, take the following steps:

1. Create an Oracle database (usually named 'EDMWebHQ').
2. Create a user on the new database with DBA privileges. (This may be the same name of 'EDMWebHQ'.)
3. Copy the contents of the installation CD to a temp directory on the hard drive.
4. Use Notepad to edit **dataconfig\config_central_app_version.xml** in the temp directory and change the **systemDataSource** section so that it looks like the following: name = "EDMWebHQ" version = "1" type = "ORACLE_JDBC" database = "*databaseName/tablespace name*" server = "*servername*" user = "*username*" password = "*password*" sid = "*SID – such as "orcl"*" port = "*port of the database instance – such as "1521"*" then close and save the file.
5. Run the **setup_central_app_version.bat** file in the temp installation directory to start the installation.

Assigning Central Database Users

If desired, you can create another role for normal users to select, insert, update, and delete records in the database by taking the following steps:

1. Log into the EDMWebHQ database using the dba user id and password.
2. Create a role called RMUSER.
3. Grant privileges to RMUSER for each table in the database.

To grant privileges on the system tables you could use the following:

```
grant select, insert, update, delete on CDMAUDIT to rmuser;
grant select, insert, update, delete on CDMLOCATION to rmuser;
grant select, insert, update, delete on CDMLOCATIONGROUP to rmuser;
grant select, insert, update, delete on CDMREMOTETABLES to rmuser;
grant select, insert, update, delete on CDMREMOTEFILES to rmuser;
grant select, insert, update, delete on CDMVALUE to rmuser;
grant select, insert, update, delete on CDMPACKAGE to rmuser;
```

For application tables, you would also grant privileges as follows:

To grant privileges on other application tables, use a grant statement like the following:

```
grant select, insert, update, delete on tablename to rmuser;
```

Assign data entry personnel to the RMUSER role.

PostgreSQL

This page describes using EDM with MSDE using JDBC Driver. To go to the Supported Databases page, click [here](#).

The system can create, read and write PostgreSQL databases. To use PostgreSQL, set the *type* property in the *systemDataSource* and/or *dataSource* sections in the *config.xml* file to *POSTGRESQL_JDBC*.

Connection strings may also be used of the form:

```
jdbc:postgresql://hostname/databasename?user=myusername&password=myspassword&stringtype=unspecified"
```

Configuration Files

The system comes with a number of configuration files that contain information about the installation. You can modify these files to customize your installation.

System	Default configuration file folder
Remote	C:\EDMWeb
Central	C:\jakarta-tomcat-5.5.7\webapps\EDMWEB-INF\classes

See Also

[Config.xml](#)
[Snapshots](#)
[Main Menu](#)
[Other Configuration Files](#)

Config.xml

The config.xml file contains all the configuration settings for the EDM system, including references to other configuration files, such as the menu file.

Sample File

Config.xml

```
<?xml version="1.0"?>

<config version = "4">
  <company
    isCentralSite = "Y"
    centralSiteNumber = "0"
    centralSiteName = "HQ"
    thisSiteNumber = "0"
    thisSiteName = "HQ"
    schemaFile = "schema_iris.xml"
    menuFile = "menu_iris_3.7.4_PR10.xml"
    snapshotFile = "snapshots_iris_3.7.4_PR10.xml"
    transientFieldFile = "transients_iris_3.7.4_PR10.xml"
    transactionConversionFile = "convert_iris_3.7.4_PR10.xml"
    localSendDir = "Outgoing\%REMOTELOCATIONID%"
    localReceiveDir = "Incoming\%REMOTELOCATIONID%"
    remoteSendDir = "Outgoing\%LOCALLOCATIONID%"
    remoteReceiveDir = "Incoming\%LOCALLOCATIONID%"
    auditHistoryDays = "14"
    auditPurgeTime = "6:00 AM"
    transactionProcessingTimes = "5:00 AM"
    bulkInsertMaxBufferSize = 500000
    ftpHostClass = "EDMserver.ftp.FTPHostPSI"
    promptForCommitOnClose = "Y"
    useDefaultPackage = "Y"
    effectiveDateRequired = "N"
    configXENIALFile = "configurationXENIAL.xml"
    irisItemImageDir = "Iris/Images"
    irisItemSoundDir = "Iris/Sounds"
    irisItemForm = "tbl_ItemMaster 374PR10">
  <systemDataSource
    name = "EDMWebHQ"
    type = "SQLSERVER_JDBC"
    database = "EDMWebHQ"
    server = "localhost"
    user = "sa"
    password = ""/>
  <serverPlugin class = "EDMserver.plugin.UserServerPlugins"/>
  <clientPlugin class = "EDMclient.plugin.UserClientPlugins"/>
  <clientPlugin class = "EDMclient.plugin.IrisClientPlugins"/>
</company>
</config>
```

Setting Descriptions

General Entries

Setting	Description
---------	-------------

allowMultiSiteEditing	Y - (default) Allows the user to select multiple sites when opening forms N - Only allow a single site to be selected when opening forms
appFile	The name of the file containing the queries and forms used to work with the data
bulkInsertMaxBufferSize	The Maximum Buffer Size to use before performing the bulk insert. This is used when performing operations that insert a lot of records, such as importing transactions from a file. * The default is 100,000 insert statements. * The minimum is 50,000 insert statements.
centralSiteName	The name of the central site
centralSiteNumber	The number of the central site
clientBackgroundColor	The background color used in the main UI of EDM. Valid entries are a comma separated list of RGB values or a valid Java color name. Java color names are white, lightGray, gray, darkGray, black, red, pink, orange, yellow, green, magenta, cyan, and blue. The default is "black".
clientPlugin	Class to load as client plugin functions
closeFormsOnPackageOpen	Y - Close any open forms when a different package is selected. N - (default) Leave forms open when a different package is selected.
configXENIALFile	The file holding configuration data for the Xenial Connector. If this is not defined, the Xenial Connector will not be activated. (configurationXENIAL.xml by default)
copyright	Copyright information to display on splash screen
dataSource	For each application database you plan to manage with the system, you must have a <i>dataSource</i> element in the configuration file. Each <i>dataSource</i> element has the same properties as the <i>systemDataSource</i> element.
ftpHostClass	Class used for FTP communication. Defaults to <i>EDMserver.ftp.FTPHostPSI</i> . Alternatives that may work differently with a given FTP server: <i>EDMserver.ftp.FTPHostIBM</i> and <i>EDMserver.ftp.FTPHostCommons</i>
isCentralSite	Y - If this is a central system N - If this is a remote system
licenseFile	The name of the file that contains the license information that enables the software to run on a server
localReceiveDir	The directory where incoming transactions are stored. The replacement variables <i>%REMOTELLOCATIONID%</i> and <i>%REMOTECUSTOMERID%</i> are replaced with either the location number of the custom ID field in the location record when creating the directory name. For example, if a location's number is 217, then <i>Incoming %REMOTELLOCATIONID%</i> would become <i>Incoming 217</i> . If a location's custom ID is 000217 then <i>Incoming %REMOTECUSTOMERID%</i> would become <i>Incoming 000217</i> . Replace <i>%THISSITEID%</i> with this site's number from <i>Config.xml</i> . Replace <i>%THISSITENAME%</i> with this site's name from <i>Config.xml</i> . Replace <i>%THISSITECUSTOMID%</i> with this site's custom ID (or ID if null or blank). Replace <i>%CENTRALSITEID%</i> with central site's number from <i>Config.xml</i> . Replace <i>%CENTRALSITENAME%</i> with central site's name from <i>Config.xml</i> . Replace <i>%CENTRALSITECUSTOMID%</i> with central site's custom ID (or ID if null or blank). Replace <i>%REMOTESITEID%</i> with remote site's number. Replace <i>%REMOTESITENAME%</i> with remote site's name. Replace <i>%REMOTESITECUSTOMID%</i> with remote site's custom ID (or ID if null or blank).

localSendDir	<p>The name of the directory where outgoing transaction files are stored. The replacement variables %REMOTELLOCATIONID% and %REMOTECUSTOMERID% are replaced with either the location number of the custom ID field in the location record when creating the directory name. For example, if a location's number is 217, then <i>Outgoing %REMOTELLOCATIONID%</i> would become <i>Outgoing 217</i>. If a location's custom ID is 000217 then <i>Outgoing %REMOTECUSTOMERID%</i> would become <i>Outgoing 000217</i>.</p> <p>Replace %THISSITEID% with this site's number from <i>Config.xml</i>. Replace %THISSITENAME% with this site's name from <i>Config.xml</i>. Replace %THISSITECUSTOMID% with this site's custom ID (or ID if null or blank). Replace %CENTRALSITEID% with central site's number from <i>Config.xml</i>. Replace %CENTRALSITENAME% with central site's name from <i>Config.xml</i>. Replace %CENTRALSITECUSTOMID% with central site's custom ID (or ID if null or blank). Replace %REMOTESITEID% with remote site's number. Replace %REMOTESITENAME% with remote site's name. Replace %REMOTESITECUSTOMID% with remote site's custom ID (or ID if null or blank).</p> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <p>If you use a polling package such as <i>RemoteWare</i>, you can configure it to check this directory when it connects to a remote location to transfer files.</p> </div>
menuFile	The name of the file that contains the options to display on the main menu
nullValue	The text to display in EDM forms for database values that are null. Default is "<NULL>".
onLogin	Perform the specified action every time a user logs into EDM. This can be any EDM expression. An example would be: onLogin = "iff(hasPermission('execute', 'function', 'showSiteManager'), showSiteManager(), 0)"
promptForCommitOnClose	Y - (default) To prompt users to commit open transactions each time they close a form or table N - To not display this prompt
useUpdateButtonForSingleSite	N - (default) Update buttons in EDM forms opened using the "open()" function can optionally be displayed when editing data for a single site using the "useUpdateButtonForSingleSite" config file option.
processUnexpectedFiles	Automatically process unexpected transaction files and query missing transactions from the central server. Default is Y.
remoteReceiveDir	<p>If you have a full time connection between locations, use this setting to specify the directory where the remote location looks for incoming transactions. You can use the replacement variables %REMOTELLOCATIONID% and %REMOTECUSTOMERID%.</p> <p>Replace %THISSITEID% with this site's number from <i>Config.xml</i>. Replace %THISSITENAME% with this site's name from <i>Config.xml</i>. Replace %THISSITECUSTOMID% with this site's custom ID (or ID if null or blank). Replace %CENTRALSITEID% with central site's number from <i>Config.xml</i>. Replace %CENTRALSITENAME% with central site's name from <i>Config.xml</i>. Replace %CENTRALSITECUSTOMID% with central site's custom ID (or ID if null or blank). Replace %REMOTESITEID% with remote site's number. Replace %REMOTESITENAME% with remote site's name. Replace %REMOTESITECUSTOMID% with remote site's custom ID (or ID if null or blank).</p>
remoteSendDir	<p>If you have a full time connection between locations, use this setting to specify the directory where the remote location puts outgoing transactions. You can use the replacement variables %REMOTELLOCATIONID% and %REMOTECUSTOMERID%.</p> <p>Replace %THISSITEID% with this site's number from <i>Config.xml</i>. Replace %THISSITENAME% with this site's name from <i>Config.xml</i>. Replace %THISSITECUSTOMID% with this site's custom ID (or ID if null or blank). Replace %CENTRALSITEID% with central site's number from <i>Config.xml</i>. Replace %CENTRALSITENAME% with central site's name from <i>Config.xml</i>. Replace %CENTRALSITECUSTOMID% with central site's custom ID (or ID if null or blank). Replace %REMOTESITEID% with remote site's number. Replace %REMOTESITENAME% with remote site's name. Replace %REMOTESITECUSTOMID% with remote site's custom ID (or ID if null or blank).</p>

schedule	<p>A task that is scheduled to execute at a specific time.</p> <p>Setting Attributes <i>frequency</i> - ONCE, DAILY, TIMED, or WEEKLY. The default is DAILY. <i>time</i> - The date/time the task should first be executed in local format. The default is now. <i>daysOfWeek</i> - Comma-delimited list of day-of-week numbers, where Sunday=1. The default is every day. <i>task</i> - EDM expression to evaluate</p> <p>Timed tasks execute in intervals respective to the hour when they are first executed. For instance, a task scheduled for <i>06/26/13 12:15 AM</i> will execute at 12:15 AM, and then every 15 minutes thereafter. A task scheduled for <i>06/26/13 12:30 AM</i> will execute every half-hour. A task scheduled for <i>06/26/13 12:01 AM</i> will execute every minute.</p>
schemaFile	The name of the file that contains the database table information, such as columns, keys, and relations
sendStatusTransactions	Y - (default) To send status transactions after received transactions are processed N - To NOT send status transactions
serverPlugin	The class to load as server plugin functions
sessionTimeoutMinutes	The number of minutes of inactivity before the user is automatically logged out of a browser session. 0 to disable this setting.
snapshotFile	The name of the file that contains named lists of tables called "snapshots", which are used for copying, sharing, and deleting site data
systemDataSource	Defines the connection to the system database
systemDataSource.database	The name of the EDM system database For MS Access databases, specify the full path, such as C: <i>Enterprise Data Manager</i> <i>CDM.mdb</i>
systemDataSource.name	The name of the system data source (stored in schema file)
systemDataSource.password	(<i>SQLSERVER_JDBC</i> driver only) The password for the user ID used to access the database
systemDataSource.server	(SQL Server database only) The name of the server where the database resides, such as (<i>local</i>) or <i>ENTERPRISE</i>
systemDataSource.type	<p>The type of database used for the EDM system database</p> <p>Valid Values <i>ACCESS</i> <i>SQLSERVER_JDBC</i> <i>SQLSERVER</i> <i>ORACLE</i></p> <p>Due to its size limitations, central systems should not use <i>ACCESS</i> as the database type. If you use Oracle, you will have to create the database manually before installing the system. To support Unicode characters, use <i>SQLSERVER_JDBC</i> and a Microsoft SQL Server 2000 database.</p>
systemDataSource.user	(<i>SQLSERVER_JDBC</i> driver only) The user ID to use for accessing the database During installation, this user needs permission to create databases, tables and indexes, and Read/Write data. During normal system operations, the user only needs permission to read table schemas and Read/Write data.
systemDataSource.version	The version number of the data source. The default is 1.
thisSiteName	This site's name
thisSiteNumber	This site's number
transactionConversionFile	The name of the file that contains conversion information, which allows the system to support multiple versions of an application at different remote sites
transientFieldFile	The name of the file that identifies the fields that should NOT be updated at remote sites, because the fields are updated by the remote application For example, the <i>OnHand</i> quantity in the inventory item record should not be overwritten when the item name is updated.
useFormColors	Y - Set the background color of EDM forms to the value specified in the EDM form file. N - (default) Set the EDM form background color to white.

useSimpleSiteSelection	Y / N. The default is "N". If set to 'Y', the site selection screen is greatly simplified. The Share and Location groups, the Site Properties and Search controls and the Save Package Defaults button are removed from the screen. Clicking the OK button always updates the package defaults.
version	The version of the <i>Config.xml</i> file set by the system
welcomeMessageExpression	The output of this expression will be displayed in the welcome bar at the top of the EDM client window.

MICROS Specific Entries

Setting	Description
alohaImageDir	Specifies the folder where images may be found.
alohaScriptDir	Specifies the folder where scripts may be found.
alohaIni	Specifies the folder where the INI file may be found.

ALOHA Specific Entries

Setting	Description
microsBitmapsDir	Specifies the folder where bitmap images may be found.
microsIconsDir	Specifies the folder where icon images may be found.
microsOpsDisplayUserFile	Specifies the path to the Micros screen templates file.

IRIS Specific Entries

Setting	Description
irisImageDir	Specifies the folder where images may be found.
irisSoundDir	Specifies the folder where sounds may be found.
removeIrisFindItemButton	Y - Do not display the Find Item button. N - (default) Display the Find Item button.
removeIrisDeleteItemButton	Y - Do not display the Delete Item button. N - (default) Display the Delete Item button.
limitIrisToItemTypes	Y - Limit the IRIS menu editor to only "item" button types in the selection list and drop down. N - (default) Do not limit the button types.
limitIrisPopupOptions	Y - Limit the IRIS menu editor's popup menu options to just alignment and sizing entries. N - (default) Do not limit the popup menu items in the IRIS menu editor.
removeIrisEditButton	Y - Do not display the Edit button. N - (default) Display the Edit button.
removeIrisCommandButton	Y - Do not display the Command button. N - (default) Display the Command button.
snapToGrid	Y - Snap buttons to grid intersections while dragging. N - (default) Do not snap buttons to grid intersections while dragging.
showGrid	Y - Show grid in menu editor. N - (default) Do not show grid in menu editor.

Master Data Management Specific Entries

Setting	Description
---------	-------------

scheduleVersionSiteAssignmentsPanel	Specifies the name of the panel to use for selecting sites in the schedule assignments editor.
-------------------------------------	--

Nucleus Specific Entries

Setting	Description
nucleusPromptForPackage	Y - Prompt for packages. N - (default) Use a system generated package.

XPRESS Specific Entries

EDM XPRESS Store Server

Setting	Description	Example
xpressLocalBrandedImageDir	The location of the foreground image files on the EDM Store Server.	"C:\TNG2.0\images\XPR\en_US\"
xpressRemoteBrandedImageDir	The location of the foreground image files on the EDM Cloud Server.	"company/customerid/images/XPR/en_US/"
xpressLocalConvertImageDir	The location of the converted image files on the EDM Store Server.	"company/customerid/images/ConvertImage/"
xpressRemoteConvertImageDir	The location of the converted image files on the EDM Cloud Server.	"C:\TNG2.0\images\ConvertImage\"
xpressGenericImageDir	The location of the background and popup image files on the EDM Store Server.	"C:\TNG2.0\images\Generic\en_US\"
xpressLocalImportImageDir	The location of the import image files on the EDM Store Server.	"C:\TNG2.0\images\ImportImage\"
xpressMenuEditorImageDir	The location of the menu editor UI image files on the EDM Store Server.	"C:\TNG2.0\images\MenuEditor\"

customerid

The customerid field is replaced with a unique customer ID by the configuration utility that the user runs in the store. It must match the customerid value in the Cloud server.

EDM XPRESS Cloud Server

Setting	Description	Example
xpressLocalBrandedImageDir	The location of the foreground image files on the EDM Cloud Server.	"company/customerid/images/XPR/en_US/"
xpressRemoteBrandedImageDir	The location of the foreground image files on the EDM Store Server.	"C:\TNG2.0\images\XPR\en_US\"
xpressLocalConvertImageDir	The location of the converted image files on the EDM Cloud Server.	"company/customerid/images/ConvertImage/"
xpressGenericImageDir	The location of the background and popup image files on the EDM Store Server.	"company/customerid/images/Generic/en_US/"
xpressMenuEditorImageDir	The location of the menu editor UI image files on the EDM Store Server.	"company/customerid/images/MenuEditor/"

customerid

The customerid field is replaced with a unique customer ID when we set up a new company. It must match the customerid value in the store.

See Also

[Setting Property Values for LDAP](#)
[Getting Property Values from the Windows Registry](#)
[Adding an Additional Company](#)
[Database Purge Process](#)

Setting Property Values for LDAP

EDM can be configured to use security values from an LDAP using the LDAP Configuration Tool. When the attribute values are saved, you are automatically logged out and required to log in using the LDAP information. Thereafter, all permissions and roles come from the LDAP. For security, the LDAP information is encrypted.

Attribute	Description
LDAPIP	The IP of the LDAP machine
LDAPPort	A port of the LDAP machine that is available for Java binding
LDAPUser	A user able to scan the context for user, permission and role information
LDAPPassword	The password for the LDAPUser
LDAPContext	The initial context for the EDM information. o=EDM by default. New users may have this set up under the <i>Setup</i> menu. Upgrading users can follow these steps to add this menu item to their menus: <ol style="list-style-type: none">1. Select the menu.2. Right-click a menu item.3. Select <i>New Item Above</i> or <i>New Item Below</i>.4. Fill out the form with a name, such as <i>LDAP Configuration</i> and the command <i>showLdapPropertyForm()</i>.

For additional information, see [Using an LDAP with EDM](#).

Getting Property Values from the Windows Registry

To read the *systemDataSource* and *dataSource* attribute values for *Config.xml* from the Windows Registry, set the attribute value to a special string, such as:

```
?Registry_Key=Software  
Application;Value=ConnectionString;Substring=DB
```

When the system sees an attribute value that starts with *?Registry_Key=*, it reads the attribute value from the Registry key that follows the equal sign (=). The attribute value string must start with *?Registry_Key=* followed by the registry key name and a semicolon (;). The value must contain *Value=* followed by the Registry value to read from the specified key followed by a semicolon.

Optionally, the value can contain *Substring=* followed by the particular substring to extract from the Registry value when it is in the format *key1=value1;key2=value2;...* such as an ODBC connection string.

The sample attribute value above reads the *ConnectionString* value from the *Software/Application* key to get an ODBC connection string, and then extracts the *DB* substring from the connection string to get the name of the database to use as the attribute value.

Adding an Additional Company

You can set up multiple companies to help organize the data at headquarters. For example, if you have a *Pizza* concept and a *Coffee Shop* concept, you can set up two companies, so that each concept has its own central database with different tables, forms, and menus.

Getting Started

1. Open *Config.xml* with a text editor.
2. Copy the company section (from *<company>* to *</company>*) to the clipboard.
3. Paste the copied company lines at the end of the file between the *</company>* line and the *</config>* line.

Configuring the New Company

Perform the following steps in the new company section of *Config.xml*.

1. Change the *centralSiteName* attribute to a new company name.
2. Change the *thisSiteName* attribute to the same new company name.
3. Change the *schemaFile* attribute to a new name (e.g. *schema_newcompany.xml*).

4. If the new company will use a different main menu, change the *menuFile* attribute to the new file name.
5. If the new company will use different table lists, change the *snapshotFile* attribute to the new file name.
6. If the new company will use different transient fields, change the *transientFieldFile* attribute to the new file name.
7. If the new company will use different data conversions, change the *transactionConversionFile* attribute to the new file name.
8. Change the *localSendDir* attribute to a new folder (e.g. *OutgoingNewCompany%REMOTELOCATIONID%*).
9. Change the *localReceiveDir* attribute to a new folder (e.g. *IncomingNewCompany%REMOTELOCATIONID%*).
10. Change the *name* attribute of the *systemDataSource* attribute to a new data source name (e.g. *EdmWebHqNewCompany*).
11. Change the *database* attribute of the *systemDataSource* attribute to a new data source name (e.g. *EdmWebHqNewCompany*).

Final Steps

1. Close and save the config.xml file.
2. Restart the Apache Tomcat service.

When you log in, you are prompted to select the company to view. To switch to another company, log out and back in again -OR- click *Switch Company*, if available.

Snapshots

Snapshots are named lists of tables used for various tasks, such as requesting a table refresh for a list of tables. Snapshots are saved to an XML file in the same folder as the [Config.xml](#) file. On remote systems, this is usually *C:\EDMWeb*. On central systems, this is usually *C:\jakarta-tomcat-5.5.7\webapps\EDMWEB-INF\classes*.

The *name* attribute of the snapshot element specifies the name of the snapshot, which is displayed when the user is prompted to select a snapshot.

The *name* attribute of the table element specifies the name of a table in the snapshot. A table can appear in any number of snapshots. Duplicate table names within a snapshot are ignored.

Sample Snapshot File

The following sample file includes two snapshots: *Item Tables* and *Price Table*.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE snapshots>

<snapshots version="1">

  <snapshot name="Item Tables" >
    <table name="IRIS_dbo_tbl_ItemMaster" />
    <table name="IRIS_dbo_tbl_ItemModifiers" />
    <table name="IRIS_dbo_tbl_ItemAttachments" />
    <table name="IRIS_dbo_tbl_ItemCatLinks" />
    <table name="IRIS_dbo_tbl_ItemCategories" />
    <table name="IRIS_dbo_tbl_ItemFuturePricing" />
    <table name="IRIS_dbo_tbl_ItemFuturePricing" />
  </snapshot>

  <snapshot name="Price Table" >
    <table name="IRIS_dbo_tbl_ItemFuturePricing" />
  </snapshot>

</snapshots>
```

See Also

[Adding a Snapshot](#)
[Changing a Snapshot](#)

Adding a Snapshot

1. Use a text editor to edit the *snapshot[app_version].xml* file.

2. Immediately before the final `</snapshots>` element, add a new snapshot element with the desired tables as follows:

```
<snapshot name="My New Snapshot" >
    <table name="first_Table_Name" />
    <table name="second_Table_Name" />
</snapshot>
```

3. Close the text editor and save the modified file. Restarting the Web server is not necessary to implement your changes.

Changing a Snapshot

1. Use a text editor to edit the `snapshot{app_version}.xml` file.
2. Find the snapshot to change using the `name` attributes.
3. Change the snapshot by editing the table name, or adding/removing tables.
4. Close the text editor and save the modified file. Restarting the Web server is not necessary to implement your changes.

Main Menu

File name

The menu file is usually named `menu{app_version}.xml`, where `app` is the name of the associated application and `version` is the version number of the application.

For example, in a system set up to manage IRIS - POS v4.0, the menu file is called `menu_iris_4.0.xml`.

File Location

The menu file is stored in the same folder as the `Config.xml` file. On remote systems, this is usually `C:\EDMWeb`. On central systems, this is usually `C:\jakarta-tomcat-5.5.7\webapps\EDMWEB-INF\classes`.

Description

The top-level menu and each sub-menu have a `name` attribute, which is displayed in the left window of the main menu.

Each menu item element has a `name` and an `action`. The `name` is displayed in the right window of the menu. The `action` is what the system does when the user selects the item on the menu.

Users can edit this file to add/edit/delete menus and items. All possible menu items are contained in this file. Security settings are used to limit the menu options a user may access.

See Also

[Editing the Main Menu](#)

Editing the Main Menu

You can edit the [menu file](#) directly -OR- to edit the menus and items from the EDM Main Menu, right-click the menu or item, and then select *Edit* from the menu that appears.

Menu Editing Options

Option	Select it to
New Menu Above	Insert a new menu above the selected menu
New Menu Below	Insert a new menu below the selected menu
New Submenu	Insert a new submenu for the selected menu

New Menu Item	Insert a new menu item as the last child of the selected menu
Edit Menu	Edit the menu name
Cut	Cut the menu to the clipboard
Copy	Copy the menu to the clipboard
Paste Above	Paste the menu from the clipboard above the selected menu
Paste Below	Paste the menu from the clipboard below the selected menu
Paste Child	Paste the menu from the clipboard as a child of the selected menu
Delete Menu	Delete the selected menu

Menu Item Editing Options

Option	Select it to
New Item Above	Insert a new menu item above the selected item
New Item Below	Insert a new menu item below the selected item
Edit Item	Edit the item name and action
Cut	Cut the item to the clipboard
Copy	Copy the item to the clipboard
Paste Above	Paste the item from the clipboard above the selected item
Paste Below	Paste the item from the clipboard below the selected item
Delete Menu	Delete the selected item

Restarting the System

To implement your changes, you need to restart the system after editing the menu file. On standalone systems that are not running on a Web server, the next time the system is started it will read the modified menu file. On central servers running the Apache Tomcat Web server, follow these steps to restart the Apache Tomcat service:

1. Open the *Windows Control Panel*.
2. Open *Administrative Tools - Services*.
3. Right-click the service, and then select *Start* from the menu that appears.

Other Configuration Files

The other configuration files referenced in [Config.xml](#) are described below.

Note

These files are typically not edited directly by the user.

appFile

This is an application file that contains all the forms and queries used in the system. This file is updated when forms and queries are modified using the designers.

schemaFile

This file contains the table structures, indexes and table relations for both the system tables and the application tables managed by the system.

transientFieldFile

This file contains a list of fields that are only updated by the application at the remote sites and NOT by the central system.

For example, the application at the remote sites stores the *OnHand* quantity of an inventory item in the inventory item record. When users update the inventory item record at central, we don't want the *OnHand* quantity value overwritten at the remote site. To accomplish this, we add the *OnHand* field to the transient field file in the *InventoryMaster* table.

```
<?xml version="1.0"?>
<!DOCTYPE transients>

<transients version="1">
<table name="IRIS_dbo_InventoryMaster">
<col name="OnHand" />
<col name="LastDateWasted" />
</table>
<table name="IRIS_dbo_AuthorizedEnvironment">
<col name="Replicated" includeInInsert="N" />
<col name="RepGUID" includeInInsert="N" />
</table>
</transients>
```

Field Attribute	Description
includeInInsert	Y - To include the transient field in Insert and Table Refresh transactions that are sent to remote sites N - To NOT include the transient field in Insert and Table Refresh transactions that are sent to remote sites

The following describes how the transient fields are handled when sent to the remote site or the central site.

Insert transaction to remote or central sites	Transient fields are included in the transaction and inserted in the table unless the <i>includeInInsert</i> flag is set to N at the sending site.
Update transactions sent to remote or central sites	Transient fields are not included in the transaction and are not updated.
Delete transactions sent to remote or central sites	Transient fields are not included in the transaction and are not deleted.
Refresh transaction sent to remote site	Transient fields are included in the field list, the transient field list and the data values unless the <i>includeInInsert</i> flag is set to N at the central site. <i>Insert</i> transactions generated by processing the refresh will insert the available transient values in the table at the remote site. <i>Update</i> transactions generated by processing the refresh will not use transient field values when searching for the row to delete, and will not update transient fields. <i>Delete</i> transactions generated by processing the refresh will not use transient field values when searching for the row to delete.
Refresh transaction sent to central site	Transient fields are ignored unless the <i>includeInInsert</i> flag is set to N at the remote site, in which case, the transient fields are left out of the field list and the field values. <i>Insert/Update/Delete</i> transactions generated by processing the refresh will be processed as if there were no transient fields. However, transient fields with the <i>includeInInsert</i> flag set to N at the remote site will be ignored.

transactionConversionFile

This file contains information the system uses to convert transactions between one version of the managed application and another.

For example, if some remote sites are running IRIS - POS v3.8 and other sites are running IRIS - POS v4.0, the conversion file ensures transactions apply to the right column names and types for both versions.

licenseFile

This file contains license information for a limited or unlimited number of remote sites. Only central systems need a license file.

Upgrading EDM or Your Application

When a new version of EDM is available, you may want to install it on your central system or on both your central and remote systems. Changes in the new version frequently only affect the central system, so it is not necessary to upgrade all the remote installations. The central system version should always be equal to or later than any version running at a remote site.

When you install a new version of your application, EDM can support both the old and new versions of the application during the rollout period.

See Also

Upgrading Your Application

When you install a new version of your application, EDM can support both the old and new versions of the application at different sites during the rollout period.

Set up a lab system with the new version of the application

Step	Description
1	To validate that your central system can support remote sites running different versions of the application during rollout, you will need two lab sites. One lab site will run the old version of the application. The second lab site will run the new version. Assign a unique site number to each lab site. To validate the upgrade process, use the lab site that is running the new version of the application.
2	Configure the new lab site for Web transfers with the central system (see EDM Installation and Setup).

- 3 If the application version number is not correct in the lab's `Config.xml` file, update the application data source version number. For example, if you installed the EDM remote with IRIS 3.8, but you used the `setup_remote_IRIS_3.7.14PR10.bat` file to run the installation, then change the `version` property in the `dataSource` element in the `Config.xml` file from:

```
<dataSource
name = "IRIS"
version = "3.7.4PR10"
type = "SQLSERVER_JDBC"
database = "IRIS"
server = "localhost"
user = "sa"
password = ""/>
```

TO:

```
<dataSource
name = "IRIS"
version = "3.7.8"
type = "SQLSERVER_JDBC"
database = "IRIS"
server = "localhost"
user = "sa"
password = ""/>
```

- 4 Run `go.bat`. EDM will reload the application database structures with the correct version number.

Load the data structures for the new version into the central system

Step	Description
1	Backup the central system database (named <code>EDMWebHQ</code> by default).
2	Backup the central system configuration files (in the <code>C:\jakarta-tomcat-5.5.7\webapps\EDM</code> folder by default).

3	On the central EDM system, send a <i>Request Schema</i> transaction to the lab site running the new version of the application.
4	On the remote EDM lab system, double-click <i>transfer_web.bat</i> (in the <i>C:\EDMWeb</i> folder by default).
5	On the central EDM system, click <i>Transactions – Process All Transactions</i> on the central site to process the received table list.

Convert the central system database to support the new application data structures

Step	Description
1	On the central EDM system, click the option on the <i>Configuration</i> menu <i>Configure Application Versions</i> to configure the new version.
2	Use the <i>Move Up</i> and <i>Move Down</i> buttons to set the sequence of the application versions, so that the oldest version of each data source is at the top and the newest version is at the bottom.
3	Click <i>Confirm Version Sequence</i> . The system will save the version sequence and automatically generate the conversions between each version.
4	Click the <i>Review Conversions</i> button to check if any tables or columns that were dropped and added were actually just renamed. To change a drop/add conversion to a rename conversion: Select the conversion, and then click the <i>Change to Rename</i> button. To change a rename conversion to a drop/add conversion: Select the conversion, and then click the <i>Change to Delete</i> button.
5	To convert the tables for a data source in the central database to the new version, select the new version, and then click the <i>Convert Central Database</i> button.

Load the new lab site data into the central system

Step	Description
1	On the central EDM system, send a <i>Request Table Refresh</i> transaction to the new lab site.
2	On the remote EDM lab system, double-click <i>transfer_web.bat</i> (in the <i>C:\EDMWeb</i> folder by default).
3	On the central EDM system, click <i>Transactions – Process All Transactions</i> on the central site to process the received data.
	At this point, the central system should be supporting two lab sites running different versions of your application. To continue updating all remaining sites to the newest version, proceed to steps 5 and 6 below.

Update the version number in the remote system configuration file as sites are upgraded

As your locations are upgraded to run the new version of the application, make sure *Config.xml* (in the *C:\EDMWeb* folder by default) is updated to the correct data source version number (see step 2).

Update the version number in the central system as sites are upgraded

Perform the following steps after the new application is installed at each remote site.

Step	Description
------	-------------

1	<p>On the central EDM system, update the record in the <i>CDMRemoteSource</i> table (<i>CDMRDSRC</i>) to have the correct version number. For example, if IRIS 4.0 is installed at site 1234, update the record in <i>CDMRemoteSources</i> from:</p> <pre>SITE NAME VERSION 1234 IRIS 3.8PR10</pre> <p>To:</p> <pre>SITE NAME VERSION 1234 IRIS 4.0</pre> <p>-OR- add a new record if no record for the IRIS data source at the site is found in the table.</p>
2	<p>If the data in the site was changed as a result of upgrading the application and you want to load the changed data into the central database, use <i>Transactions – Request Table Refresh</i> to ask the remote site to send the upgraded data.</p> <div style="border: 1px solid #f0e68c; padding: 10px; margin-top: 10px;"> <p>Note Requesting table data only works if the requested data is NOT shared, so be sure to unshare any shared site table before requesting a table refresh.</p> </div>

Upgrading EDM from JAR to an EXE File Structure

The following describes an EDM upgrade from Apache Tomcat 6.0.18 (or jakarta-tomcat-5.5.7) to the newer EDM Installations.

Stop the Apache Tomcat Service

1. Open the Windows Control Panel.
2. Open *Administrative Tools*.
3. Open *Services*.
4. Right-click the *Apache Tomcat* service, and then select *Stop* from the menu that appears.

Run the EDM Server Setup

Run *EDM-Server-Setup.exe* as Administrator selecting the default prompts throughout the install.

Notes

- Select IRIS 4.0 (even if IRIS 4.0 is not installed yet).
- System Data Source Info Type=SQLSERVER (remove JDBC)
- Active MQ = No, do not use
- Enter *Install Path C:\EDMServer* or *D:\EDMServer*

Copy Files

1. Copy any files from the *C:\jakarta-tomcat-5.5.7\webapps\EDM* folder to *C:\EDMServer\webapps\EDM* that may be needed (e.g. files pushed from EDM to the locations, the image folder for register buttons).

Note

Do NOT overwrite the *EDM.jnlp* and *loginPage.jsp* files in the new *webapps/EDM* directory.

2. Copy the following files from the *C:\apache-tomcat-6.9.18\webapps\EDM\web-inf\classes* folder to the *C:\EDMServer\webapps\EDM\web-inf\classes* folder.
 - a. Image folder (or just the files that aren't present in the new image folder)
 - b. Incoming folders
 - c. IRIS folder
 - d. Outgoing folder
 - e. QryExports folder
 - f. Work folder
 - g. Config.xml (rename in current)
 - h. Convert_iris-xxxx.xml
 - i. Mail.properties
 - j. Menu_iris-xxxx.xml

- k. Policy.xml (rename in current)
- l. All related *Schema_xxx.xml* files
- m. Snapshots_iris_xxxx.xml
- n. Transients_iris_xxxx.xml

Follow these steps if you want to keep your current forms.

1. Copy the *app_iris.xml* file to the *C:\EDMServer\webapps\EDM\web-inf\classes* folder.
2. Go to the *C:\EDMServer\webapps\EDM\web-inf\classes* folder. Rename the *Installed Forms* folder to *FormsRel*.
3. Go to the *C:\EDMServer\webapps\EDM\web-inf\classes* folder. Rename the *Installed Queries* folder to *QueriesRel*.
4. Add *appfile=app_iris.xml* to the *Config.xml* file.

Config.xml

Ensure the paths in the *Config.xml* file are correct (*C:\EDMServer*.**).

Stop the Apache Tomcat Service

1. Open the Windows Control Panel.
2. Open *Administrative Tools*.
3. Open *Services*.
4. Right-click the *Apache Tomcat* service, and then select *Stop* from the menu that appears.

Configure the EDM Central Tomcat Service

1. Open the Windows Control Panel.
2. Open *Administrative Tools*.
3. Open *Services*.
4. Verify the EDM Central Tomcat service is present.
5. If the service is started, right-click the *Apache Tomcat* service, and then select *Stop* from the menu that appears.
6. Ensure the service is set to automatically start.

If the service was not installed, then follow these steps.

1. Open a command prompt as Administrator.
2. Go to the *C:\EDMServer\bin* folder.
3. Type *Service.bat install*.

Start the EDM Central Tomcat Service

1. Open the Windows Control Panel.
2. Open *Administrative Tools*.
3. Open *Services*.
4. Right-click the *EDM Central Tomcat Service* service, and then select *Start* from the menu that appears.

Note

If you had any login credentials for the existing service, you'll need to add those credentials to this service as well.

5. Reference *C:\EDMServer\webapps\EDM\log.txt* to verify the installation was successful. If you receive the HOST info for a new LICENSE, then the installation was successful.
6. Request a new license file from XPIENT.
7. Once the license file is provided, save it in the *C:\EDMServer\webapps\EDM\web-inf\classes* folder.
8. Restart the *EDM Central Tomcat Service*.
9. Reference *C:\EDMServer\webapps\EDM\log.txt*. Look for *Text of Server Ready!*. This entry indicates the license is valid.
10. Open a browser and try to connect to EDM.

Upgrading from EDM 3.1 or Later

Upgrading a Central System

1. Determine the folder where EDM is installed on the central web server. For example, your *[EDMfolder]* may be *C:\jakarta-tomcat-5.5.7\webapps\EDM*.
2. Install the new *EDMclient.jar* in the *[EDMfolder]*.
3. Install the new *system.jar* in *[EDMfolder]\WEB-INF\lib*.
4. Restart the Apache Tomcat service on the Web server.

5. Open the *log.txt* file in the [EDMfolder] to verify that the first line contains the new version number and the last line contains "Server ready!". See example below.

Successful Startup Log.txt Example

The timestamp and code locations have been removed.

```
INFO Started web version Version 3.7.1 at Thu Jan 12 11:45:43 MST 2013
INFO Loading configuration file: 'C:\jakarta-tomcat-5.5.7\webapps\EDM\web-inf\classes\config.xml'
INFO Loading configuration for company 'HQ'
INFO Loading conversion file: C:\jakarta-tomcat-5.5.7\webapps\EDM\web-inf\classes\convert_iris_3.7.4.xml
INFO Loading security configuration.
INFO Initializing company 'HQ'
INFO System database not created. Error code: 1801: [Microsoft][SQLServer 2000 Driver for JDBC][SQLServer]Database EDMWebHQ already exists.
INFO Checking database version
INFO Purging audit trail to 14 days
INFO Loading sites...
INFO Loading site groups...
INFO Loading remote tables...
INFO Checking transaction version...
INFO Server ready!
```

Upgrading a Remote System

1. Determine the folder where EDM is installed on the remote system. For example, your [EDMfolder] may be C:/EDMWeb.
2. Install the new *system.jar* in [EDMfolder]/lib.
3. Start EDM by running *go.bat* in the [EDMfolder].
4. Verify that the new version number is displayed on the splash screen.

Upgrading from EDM 3.0.x

Upgrading a Central System

1. Determine the folder where EDM is installed on the central web server. For example, your [EDMfolder] may be C:/jakarta-tomcat-5.5.7/webapps/EDM.
2. Install the new *EDMclient.jar* in the [EDMfolder].
3. Install the new *system.jar* in [EDMfolder]/WEB-INF/lib.
4. Install the new *app_application.xml* in [EDMfolder]/WEB-INF/classes.
5. Install the new *convert_application_version.xml* in [EDMfolder]/WEB-INF/classes.
6. Restart the Apache Tomcat service on the Web server.
7. Open the *log.txt* file in the [EDMfolder].
8. Verify that the first line contains the new version number and the last line contains "Server ready!".

Upgrading a Remote System

1. Determine the folder where EDM is installed on the remote system. For example, your [EDMfolder] may be C:/EDMWeb.
2. Install the new *system.jar* in EDMfolder/lib.
3. Install the new *app_application.xml* in [EDMfolder].
4. Install the new *convert_application_version.xml* in [EDMfolder].
5. Start EDM by running *go.bat* in the [EDMfolder].
6. Verify that the new version number is displayed on the splash screen.

Upgrading from EDM 2.9.x or Earlier

Upgrading a Central System

1. To install the central system and initialize the central database, follow the instructions for [Installing Central System](#).
2. To delete the remote location data that is loaded during installation, select the *Sites* module, and then select *Delete Location Data* from the *Sites* menu.
3. To delete the remote location record for the lab system, select *Edit Locations and Groups* from the *Sites* menu.
4. Run the import script obtained from your vendor to import the data from the previous EDM central database to the new database.

Upgrading a Remote System

Note

Make sure all transactions both to and from the remote location are transferred and processed before upgrading the remote system.

1. To install EDM on the remote system, follow the instructions for [Installing Remote System](#).
2. Uninstall the EDM software on the remote system by deleting the EDM folder and dropping the EDM database.
3. On the central system:
 - a. Select the *Setup* module.
 - b. Expand the *Applications* menu.
 - c. Expand the *Tables* menu.
 - d. Open *CDMLocations* to edit the remote location's record.
 - e. Set the *SENTTFN* and *RCVDTFN* fields to 0.

Frequently Asked Questions

How does EDM integrate with our POS and Back Office system?

EDM can directly access the application database at a site using JDBC or ODBC drivers. It reads and writes to the database directly in response to user editing via EDM forms and transactions received from headquarters, such as a price change. If a user makes a change at the store using the EDM editors, the change is sent to headquarters in a transaction file. To track changes to tables that are not edited by users (such as transaction tables), EDM can take a snapshot of the data at regular intervals and send the changes it detects to headquarters in a transaction file.

What kind of database is maintained by EDM?

EDM maintains a very small database at the store for system data and interacts with the POS and Back Office application databases directly. The system database can be Microsoft Access, Microsoft SQL Server, or Oracle (other ODBC databases may be supported). At headquarters, EDM maintains system data as well as data collected from the stores in a central database, which does not have to be the same as the store database. We recommend using Microsoft SQL Server since the size of the database is many times larger than a single store database.

How are updates delivered to the stores?

When you use an EDM form to change data in the central database, such as item prices, the old and new price record is written in a numbered XML file, which is placed in an *Outgoing* directory on the central system. The user supplies the communication software, such as *RemoteWare*, to physically move the file to the *Incoming* directory at the store. The communication process can be scheduled to run daily or more often if the data changes are more time sensitive.

Does EDM include communications software?

EDM supports transmitting transaction files and other files to an EDM Web server, an FTP server or over the local area network. Users can also use their existing communication systems, such as *RemoteWare* to transfer transaction files.

How does EDM collect data from the stores?

Data changes made at the store using EDM forms are sent to headquarters in transaction files. Other changes can be sent to headquarters by either sending entire tables, a selected group of records from a table (such as the last 7 days of transactions), or only updated data by comparing a previous snapshot of the data with a current snapshot. To ensure data is not lost if communications fails, all data is sent in the form of numbered XML transaction files that are processed at headquarters when they are received.

What are the system requirements at remote locations?

EDM is a Java program, so it is compatible with any OS that supports Java. An installation at a remote location includes a Java runtime environment. The only other necessary software is the ODBC driver to access the store database. The system takes about 20MB of disk space and needs about 40MB of memory when it is running to avoid memory-to-disk swaps, which slow the system down. Total memory requirements depend on other software running on the system. EDM only runs when transactions are processed or when the user in the store wants to use an EDM form to make data changes. It does not use any memory when it is not running.

What are the system requirements at headquarters?

At headquarters, the server needs a database server large enough to handle the data from all the stores, the EDM software (which includes the

Java runtime environment) and ODBC drivers to access the database server. We recommend using Microsoft SQL Server on a system with fast disk drives, 10MB of free disk space per store and 4GB of memory. Additional memory on the database server is the best method for improving performance since most database servers cache results in memory.

Can I still access the remote locations with PCAnywhere when EDM is running?

EDM can be run from inside a *PCAnywhere* session just like any other program. EDM will not interfere with *PCAnywhere* or any other communication software because it does not contain any communication software itself.

Does EDM have to be loaded or running all the time at remote locations?

At remote locations, EDM typically only runs once a day during end-of-day processing when it processes incoming transactions and sends updates to headquarters. If the users at the remote location are allowed to make data changes using the EDM forms, then they will have to run the program to use the forms.

Can I use an event manager or scheduler to run EDM from a command line?

Typically, EDM transaction processing is run from a command line as part of end-of-day. It could also be started from a scheduler or event manager, or by the communications program itself. All EDM functions are available from the command line. See *C:\EDMWeb\process.bat* or *C:\EDMWeb\transfer_web.bat* for examples.

How do I handle the error 'Violation of PRIMARY KEY constraint 'CDMAudit_Primary'?'

Whenever there is a primary key error on *CDMAudit*, it means that an incoming transaction file includes a transaction with a transaction ID that matches one already in the audit trail. Usually, this is due to the transaction file being sent twice or when a backup of the central database is restored it thinks the next transaction number is one that has already been used.

To resolve this problem, delete the transaction file that won't process and increment the received transaction file number in *CDMLocation.RCVDT FN* at the store. Then send a table refresh from central to the store to synchronize the data.

How do I resolve "Error locking 'audit trail, already locked" when committing transactions?

To avoid committing transactions twice, the audit trail is locked when more than one user is committing transactions. If the Apache Tomcat service is restarted while a user is committing transactions, the audit trail lock record may not be properly removed the *CDMAudit* table.

To resolve this situation:

1. If another user is currently committing transactions, wait until they are finished. If no other users are committing transactions, ask all users to wait until you have unlocked the audit trail.
2. Select the *Setup* module.
3. Expand the *Application* menu.
4. Expand the *Tables* menu.
5. Open *CDMAudit*.
6. Manually unlock the audit trail by deleting the row with the -1 in the *ID* column.

Exercise caution when manually deleting the lock record, because you risk committing transactions twice if other users are currently committing transactions.

How do I resolve "Error locking 'xxx', already locked"?

This message appears when the system has determined that more than one user is attempting to run the same process at the same time.

To resolve this situation:

1. If another user is currently running the process, wait until they are finished. If no other users are running the process, ask all users to wait until you have unlocked the process.
2. Select the *Setup* module.
3. Expand the *Application* menu.
4. Expand the *Tables* menu.
5. Open *CDMValue*.
6. Manually unlock the process by deleting the row with a negative number in the *VALUEID* column, and the process name in the *COMMENTS* column.

Exercise caution when manually unlocking processes, because you risk duplicate processing errors if other users are currently running the process.

How do I resolve "[SQLServer]Violation of PRIMARY KEY constraint dboCDMAudit_Primary. Cannot insert duplicate key in object CDMAudit"?

This error is generated when the *Last used transaction ID* value in the *CDMValue* table is incorrect. A database administrator can resolve this situation by taking the following steps.

First, determine the highest used transaction ID number:

1. Select the *Setup* module.
2. Expand the *Application* menu.
3. Right-click *Queries*, and select *New* from the menu that appears.
4. Double-click *CDMAudit+*, and click *Close*.
5. Double-click *FromLoc* and *Id* in the list of fields in the top-half of the window.
6. Type a zero (0) in the condition row under the *FromLoc* column.
7. Select the sigma tool (sideways M) to convert it to a *Totals Query*.
8. Select the Totals row under the *Id* column. Open the list and select *Maximum*.
9. Select the *Display As Spreadsheet* tool (first tool on the toolbar).
10. Note the value in the *Id* column.
11. Close the query and save it as *Highest Transaction Id*.

Second, update the last used transaction number:

1. Right-click *Queries* (beneath *Application Objects* on the EDM menu), and select *New* from the menu that appears.
2. Double-click *CDMValue+* and click *Close*.
3. Double-click *VALUEID* and *VALUE* in the list of fields in the top-half of the window.
4. Type a one (1) in the condition row under the *FromLoc* column.
5. Select *Query - Update* from the window menu, and click *OK* on the *Query Attributes* dialog to convert it to an *Update Query*.
6. In the *Update To* row under the *VALUE* field, type the highest transaction ID + 100. For example, if the highest transaction ID is 891, type 991.
7. Click the *Run* tool (exclamation point).
8. Close the query without saving it.

How do I resolve a remote system that is stuck on "Contacting Server Please Wait"?

Usually this behavior is caused by the system failing to connect to the database server. If you are using the default installation and connecting to a SQL Server database, this behavior could have several causes.

Cause	Resolution
SQL Server is not installed	Install the database server
SQL Server is set up to use Windows authentication instead of SQL Server authentication	Modify the SQL Server properties to allow Windows and SQL Server authentication as required by the JDBC driver
The password for the SQL Server sa user is not blank	Either update the Config.xml file with the correct password for the database server -OR- delete the password in SQL Server for the sa user and leave it blank

How do I resolve the Tomcat startup error "javajni.c not found" on Windows 64 server?

Copy *MSVCR71.dll* in the *C:\Windows\SysWOW64* folder from another Win64 system to *C:\Windows\SysWOW64* on the server running Tomcat.

How do I resolve the Internet Explorer security error concerning the Java plugin?

1. Open the *Windows Control Panel*.
2. Select *Add or Remove Programs*.
3. Select *Add or Remove Windows Components*.
4. Select *Internet Explorer Enhanced Security Configuration* from the list of Windows components.
5. Click *Details*.
6. Remove the checkmark from *For administrator groups* -OR- remove the checkmark from *For all user groups*, if appropriate.
7. Click *OK*.

How do I resolve Bulk Insert errors when copying records in a form?

The *Bulk Insert* capability of EDM uses a SQL Server script that writes multiple records to a file, and then inserts all the records with one statement. On some systems, the default file location is not compatible, so the administrator needs to specify the directory on the SQL Server PC

where the files should be written.

Add a line to the `Config.xml` file that is similar to the following with a valid path for SQL Server:

```
bulkInsertFilePath="C:\temp\"
```

Glossary

Action Queries	Queries used to add/edit/delete records in a table or create a new table
Application	A group of tables, queries and forms that work together to accomplish a task
Audit Trail	Changes to the database are recorded here and are available for review in the Audit report
Avg	A query design total function that reports the average of the column values in all the included rows in the column. For example, if you query the <i>Salary</i> column in the <i>Employee</i> table using the <i>Avg</i> total function, the query result is one row with a <i>Salary</i> column that contains the average salary for all employees.
Bitmap	A form design control tool that creates a bitmap control used to display a picture
Blank Form	A form created with a data source, but without any fields
Bold	Emboldens the text of the selected controls
Bound Column	The number of the column that is saved to the database when a row is selected Example The <i>Row</i> source of your list is the <i>Customer</i> table. The first three columns of the <i>Customer</i> table are: <i>Id</i> , <i>LastName</i> , and <i>FirstName</i> . To save the <i>Id</i> of the selected customer to the database, set the <i>Bound</i> column attribute to 1. To save the <i>LastName</i> to the database, set the <i>Bound</i> column attribute to 2. The bound column does not have to be displayed in the list.
Bound Control	A form design control used to display and edit data from a field in a table or query
Box	A form design control tool that draws a box on the form
Button	A form design control tool that creates a button that when pressed, evaluates the expression defined for the <i>On Push</i> attribute of the control
Calculated Control	A form design read-only control that displays the results of a calculation or function call. A calculated control does not update any fields in the database. Use calculated controls to display read-only information, such as the total of <i>Quantity * Price</i> when entering order items.
Case Sensitive	Check this box if you want the <i>Find</i> tool to only find text with the same upper- and lowercase letters that you typed in the <i>Search</i> field
CDMKeyFields	An invisible field added to a central form that contains a list of field names used as key fields
CDMLOCID	The first field of every record in the central database tables. This field specifies the location ID of the remote location associated with the data.
CDMRemoteTables	A table that contains records identifying tables that contain data that is shared by more than one remote location
CDMSnapshot	A function to create a record of tables to include in a snapshot

Center	Centers the text of selected controls
Central Location	A location with one or more tables that contains data for multiple remote locations
ChartToDate	A condition function that converts a date string to a date value, which can be compared to the value in a date field
Check Box	Creates a check box that the user can check and uncheck
Combo Box	A form design control tool that creates a combo box, which is an entry field combined with a listbox that allows the user to either type a value or select a value from the list
Compare Snapshot	A function designed to compare a previous snapshot of database tables with a current snapshot to determine what changes were made
Condition Expression	A user-defined expression (or criterion) a query conforms to when executed. Condition expressions are similar to normal expressions except they do not include the first operand. Instead, the column value is considered the first operand in the expression.
Constants	Values that are used in expressions, which include string constants, numeric constants, date constants and binary constants
Control Attributes	Criterion that determine where the data for a control is retrieved, how the control is validated, the default value of the control, etc.
Copy	A function of the <i>Edit</i> menu that copies selected data to the clipboard enabling you to paste the data somewhere else. The data remains on the clipboard until you cut or copy different data.
Count	A query design total function that reports the total number of included rows in the column For example, if you query the <i>Salary</i> column in the <i>Employee</i> table using the <i>Count</i> total function, the query result is one row with a <i>Salary</i> column that contains the total number of employees.
Cut	A function of the <i>Edit</i> menu that deletes selected data and copies it to the clipboard enabling you to paste the data somewhere else
Data Source	Database tables or queries from which queries and forms retrieve data
Delete	A command button to delete information in a record
Delete Query	A query that creates a result set like a normal <i>Select</i> query, and then deletes each row in the result set from the underlying tables For example, you could use a <i>Delete</i> query to delete a specific employee from the <i>Employee</i> table.
Design View	A window where you design tables, queries and forms
Direction	A <i>Find</i> tool option used to instruct the system to search forward or backward. If the system reaches the end of the data without finding the value, it will prompt you to search the data in the opposite direction.
Effective Date	The date a pending change to the database is to take effect
Expression	Any combination of operators, constants, literal values, functions, fields, controls and properties that evaluates to a single value. You can use expressions as settings for many properties and action arguments to set criteria or define calculated fields in queries.
Field	An area where the user may type information
Field Expression	An expression entered as a record in a query that is used to get the value of a field for the result set
Find	A tool used to locate particular records on spreadsheets and forms
Font	Sets the font for the selected controls

Font Size	Sets the font size for the selected controls
Form	A screen used to edit data in tables, display information and/or request information from the user. Unlike spreadsheets, you can control the layout of the fields on the form, the fonts and colors, and the types of controls used.
Form Links Field	A form design attribute that contains a list of the fields on the main form that must match fields on the sub-form.
Format Strings	Design tools used to control how data is displayed on spreadsheets and forms. They are used when defining fields in a table and when defining controls on forms.
Generate Snapshot	A function designed to create a snapshot of tables that are being edited by another application
Generate Test Database	A tool designed to test the changes made to a database by the end-user's applications
Graph	A form design control tool that draws a graph on the form
Group Box	A form design control tool that draws a box on the screen to group together toggle boxes, check boxes and/or radio buttons
Group By	A query design total function that totals all rows of the specified value into a single row. When you group by a column, the query result set will have as many rows as there are unique values in the columns. For example, if you <i>Group By</i> the <i>Department</i> column and <i>Sum</i> the <i>Salary</i> column, the query result will contain one row for each department with a <i>Salary</i> column that contains the sum of all the salaries of all employees in that department.
Help	This function is not yet available.
In Group	A form design attribute of a control that designates the name of the option group to which the control belongs
Insert Query	A query that creates a result set like a normal <i>Select</i> query, and then inserts those rows in an existing table For example, you could use an <i>Insert</i> query to add a new employee to the <i>Employee</i> table.
Italic	Italicizes the text of the selected controls
Left Justify	Left justifies the text of the selected controls
Limit to List	A form design attribute that determines whether or not the user is able to type a value in a combo box that is not included in the original list
Line	A form design control tool that draws a line on the form
List box	A form design control tool that creates a list box that lists the rows in a table or query, a list of values or a list of fields in a table or query. The user is able to select a value from the list.
List Rows	A form design attribute that designates the number of rows to display in a list box when the user opens the list
Lock	A form design tool that creates a lock button. When the lock button is pressed, the current control tool remains selected until you select a different button from the tool box.
Make Table Query	A query that creates a table using selected records. The table columns match the columns in the result set. <i>Make Table</i> queries can be used to take snapshots of data for use in later queries or for partial backups.
Many-to-Many Join	A query design join function where neither side of the join includes all the fields of a unique key

Match	<p>An option available for the <i>Find</i> tool to refine your search results</p> <p>Select <i>Whole field</i> to only return values that wholly match the characters you specify, e.g. <i>Smith</i> returns <i>Smith</i>.</p> <p>Select <i>Start of field</i> to return all values that begin with the characters you specify, e.g. <i>Smith</i> returns <i>Smith</i>, <i>Smithson</i> and <i>Smithsonian</i>.</p> <p>Select <i>Any part of field</i> to return all values that include the characters you specify, e.g. <i>Smith</i> returns <i>Smith</i>, <i>Smithson</i>, <i>Smithsonian</i>, and <i>Blacksmith</i>.</p>
Max	<p>A query design total function that reports the largest value of all the included rows in the column</p> <p>For example, if you query the <i>Salary</i> column in the <i>Employee</i> table using the <i>Max</i> total function, the query result is one row with a <i>Salary</i> column that contains the largest salary of all the employees.</p>
Min	<p>A query design total function that reports the smallest value of all the included rows in the column</p> <p>For example, if you query the <i>Salary</i> column in the <i>Employee</i> table using the <i>Min</i> total function, the query result is one row with a <i>Salary</i> column that contains the smallest salary of all the employees.</p>
Multi-Line Text	A form design control tool that creates a multi-line text entry field that can be edited
Object	A component of an application, such as a table, a query or a form
One-to-One Join	A query design join function where a join is created between all the fields of a unique key in the first table to all the fields of a unique key in the second table. In this case, there will be either 0 or 1 matching record in the first table for every record in the second table.
One-to-Many Join	A query design join function where the fields on one side of a join include all the fields of a unique key and the fields on the other side of the join do not. In this case, there may be many records found to match each record from the table on the unique side of the join.
Option Group	A form design feature that groups radio buttons, check boxes, or toggle buttons, each allowing one selection
Option Value	A form design attribute of a control that designates the value to assign if the control is selected
Paste	A function of the <i>Edit</i> menu that pastes data that has either been cut or copied to the clipboard
Print	A command that prints the currently selected document
Process Transactions	A tool on the main menu that reads through the pending transaction file to see if any of the transactions have reached their effective date and, if so, applies the transaction to the database
Query	A query is a request for specific information from the database
Quick Form	An option to create a form with all the fields from a specified data source
Radio Button	A form design control tool that creates a radio button that the user can turn on/off
Receive Transactions	This function reads all the transaction files in the current location's directory and loads them into the pending transaction file where they will wait until their effective date is reached
Result Set	The data that is retrieved from a query
Right Justify	Right-justifies the text of the selected controls
Row Source	A form design attribute that designates the table or query from which the data displayed in a list box is retrieved
Save	A command that saves user input

Scroll	To move through the text on the screen (upwards, downwards, left or right) to see the parts of a list that do not fit on the screen
Search	An option for the <i>Find</i> tool that instructs the system to search in the current field or in all the fields in the record for the specified value
Select Queries	Queries that collect data from one or more tables to display in a spreadsheet
Selector Buttons	Selector buttons appear on the left side of a spreadsheet beside each row and are used to select records to move, copy or delete
Size to Fit	An option on the Form Design Layout menu to size controls to match the size of the data in the controls
Spreadsheet	Table data, query results and form data can all be displayed in spreadsheet format, which is easy to browse and edit. There is one record on each row and one field in each column.
Static	A form design control tool that creates a text label that cannot be edited
Sub form	A form design control tool that creates a sub-form, which is used to display related information from a different table or query on the same form For example, you could display order information on the main form and the items in the order on the sub-form.
Subform Link Fields	A form design attribute that contains a list of the fields on the sub-form that must match the fields on the main form
Sum	A query design total function that puts the sum of the column values in all the included rows in the column For example, if you query the <i>Salary</i> column in the <i>Employee</i> table using the <i>Sum</i> total function, the query result is one row with a <i>Salary</i> column that contains the sum of all the salaries of all the employees.
Tab Control	A form design control tool that creates a tab control allowing related fields to be grouped together on separate tabs
Table	A collection of related information, such as customers, orders, or employee time cards
Table Relation	A saved join between tables in a query design
Termination Date	The date on which a changed value in the database will revert back to its original value
Text	A form design control tool that creates a text entry field that can be edited
Toggle Button	A form design control tool that creates a toggle button that is depressed when its source is True
Total Query	A query that combines all the records of its result set using the functions <i>SUM</i> , <i>AVG</i> , <i>MIN</i> , <i>MAX</i> and <i>COUNT</i>
Transaction File	A file containing information concerning added, changed, or deleted records that is transmitted to either a remote location from the central location or from the central location to a remote location
Transfer Files	A tool found on the Main Menu that transmits transaction files between locations
Unbound Control	A form design control that is typically used when you are creating a form to prompt the user to enter parameters for a query, or when you want the user to enter a value that is used by another part of the form
Underline	Underlines the text of the selected controls
Undo	A command that allows the user to undo their changes

Union Join	A query design join function where no fields from either source are joined together. In this case, all records from the first table are matched with all records in the second table. For example, if there are 10 records in the first table and 20 records in the second table, there will be 200 records in the result set.
Update Query	A query that creates a result set like a normal <i>Select</i> query, and then updates each row in the result set using an expression that you provide for each column to update. For example, you could use an <i>Update</i> query to give all employees in Department D14 a 10-percent raise.
VCR Controls	On both forms and spreadsheets, the user can easily navigate records by using the VCR controls on the toolbar or by using the keyboard navigation keys.
Validation Rule	A form design attribute that contains an expression that must be True for the data in the field. For example, if you want the <i>Married</i> field to always contain a Y or N, then you would set the <i>Validation</i> rule attribute to Y or N.
Validation Text	A form design attribute that specifies the error message to display if the validation rule is broken
Where	A query design total function that limits the records that are included in the query by setting conditions on the column. For example, if you set the total function of the <i>Id</i> column to <i>Where</i> and the condition to <1000, then only records with <i>Id</i> 's less than 1000 will be included.

Extending EDM Functionality

The system architecture supports plugin functionality on both the client- and server-side enabling engineers to create and add their own functions.

Implementing Plugins

1. Create a [client side command class](#) that will gather the necessary parameters from the user and send them to the server for processing.
2. Create a [server side command executor class](#) that will do the processing.
3. Create a [Client Plugin Loader Class](#) class file to add the command to the system on the client side.
4. Create a [Server Plugin Loader Class](#) class file to add the command to the system on the server side.
5. Modify [Config.xml](#) to load the class files for the [client](#) and the [server](#) when Tomcat loads the EDM Web application.

Client Plugins

Client plugins can be used to add functionality to the client side of the system, which is the part the users interact with via the browser or standalone Java application.

```
package mypackage.plugin;

import EDMcommon.plugin.*;

/**
 * Get the value of a property in a property file.
 */
public class GetPropertyFileValueCmd extends AbstractCommand implements
ExecutableCommand<GetPropertyFileValueCmd>, PluginLoader {
    /** Name of the property whose value is to be returned. */
    private String propertyName;
    /** Path and name of property file. */
    private String propertyFileName;
```

```

/**
 * Get the value of a property in a property file.
 */
public GetPropertyFileValueCmd() {}

/**
 * Get the value of a property in a property file.
 *
 * @param propertyFileName Path and name of property file.
 * @param propertyName Name of the property whose value is to be returned.
 */
public GetPropertyFileValueCmd( String propertyName, String
propertyFileName ) {
    this.propertyName = propertyName;
    this.propertyFileName = propertyFileName;
}

/** {@inheritDoc} */
public void addPluginsToLibrary(PluginLibrary pluginLibrary) {
    pluginLibrary.add(
        this.getClass(),
        "getPropertyFileValue",
        "Get the value of a property in a property file.",
        "Property value",
        PluginLibrary.EDM_CATEGORY,
        new PluginParm[] {
            new PluginParm("propertyName", "Name of the property whose value is to
be returned."),
            new PluginParm("propertyFileName", "Path and name of property file."),
        }
    );
}

/**
 * Get the value of a property in a property file.
 * @param propertyFileName Path and name of property file.
 * @param propertyName Name of the property whose value is to be returned.
 * @return Property value or null if property name not found.
 */
public static String getPropertyFileValue( String propertyName, String
propertyFileName ) {
    return new GetPropertyFileValueCmd( propertyName, propertyFileName
).send();
}

/** @return Command executor. */
public Executor<GetPropertyFileValueCmd> getExecutor() {
    return new EDMserver.cmd.GetPropertyFileValueCmdExecutor();
}

/** {@inheritDoc} */
@Override
public String send() {

```

```
    return (String)super.send();
}

/**
 * @return Path and name of property file.
 */
public String getPropertyFileName() {
    return propertyFileName;
}

/**
 * @param propertyFileName Path and name of property file.
 */
public void setPropertyFileName(String propertyFileName) {
    this.propertyFileName = propertyFileName;
}

/**
 * @return Name of the property whose value is to be returned.
 */
public String getPropertyName() {
    return propertyName;
}

/**
 * @param propertyName Name of the property whose value is to be returned.
 */
public void setPropertyName(String propertyName) {
    this.propertyName = propertyName;
}
```



```
}
```

Client Plugin Loader Class

The plugin load class loads one or more plugins into the system and makes them available.

```
package mypackage.plugin;

import EDMcommon.plugin.PluginLibrary;
import EDMcommon.plugin.PluginLoader;

/** My client plugins. */
public class ClientPlugins implements PluginLoader {

    /** Default Constructor. */
    public ClientPlugins() {}

    /** Add plugin functions to the given function library. */
    @Override
    public void addPluginsToLibrary( PluginLibrary pluginLibrary ) {
        pluginLibrary.add( new GetPropertyFileValueCmd() );
        pluginLibrary.add( new EditVersionsFunction() );
    }
};
```

Loading Client Plugins at Runtime

Client plugins are loaded from jar files. To load the plugin loader class, add a line to [Config.xml](#) similar to the following:

```
<clientPlugin class = "mypackage.plugin.ClientPlugins"/>
```

The *GetPropertyFileValueCmd* class is on the client-side and must be available to the applet, which means the jar file must be included with the applet. You can package your plugin code into a jar file and have *webapps/EDM/LoginPage.jsp* load your jar file by adding the jar file name to the two lines that contain the list of jar files as shown below.

```
<PARAM NAME = ARCHIVE VALUE =
"EDMclient.jar,imageconversion_Runtime.jar,forms-1.2.1.jar,jcalendar-1.3.2
.jar,myplugins.jar" />
```

```
ARCHIVE =
"EDMclient.jar,imageconversion_Runtime.jar,forms-1.2.1.jar,jcalendar-1.3.2
.jar,myplugins.jar"
```

Server Plugins


```
* Get the value of a property in a property file.
*
* @param propertyFileName Path and name of property file.
* @param propertyName Name of the property whose value is to be returned.
* @return Property value or null if not found in file.
*/
public static String getPropertyFileValue( String propertyName, String
propertyFileName ) {
    String value = null;
    try {
        if (propertyFileName == null || propertyFileName.length() == 0) {
            throw new IllegalArgumentException("Property file name not
specified.");
        }
        if (propertyName == null || propertyName.length() == 0) {
            throw new IllegalArgumentException("Property name not
specified.");
        }
        File file = CurrentDir.getRelativeFile( propertyFileName );
        propertyFileName = file.getAbsolutePath();
        if (!file.exists()) {
            throw new IllegalArgumentException("Property file not found.");
        }
        Properties properties = new Properties();
        properties.load( new FileInputStream( file ) );
        value = properties.getProperty( propertyName );
    } catch( Throwable t ) {
        throw new MsgError("Error getting property {0} value from file {1}.",
propertyName, propertyFileName );
    }
    return value;
}
```

```
}
```

Server Plugin Loader Class

The plugin load class loads one or more plugins into the system and makes them available.

```
package mypackage.plugin;

import EDMcommon.plugin.PluginLibrary;
import EDMcommon.plugin.PluginLoader;

/** My server plugins. */
public class ServerPlugins implements PluginLoader {

    /** Default Constructor. */
    public ServerPlugins() {}

    /** Add plugin functions to the given function library. */
    @Override
    public void addPluginsToLibrary( PluginLibrary pluginLibrary ) {
        pluginLibrary.add( new GetPropertyFileValueCmdExecutor() );
    }
}
```

Loading Server Plugins at Runtime

Server plugins are loaded from jar files. To load the plugin loader class, add a line to [Config.xml](#) similar to the following:

```
<serverPlugin class = "mypackage.plugin.ServerPlugins" />
```

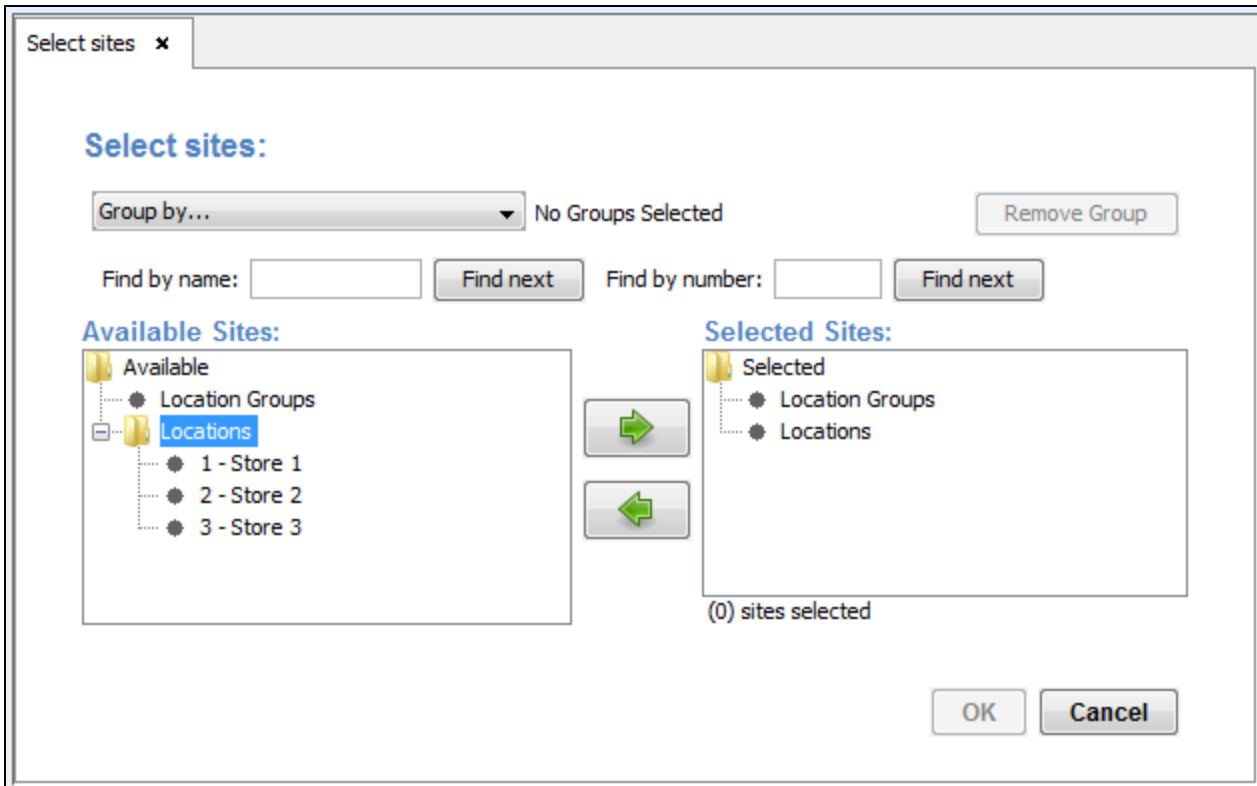
The *GetPropertyFileValueCmdExecutor* class is on the server-side and must be available to the Web application. You can package your plugin code into a jar file and save the jar file in the *webapps/EDM/WEB-INF/lib* directory.

Translating EDM System Text

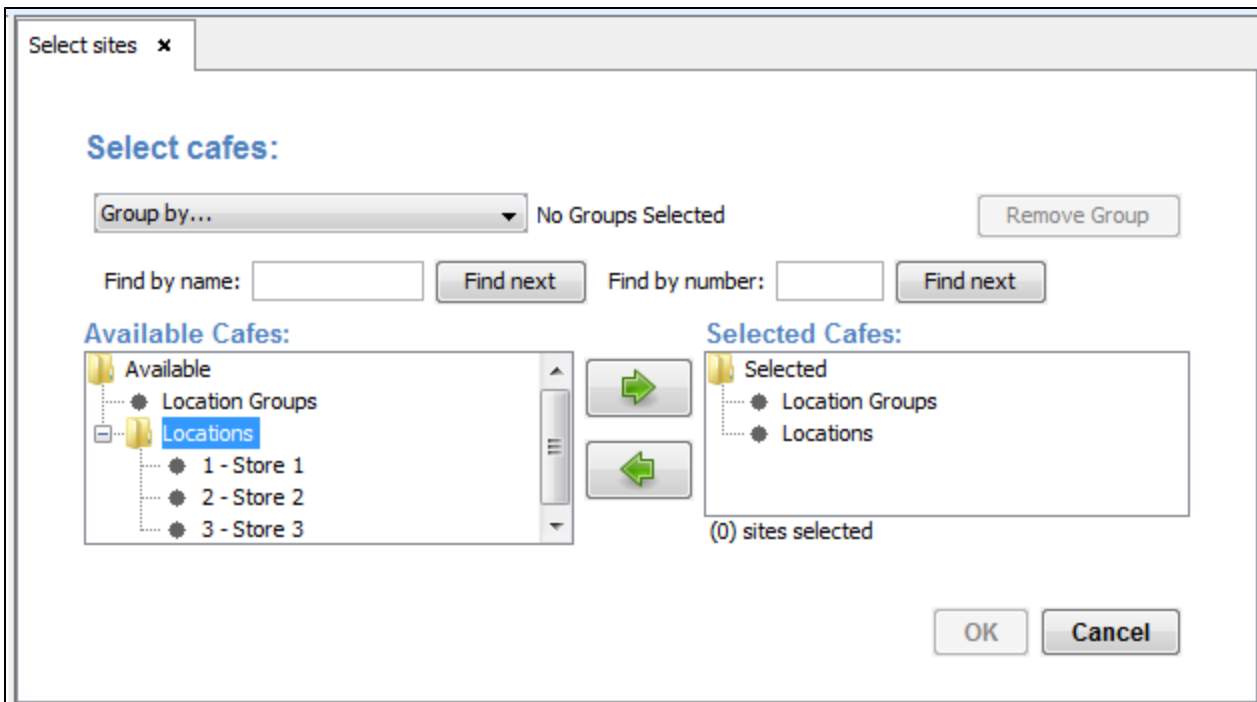
You can translate all the text used within EDM to other languages, or change the text to match the terms used by your organization. The text is translated using message files, which have different suffixes to represent different languages. For example, the French translation would be stored in the *Message_fr.java* message file. The *Messages_en_US.java* file contains the English translation. The Java Runtime loads the appropriate language file based on its locale settings.

In the following example, the English language is used, but we are changing the term "Site" to "Cafe" on the EDM forms.

Before Translation



After Translation



Updating the Message Files

To translate the EDM text, edit the message files as shown below. In this example, we are editing the *Messages_en_US.java* file.

```

/* Copyright 1998-2004 by xpient Solutions, LLC. All rights reserved. */
package EDMcommon.msg;

import java.util.*;

/**
 * Localized messages. The contents array contains string pairs. The
 * first
 * string in each pair is the message string passed to the Msg.get() method
 * in the program. The second string in each pair is the localized
 * version.
 */
public class Messages_en_US extends ListResourceBundle {

    private static final Object[][] CONTENTS = {
        // LOCALIZE THIS
        {"Select locations and dates", "Select cafes and dates" },
        {"Select sites:", "Select cafes:"},
        {"Available Sites:", "Available Cafes:"},
        {"Selected Sites:", "Selected Cafes:"},
        // END OF MATERIAL TO LOCALIZE
    };

    protected Object[][] getContents() {
        return CONTENTS;
    }

}

```

Updating EDM with the Translated Messages

Once the appropriate message files have been updated, the files need to be compiled and added to the *system.jar* and *EDMclient.jar* files.

1. Copy *EDMclient.jar* from the *WebApps/EDM* directory to the directory that contains the updated message files.
2. Copy *system.jar* from the *WebApps/EDM/WEB-INF/lib* directory to the directory that contains the updated message files.
3. To compile the message files and add them to the jar files, run a script like the DOS batch file shown below:

```

@echo off

echo Compiling EDM English Messages..
set JAVA_HOME=C:\Program Files (x86)\Java\jdk1.6.0_07
"%JAVA_HOME%\bin\javac" \-J-Xmx256m \-d . \-deprecation \-Xlint
\Xlint:-serial Messages_en_US.java
if ERRORLEVEL 1 goto End

echo Updating JAR files...
"%JAVA_HOME%\bin\jar" \-uf system.jar
EDMcommon/msg/Messages_en_US.class
"%JAVA_HOME%\bin\jar" \-uf EDMclient.jar
EDMcommon/msg/Messages_en_US.class

goto End

:End

```

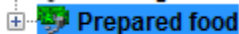
4. Copy the updated *EDMclient.jar* to the *WebApps/EDM* directory to replace the existing file.
5. Copy the updated *system.jar* to the *WebApps/EDM/WEB-INF/lib* directory to replace the existing file.
6. Restart the web server and verify the EDM text has been properly translated.

Tax Rules Editor for IRIS

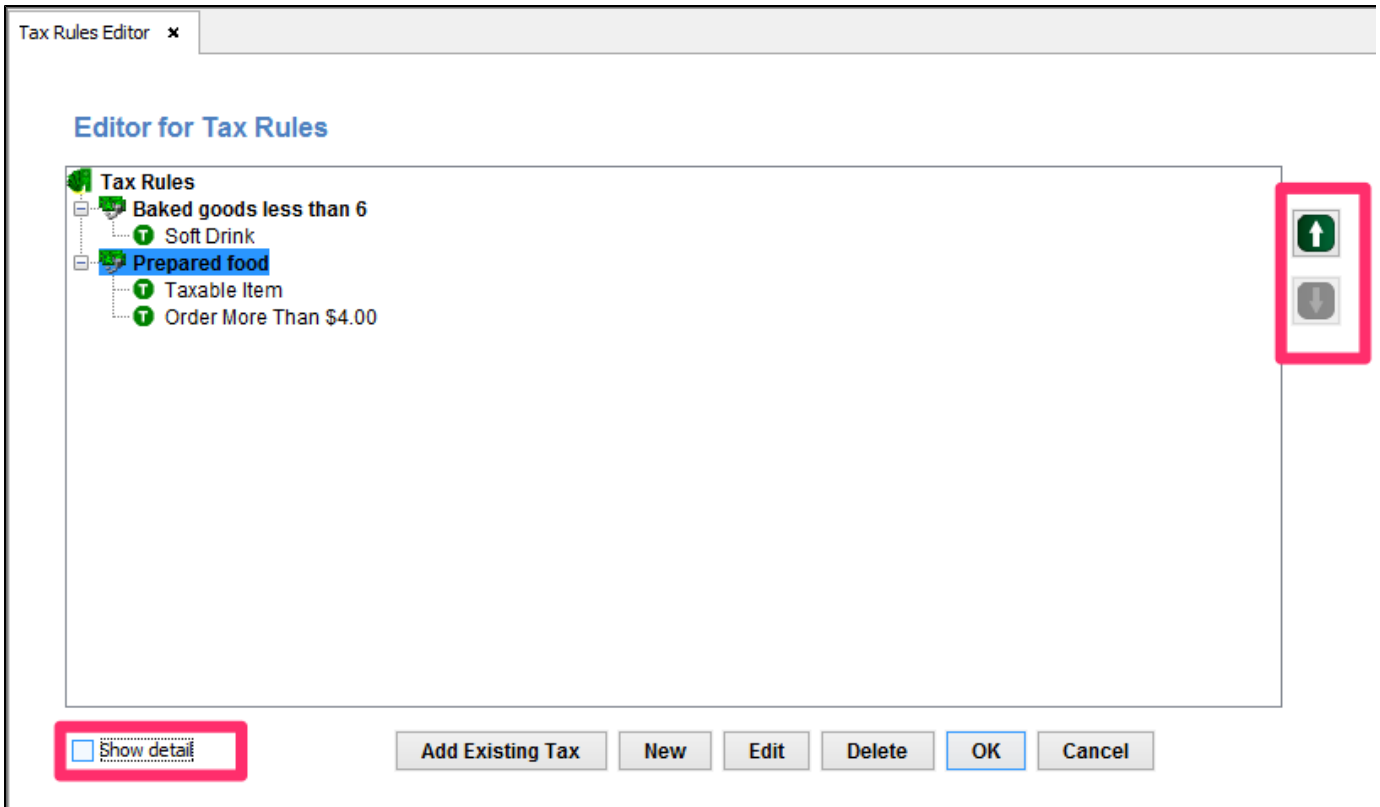
Use the *Tax Rules Editor* to define tax rules and corresponding tax rule conditions. On new installations, the *Tax Rules Editor* is opened by clicking *POS - Tax Rules*. If this is not found on your menu, you can add it by creating a menu item where the action is 'editIrisTaxRules()'.

The *Tax Rules Editor* lists existing tax rules. The tax rules are listed by priority where the tax rule at the top of the list has the highest priority. To move a tax rule up higher in the list, click the tax rule, and then click the *Move Up* button on the right side of the window. You can also move a tax rule's priority down by clicking the *Move Down* button. The *Move Up* and *Move Down* buttons will be enabled only when a tax rule's priority can be changed.

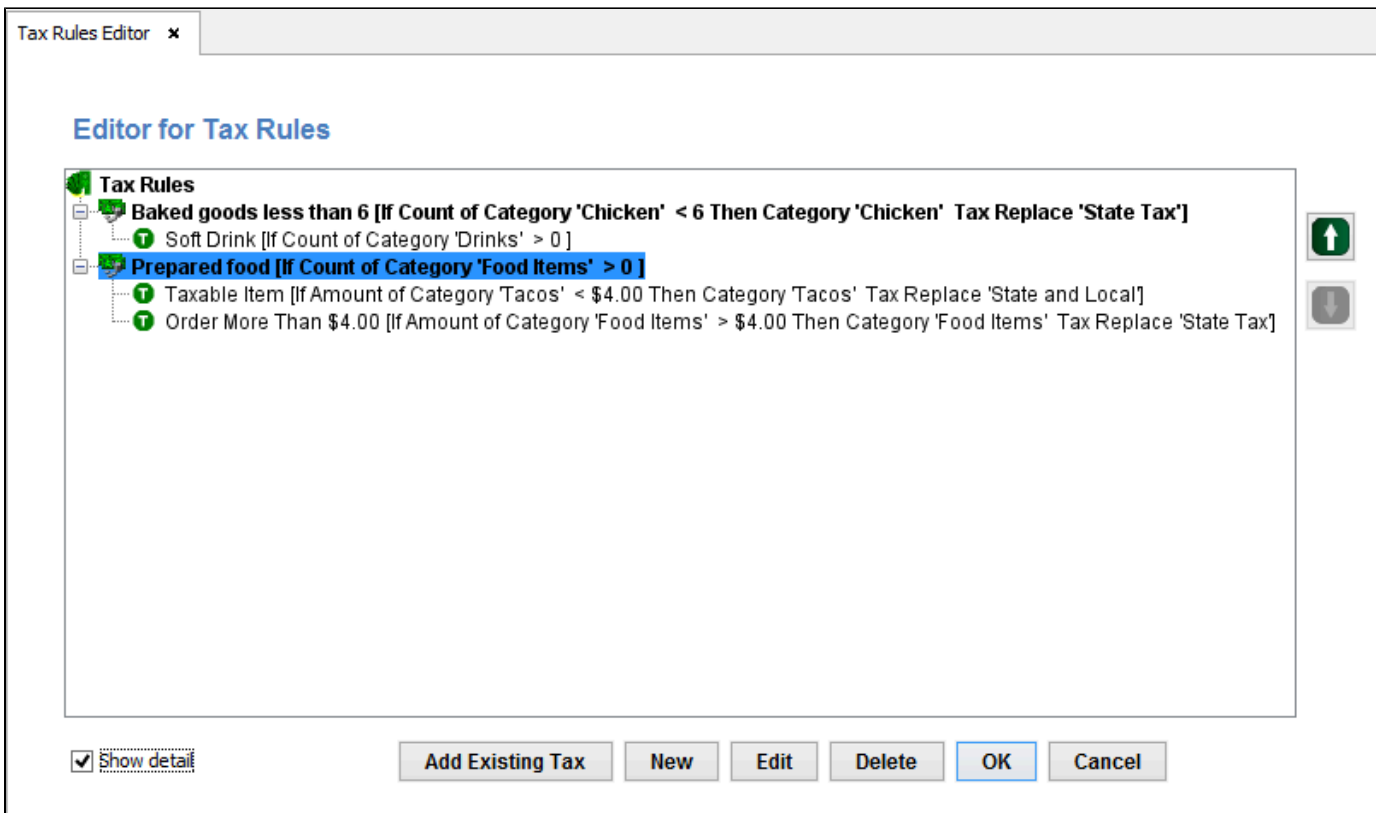
To display the conditions defined within a tax rule, click the plus sign



next to the tax rule name. In the following sample screen, the *Prepared Food* tax rule is selected.



To view the details of your tax rules and their corresponding conditions in the main window, check the *Show detail* checkbox.



To view the properties of a tax rule or a tax rule condition, select the tax rule or condition, and then click the *Edit* button. The *Tax Rule Properties* window opens.

To deactivate a tax rule, deselect the *Active* checkbox, and then click *OK*.

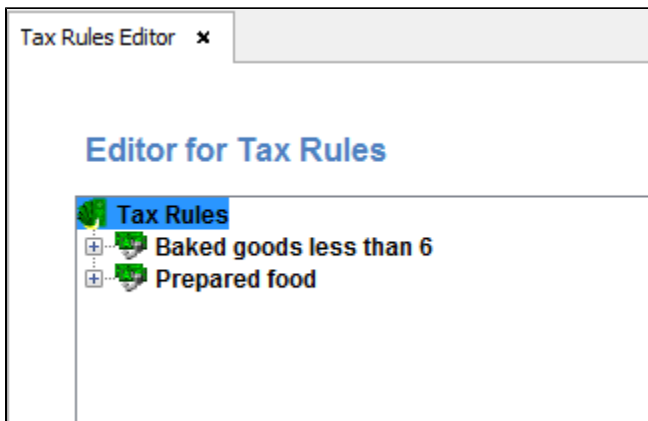
The screenshot shows a window titled "IRIS_dbo_tblTaxRules" with a tab icon. The window contains the following fields:

- Rule Name: Taxable Item
- Active: (highlighted with a red box)
- If: Amount (dropdown) Of: Category (dropdown) 20, Tacos (dropdown) Is: < (dropdown) 4.01 (text box)
- Then: Category (dropdown) 20, Tacos (dropdown) Tax: Replace (dropdown) State and Local (dropdown)

At the bottom of the window, there are "OK" and "Cancel" buttons, and a small icon in the bottom right corner.

Adding a New Tax Rule

1. Click the *Tax Rules* icon at the top of the list.



2. Click the *New* button. The *New Tax Rule* window opens.

IRIS_dbo_tblTaxRules x

Rule Name: Active:

If Of Is

Then Tax

OK Cancel

3. Type the name of the tax rule in the *Rule Name* field.
4. Specify whether the tax rule is *Active* or *Inactive*.
5. Select the specifics of the tax rule from the drop-down boxes of the remaining fields. The remaining fields together with the interjecting words, *If*, *Of*, *Is*, *Then*, and *Tax*, form a sentence, which states the rule.
In the following sample screen, a tax rule has been created that instructs the system to add Sales Tax to the item "Deal 1", if the amount of the item is greater than \$3.00.

IRIS_dbo_tblTaxRules x

Rule Name: Active:

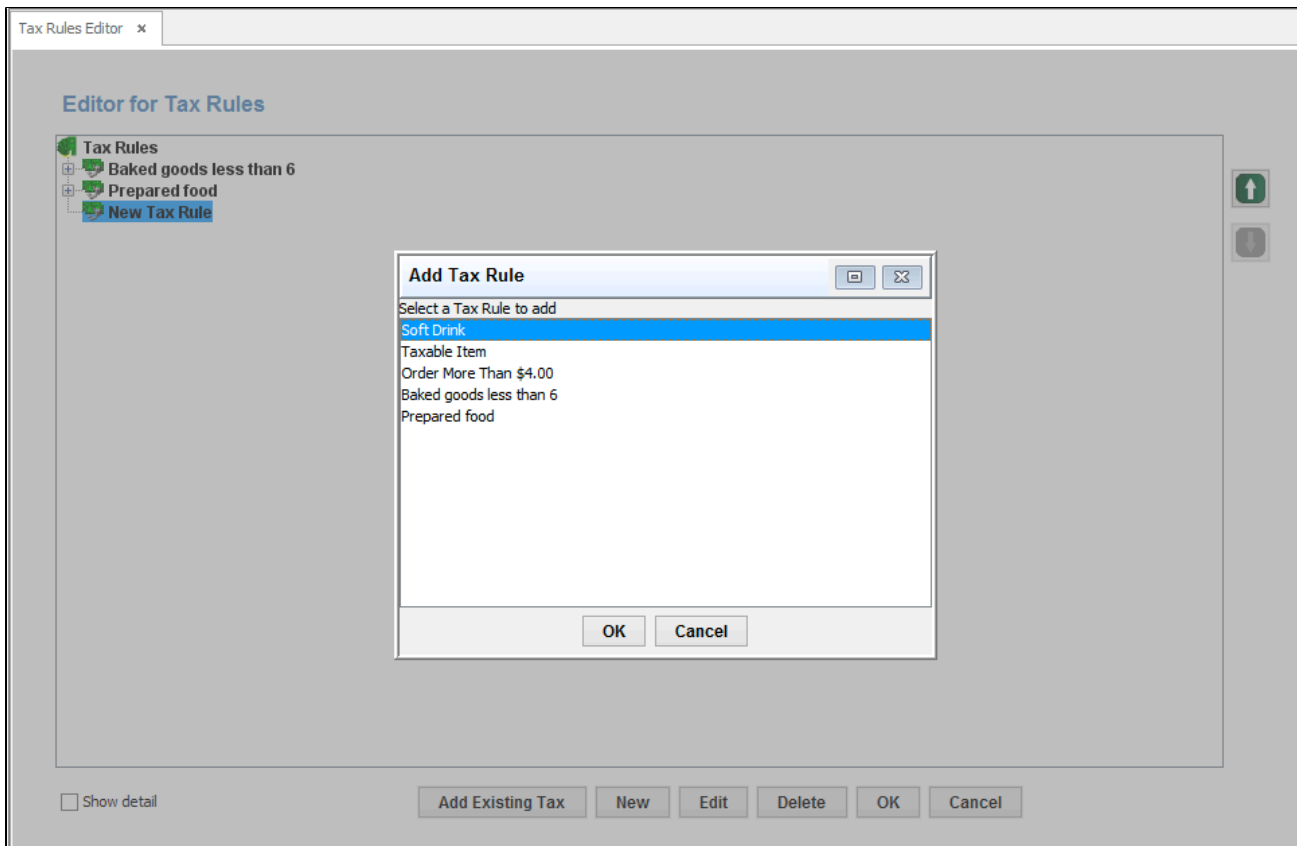
If Of Is

Then Tax

6. Click *OK* to save your changes.

Adding an Existing Condition to a Tax Rule

1. Select the tax rule to add the condition to.
2. Click the *Add Existing Tax* button. The *Add Tax Rule* window opens. This window lists the existing tax rules and conditions.
3. Select the condition to add from the list.
4. Click *OK*. The condition is included as part of the tax rule you selected.



Adding a New Condition to a Tax Rule

1. Select a tax rule to add the new condition to.
2. Click the *New* button.
3. Type the name of the condition in the *Condition Name* field.
4. Specify whether the Condition is *Active* or *Inactive*.
5. Select the specifics of the condition from the drop-down boxes of the remaining fields. The remaining fields together with the interjecting words, *If*, *Of*, *Is*, *Then*, and *Tax*, form a sentence, which states the condition. In the following sample screen, a tax rule condition has been created that instructs the system to add Sales Tax to the order if there are more than 2 drinks.

6. Click *OK*. The condition is included as part of the tax rule you selected.

IRIS Config Group Editor

The *IRIS Config Group Editor* allows users to add, update, and delete config groups (INI files), sections, keys, and key values in the IRIS database from the EDM Server. This editor is was implemented as part of the IRIS2W project. Click [here](#) to go to the IRIS2W home page.

Config Group Components

A Config Group consists of:

- A Config Group definition
- One or more Sections associated with that Config Group

Each Section consists of:

- A section definition
- One or more Keys

A Key consists of:

- A key definition
- One or more Values, where each value is associated with a register (either Default, Back Office, or a numbered register)

The following describes how a Config Group and its components may be created and edited.

Selecting Sites for a Config Group Package

On new installations, the *IRIS Config Group Editor* is opened by clicking *Menus – Edit Config Groups*. If this is not found on your menu, you can add it by creating a menu item, where the action is 'editIrisConfigGroups()'. Opening the *IRIS Config Group Editor* will create a new tab.

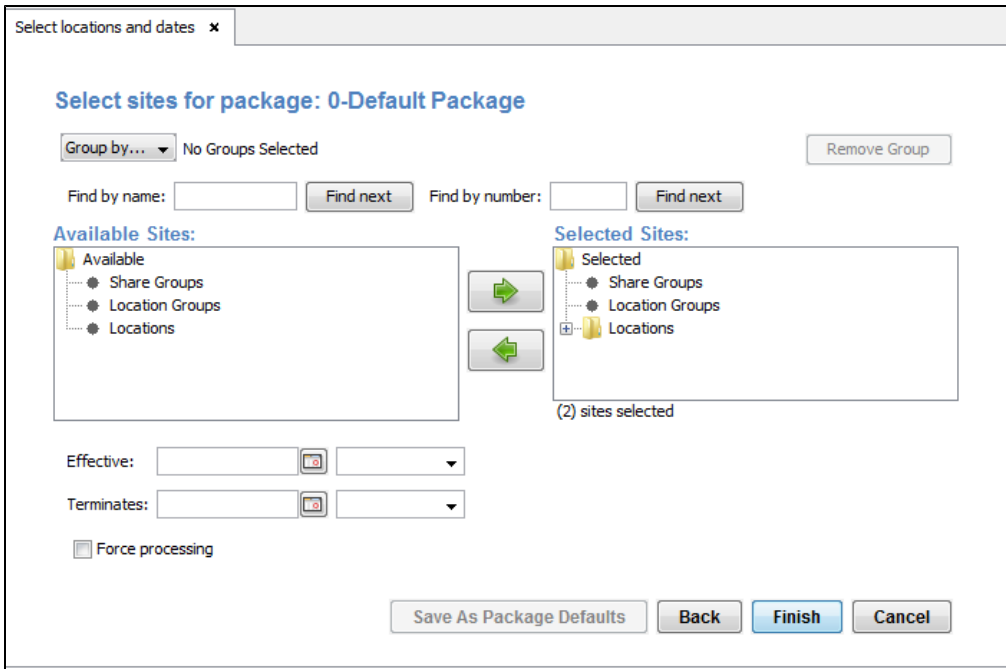
The first step in editing Config Groups is to select or create a package.

Select locations and dates x

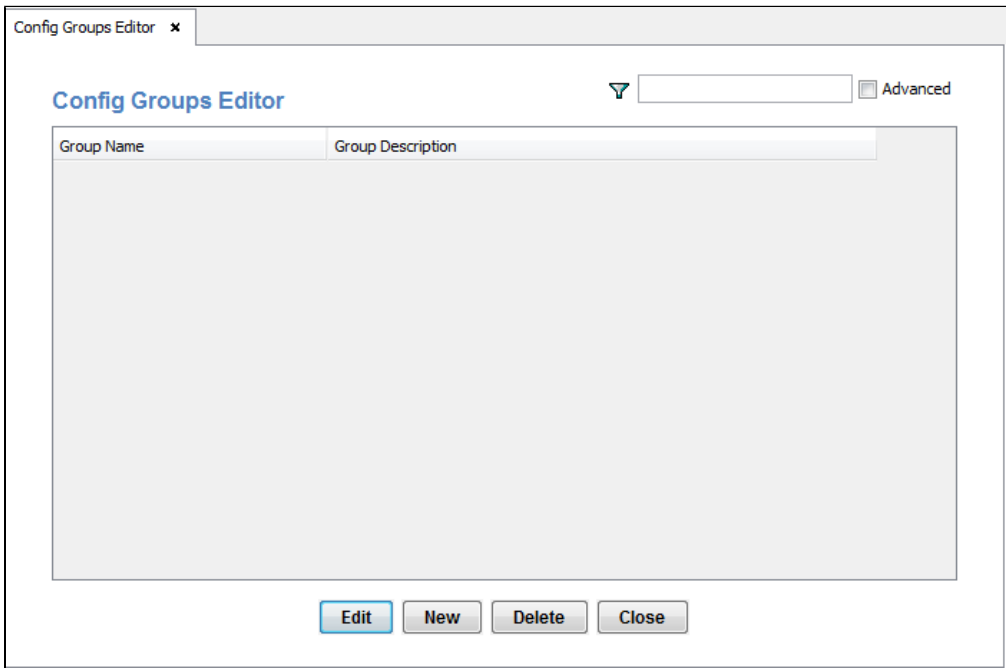
Select package or create a new package Advanced

0-Default Package
Created by 'Administrator' on '2/24/15'

Next, select the sites for the package.



Once the site selection is finished, the *IRIS Config Group Editor* is opened.



An incremental filtering of existing Config Groups can be performed using the search panel:



Adding a New Config Group

To add a new Config Group, click *New*. This will open the Config Group Definition panel.

Type the *Name*, *Description*, and *File Name* associated with the group in the provided fields.

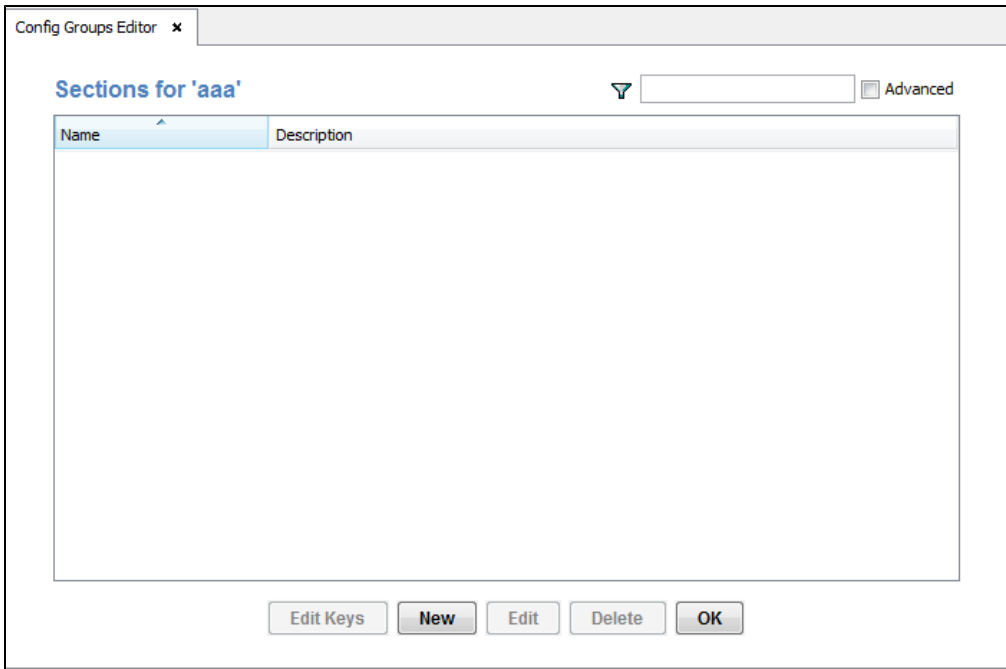
Select the applicable output type from the *Output type* drop-down listbox. The following table identifies the available output types.

Output Type	Description
CCA	Select CCA if the Config Group is related to CCA .
IRIS	Select IRIS if the Config Group is related to IRIS POS terminals.
NONE	Neither <i>Output Path</i> or <i>File Name</i> is required. These fields will be disabled.
PATH	Type the output path in the <i>Output Path</i> field.

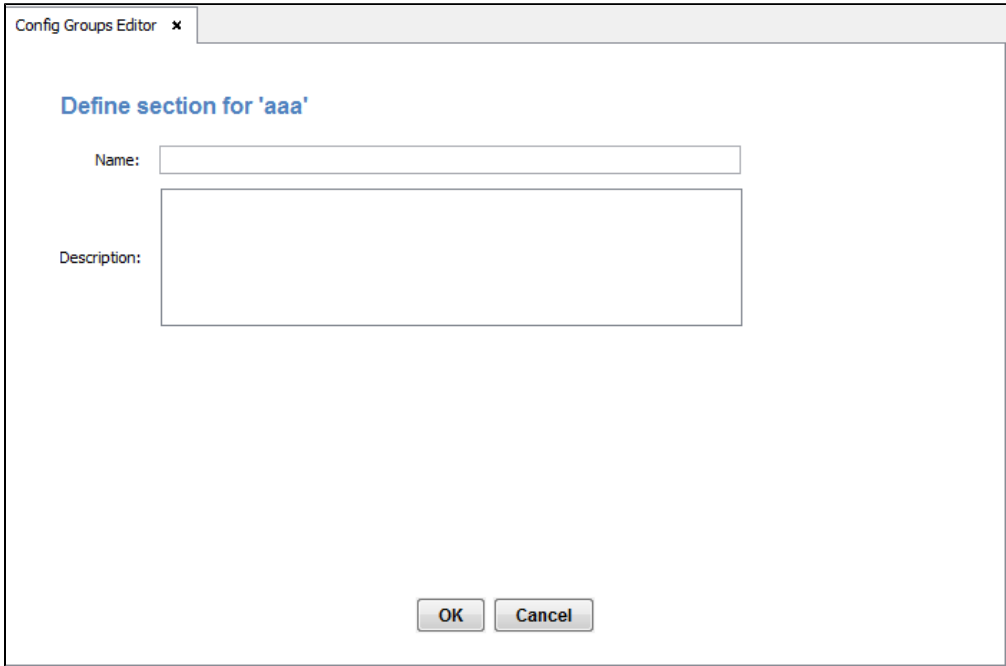
Defining Sections

Select *Edit Sections* to open the *Sections* page. From this page you will define Sections for the Config Group. Sections may be added to a Config Group even if this is a new Config Group for which no fields have been defined. However, the Config Group and its component sections cannot be saved until the required Config Group Definition fields have been provided.

In this example, the Config Group is named 'aaa'.



Select *New* to open the Section definition page. Provide a name and description for the Section in the provided fields



In this example, the Section name is 'xxx'. Select *OK* to add the section to the *Sections* page.

Sections for 'aaa' ▼ Advanced

Name	Description
xxx	

Any existing section may be selected on the page and edited or deleted. Selecting *OK* will return you to the Config Groups page.

Defining Keys

Select *Edit Keys* from the *Sections* page to open the Keys page. The *Keys* page displays any existing Keys for the Section.

Keys for 'xxx' ▼ Advanced

Name	Description
------	-------------

Click *New* to add a new Key. Type a name and description for the key in the provided fields.

Define key for 'xxx'

Name:

Description:

In this example, the Key name is 'yyy'. Select *OK* to add the Key to the *Keys* page.

Keys for 'xxx' Advanced

Name	Description
yyy	

Any existing Key may be selected on the page and edited or deleted. Selecting *OK* will return you to the *Sections* page.

Defining Key Values

From the *Keys* page, select the Key for which you want to define values, and then click *Edit Key Values* to open the *Values* page. The *Values* page displays any existing values for the selected Key.

Values for 'yyy' [Filter] [Advanced]

Register Number	Value	Comment

Key Values consist of the following properties.

Key Value Property	Description
Register Number	The IRIS Destination is either 'All Default', 'Register Default', 'Back Office', or a positive integer. The IRIS destination is only required if the Configuration Group output type is 'IRIS'. Each Value associated with a particular Key must have a unique register designation (that is, there cannot be more than one 'Default' value, or more than one value with the same register number).
Value	The Value can be an arbitrary string.
Comment	The Comment can be an arbitrary string.

Select *New* to open the Value definition page. If a Key Value having a Default or Back Office register designation already exists for this Key, the Default and Back Office options will be disabled.

Select an unused Register Designation, and then enter a Value and Comment.

Define value for 'yyy'

IRIS Destination

All Default
 Register Default
 Back Office
 Register Number

Value:

Comment:

Select *OK* to add the Value to the selected Key.

Values for 'yyy' [Filter] Advanced

Register Number	Value	Comment
Default	zzz	zzz comment

Any existing Key Value may be selected on the page and edited or deleted. Selecting *OK* will return you to the *Keys* page.

Managing Different Config Group Versions

When editing a Config Group, your changes will only apply to the Sites you selected for the package (as described in the *Selecting Sites for a Config Group Package* section above). As a result, it is possible for different versions of a Config Group to exist on different Sites, i.e. different Sites may have a Config Group of the same name, but with different information (differing output types, file names, etc.).

If the Site selection includes a site for which differing Config Group versions exist, then when the user edits a particular Config Group, the Version selection page will be shown. If different Sites have different Section definitions for the same Config Group, these will be regarded as different group versions, and will be listed separately in the Version selection page. If different Sites have different Key definitions for the same Config Group Section, these will be regarded as different group versions, and will be listed separately in the Version selection page.

Here we see the Version selection page for a Config Group named 'a', whose description is 'description group a'.

Versions of: a, description group a

Locations

1
2,3

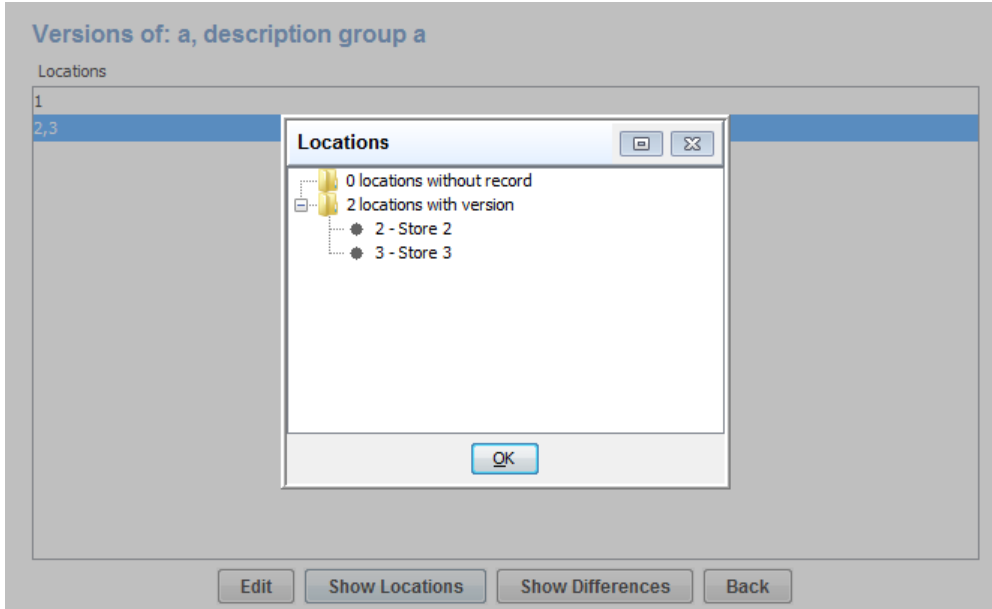
Each distinct version is listed on the page. Here, the first version exists only on Site 1. A second version is shared by Sites 2 and 3. You may select a single version, and then select *Edit* to edit that version.

If a Config Group site version is edited so that it has no differences with an existing version, this will be reflected on the Version selection page.

Either the site will be added to the existing version, or the edited version may be incorporated completely into the existing version and no longer listed separately.

Show Locations

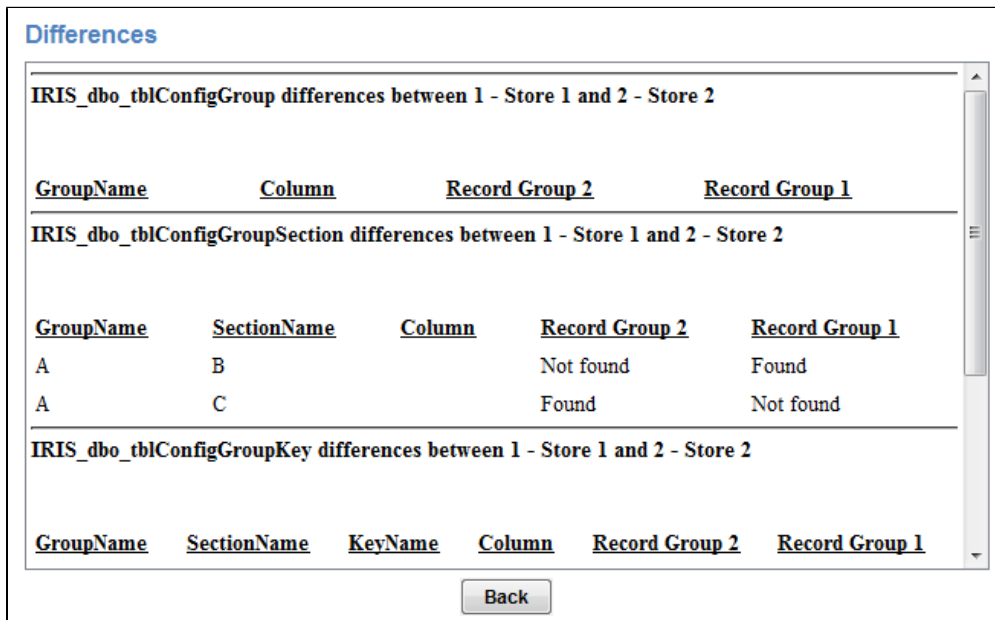
You may select a version, and then see details of the version Sites by selecting the *Show Locations* button.



If there are any locations without a Config Group named 'a', they would be listed first followed by the locations that have this version (including the Site names). Note that Site, Location, and Store are interchangeable terms. Selecting OK will return you to the Version selection page.

Show Differences

Existing versions can be compared from the Version selection page. Select the versions to compare, and then select the *Show Differences* button to open the Differences page.



This will list each difference between the versions by *Group Name* and *Field Name*. In the example above, it is reported that Store 1 has a Section named 'B' that the Store 2 Config Group does not have. The opposite is the case with Section 'C'. Differences between Keys and Values will also be reported here. Select *Back* to return to the Version selection page.

Editing a Key Value

When editing a Key Value, it is possible for different Sites to have different values for the same register designation (This is related to the *Selecting Sites for a Config Group Package* section above). If this is the case, the option to view the values and comments for all sites will be available:

The dialog box is titled "Define value for 'yyy'". It contains the following elements:

- IRIS Destination:** A group box containing four radio buttons: "All Default" (selected), "Register Default", "Back Office", and "Register Number". A small text input field is next to "Register Number".
- Value:** A text input field containing "zzz".
- Values for all sites in package:** A button.
- Comment:** A text input field containing "zzz comment".
- Comments for all sites in package:** A button.
- OK** and **Cancel** buttons at the bottom.

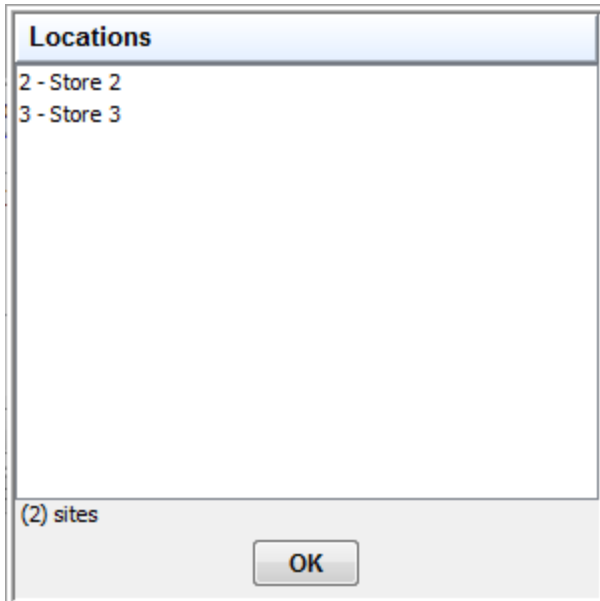
Selecting the *Values for all sites in package* option will open the *Location Values* dialog box:

The dialog box is titled "Location Values" and features a list box with the following entries:

- e site 1
- value e site 2

The "value e site 2" entry is highlighted with a blue border. At the bottom, there are three buttons: "Set to NULL", "Locations", and "OK".

This box will display all the values of the selected Config Group Section Key for all selected Sites in the Package having the same Register Designation. Each value can be selected and edited. Selecting the *Locations* button will list the Sites that have the selected value.



Selecting *Set to NULL* will remove the value from the locations. Similar operations can be performed on the value comments.

Master Data Management

This is the EDM Master Data Management home page. In these pages we describe what is this new EDM module, what are its benefits, and how it works.

If you need assistance or have any comments and doubts, please feel free to contact [Octavio Povoá](#).

Introduction

Master Data Management (MDM) helps business manage critical data used by one or more systems in a centralized way. For example, a restaurant chain may have a point-of-sale (POS) system in each restaurant, a web application for placing orders online, and an inventory system. Each of these different systems needs a list of items and categories for sale, and these lists need to be synchronized at all times for the systems to work properly. MDM allows users to share a single data set across multiple systems.

In Enterprise Data Manager (EDM), master data is stored in tables in the central database. When users change the master data, EDM automatically applies the changes to all the applications that use that information so they always have the same view of the data.

Benefits of using the MDM module of EDM

This section highlights the main benefits of using this new EDM module.

Eliminate data inconsistencies across multiple systems

Common data such as the list of item prices can be synchronized across all systems that require this information, such as the POS system, the online ordering system, enterprise reporting systems, and so on.

Increased productivity

Users can make one change that correctly updates multiple systems instead of having to make the change in each system.

Fulfill business requirements not met by applications

Master tables can be custom designed by each business to meet their specific requirements. For example, one business may need to keep track of documentation for certain items sold and can put a reference to the documentation right in the table - even though none of the applications that use the item list need that information.

Simplified change management

When users want to change the data in EDM without MDM, they need to create or use a "package" to hold the changes until they are completed and sent to the applications running at remote sites.

The MDM module eliminates the package step entirely by creating a new version of the data set that needs to be changed, making changes to the new version, then sending the new version to the applications. By using this procedure, there is no need for the user to create change "packages". Users don't edit the data that is currently in use by the applications anymore - they now edit a new version of that data.

Increased usability

Without MDM, EDM users have to select the sites to be changed individually. When there are many thousands of sites, this can be a time-consuming process. With the MDM module, users can simply choose from a short list of the existing versions of the data instead of browsing through a long list of individual remote sites.

Reduced data redundancy

If there are business requirements that are not easily fulfilled with the existing data structures in the application, the master tables can be structured as needed. For example, if the POS item table contains a tax category but not all the restaurants in a chain use the same tax category, then the users cannot share a single item list across all restaurants using the existing table structure. With MDM, the tax category can be moved out of the item table and placed in its own table so that a single set of item data can be shared across all restaurants.

Increased performance

By using EDM's MDM module, since changes are made to master data instead of tables that are assigned directly to sites, there are far fewer transactions to be processed by the system. For example, if a company has 5,000 remote sites that share an item list, 10 changes to a new version of the master item list only create 10 transactions for the change and a single transaction to send the updated item list to each site for a total of 5,010 transactions. Without the EDM module, all 10 transactions would be created for each site for a total of 50,000 transactions.

Improved collaboration

Master data that affects multiple applications can be created and updated by multiple users in different departments through a single system.

Enable cross-system processes

Business processes can use master data instead of having to access data from multiple information silos and departments.

Simplified interaction with business partners

System that exchange data with business partners can use master data instead of having to interface to separate individual system with different interfaces.

Better business process improvement metrics

Metrics can be more easily generated on processes that interact with master data rather than having to interact with multiple systems and interfaces.

Topics

How Does Master Data Management Work

MDM introduces several key new concepts to EDM, including shared table groups, configuration types, configurations, versions, and subscriptions. This page explains each of these concepts, along with the steps to configure Master Data Management on EDM.

Click [here](#) to return to the main EDM Master Data Management page.

MDM Concepts

Shared Table Groups

Shared Table Groups are used to group related data tables that will be used by MDM to update specific areas of the enterprise solutions (such as the price tables for the Point-Of-Sale application in the stores).

Configuration Types

Configuration Types are groups of related data. For example, there are typically several different types of data related to items such as categories, SKU's, etc. A *Configuration Type* for items would contain all the item-related data. There may be hundreds of tables to hold different kinds of master data and each table will be assigned to a *Configuration Types*. A restaurant chain may group all the master tables related to POS data in the *Configuration Types*: "Items", "Prices", "Menus" and "Restaurant-Specific". Each *Configuration Type* contains a list of the tables that are included in the *Configuration Type* and a table can only be in one *Configuration Type*.

To determine how to divide the master tables into *Configuration Types*, the administrator should consider how the data will be applied to the different applications. For example, if the master item list and the related tables should be shared across all applications and remote sites then they can all be put into a single *Configuration Type* called *Menu Items*. Typically, remote sites are assigned to different price groups or tiers, so the tables related to pricing would be assigned to a different *Configuration Type* called "Prices", so that different sets of prices can be assigned to different remote sites. Since item information is shared by all sites but prices are only shared by a subset of the sites, they must be in different *Configuration Types*.

These *Configuration Types* are mapped to the specific application tables identified in the *Shared Table Groups*.

Configurations

A *Configuration* is a specific set of data for the tables in a *Configuration Type* such as a single price tier. For example, the "Price" *Configuration Type* may have multiple *Configurations* called "Price Tier A", "Price Tier B", and "Price Tier C" (and each tier has different prices).

Versions

A *Version* is a specific set of data for a *Configuration*. As *Configuration* data changes over time, new *Versions* of the data are created. For example, the initial *Version* of "Price Tier A" may have the price of coffee at \$1.99, but the following *Versions* change coffee prices to \$2.09 (version 2) and \$2.19 (version 3). Each *Version* has a name such as "Half Price LTO" or "July 2012 Marketing Window" to associate it with the business reason for the new *Version*.

Subscriptions

Remote sites subscribe to one *Version* for each *Configuration Type* so that the system knows which data to put in each of its tables. For example, the "Main Street" site may use the following *Versions*:

Configuration Type	Configuration	Version	Start Date
Items	Standard	July 2012	July 4, 2012
Prices	Price Tier A	Fall Marketing Window	September 1, 2012
Menus	Sweet Tea Test	Fall Marketing Window	September 1, 2012
Site-Specific	Main Street Site	New kitchen configuration	March 15, 2010

When the data needs to be changed at a remote site, the site subscribes to a different *Version* of the data. The subscription *Start Date* tells the system when to update the application tables at each site. When a user subscribes a remote site to a new *Version*, the system updates the central application tables for the site from the master data version and sends the data to the site on the start to refresh the application tables on the *Start Date*.

Configuring Master Data Management on EDM

The following pages detail how to setup Master Data Management for EDM.

Step 1 - Shared Table Groups

This is the first step to configure Master Data Management on EDM, creating *Shared Table Groups*.

You can navigate between each step using the Hierarchy Tree on the left of the page, or by choosing the desired step on the [How Does Master Data Management Work](#) page.

Introduction

Shared Table Groups are used to group related data tables that will be used by MDM to update specific areas of the enterprise solutions (such as

the price tables for the Point-Of-Sale application in the stores).

Example of Shared Table Groups for POS Data

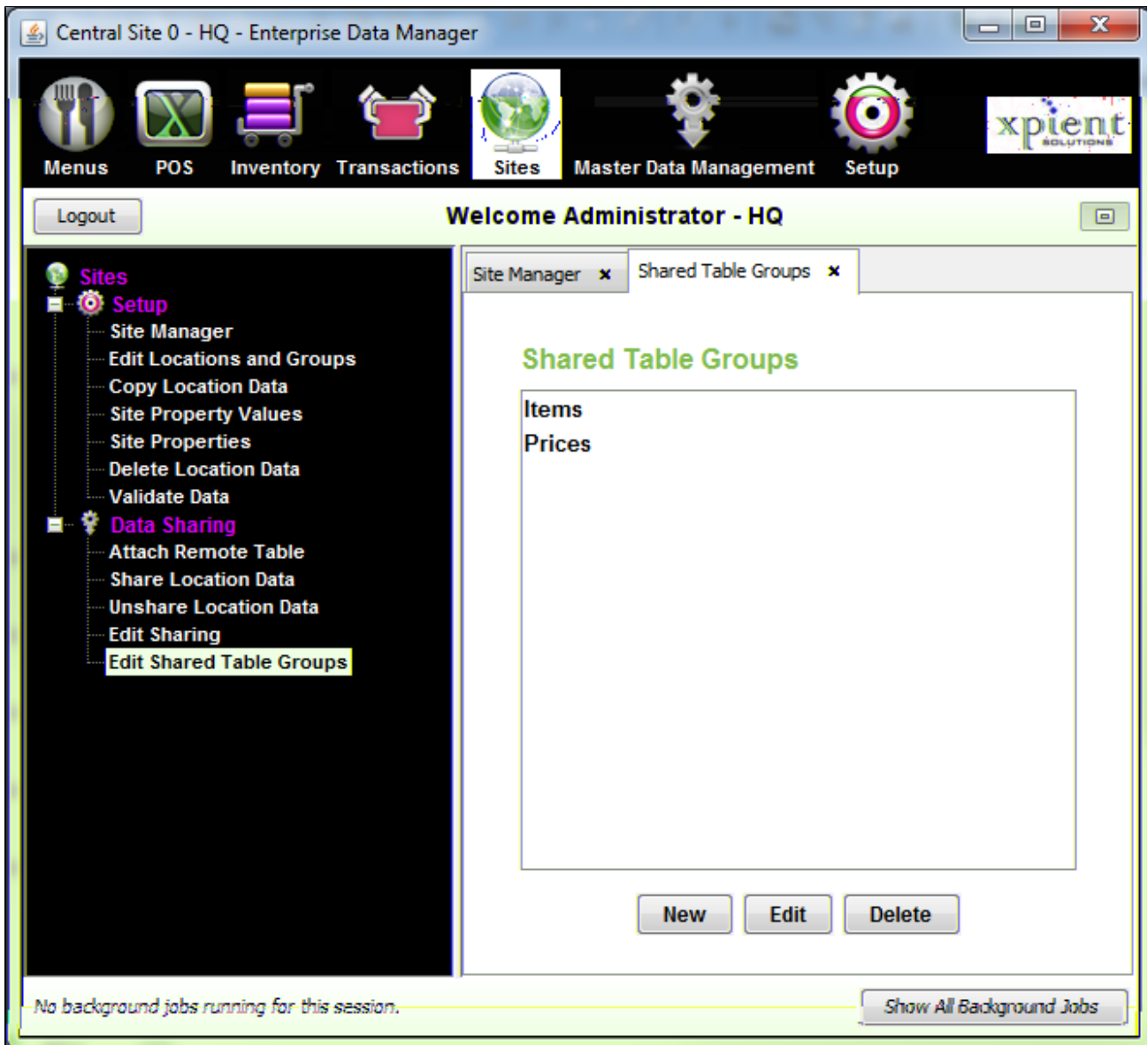
The following shared table groups are typical for POS configuration data.

Menu Items	Primary list of menu items and related data such as categories, modifiers, and attachments. Ideally, there is only one set of item data shared by all locations.
Monitor Routing	Kitchen monitors may vary from location to location, so monitor routing configurations will likely be in their own group.
Printer Routing	Receipt and expeditor printers may vary from location to location, so printer routing configurations will likely be in their own group.
Prices	Locations usually belong to price groups that may have different prices for certain items even though they share the same item list.
Menus	When new items are being tested, some locations will have the item on their menu while others do not.
Taxes	Since taxes are determined by state, county, etc, taxes have their own configuration type.
Discounts	Some companies may be able to share the same list of discounts across all locations, in which case the discount data can be included in the Menu Items group. If not, a Discounts group will be needed.
Sites	Site-specific information goes in this group. For example, in the IRIS POS system, <i>tbl_StoreInfo</i> would be in this group since two locations never share the same site settings, which include the address of the site.

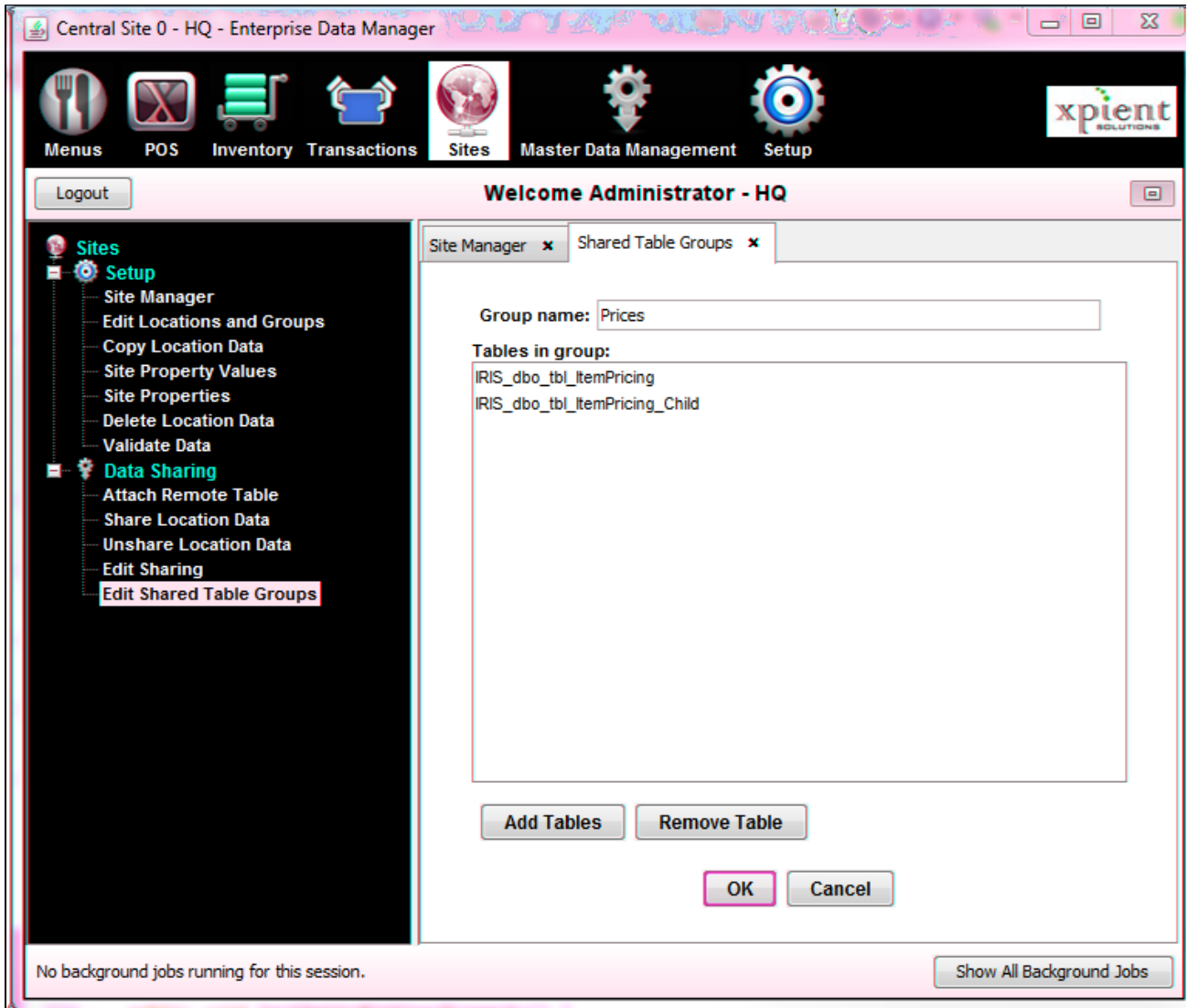
Create a Shared Table Group

You start configuring EDM for Master Data Management by creating a *Shared Table Group*.

1. Select *Sites* from the EDM menu bar.
2. From the pane on the left, expand *Data Sharing*, and then select *Edit Shared Table Groups*.



3. Click the *New* button.
4. In the *Group name* field, type the name of the *Shared Table Group*.
5. Click *Add Tables* to add the application tables that this group will update. In the following example, two IRIS price tables have been added.



6. Click OK to save your changes.

Step 2 - Configuration Types

This is the second step to configure Master Data Management on EDM, creating *Configuration Types*.

You can navigate between each step using the Hierarchy Tree on the left of the page, or by choosing the desired step on the [How Does Master Data Management Work](#) page.

Introduction

Once you have created your *Shared Table Groups*, you create the *Configuration Types*. Users create *Configuration Types* to group related data together and make it easier to assign the right data to the right sites. Each *Configuration Type* must have a single *Shared Table Group* assigned to it, along with a mapping of its dependencies and whether this *Configuration Type* supports multiple *Configurations*. For example, the *Configuration Type* "Prices" may have three *Configurations*: "Price Tier 1", "Price Tier 2", and "Price Tier 3".

Example of Configuration Types for POS Data

The following configuration types are typical for POS configuration data and usually correspond exactly with the shared table groups.

Menu Items	Primary list of menu items and related data such as categories, modifiers, and attachments. Ideally, there is only one set of item data shared by all locations.
-------------------	--

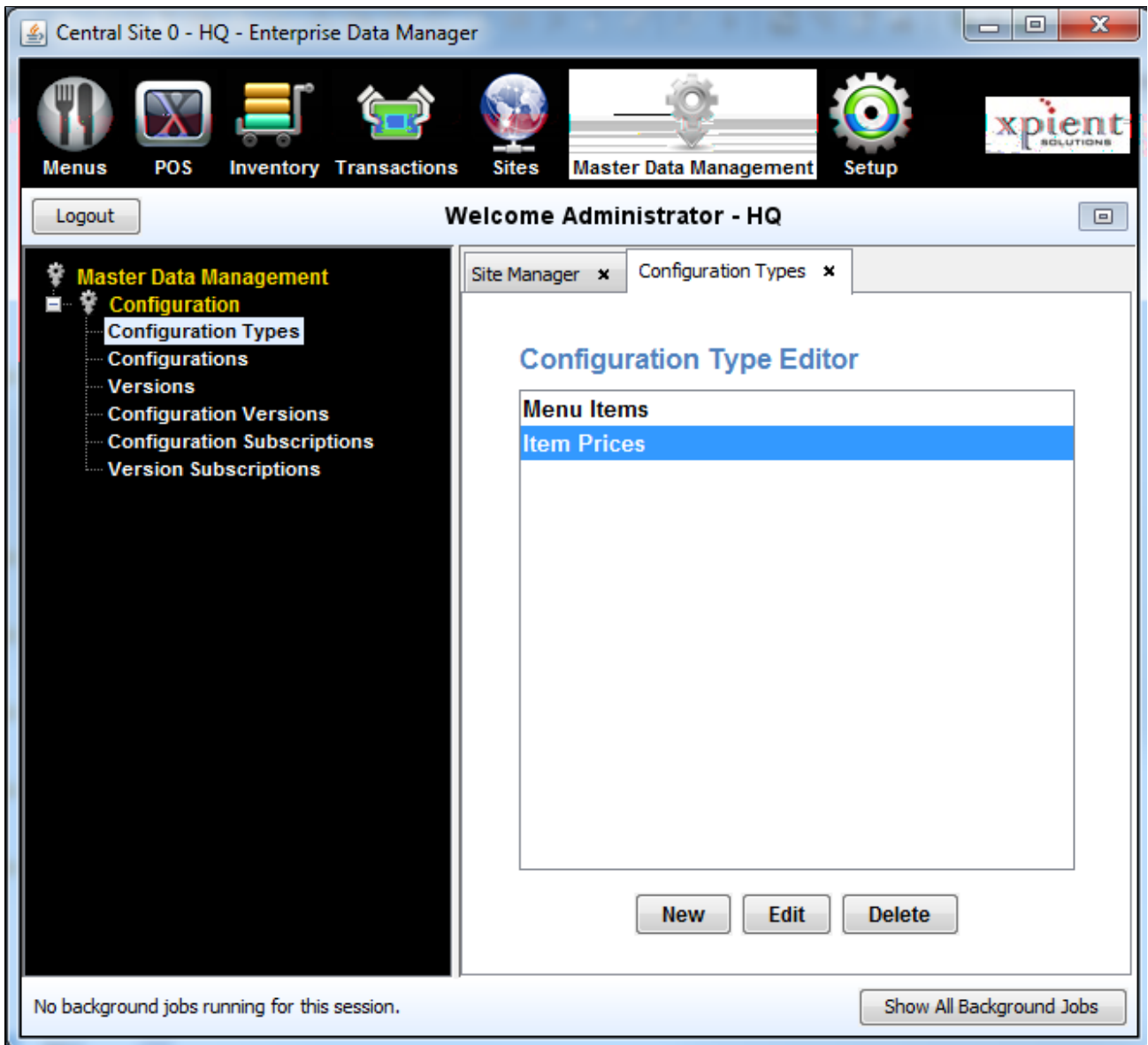
Monitor Routing	Kitchen monitors may vary from location to location, so monitor routing configurations will likely be in their own configuration type.
Printer Routing	Receipt and expediter printers may vary from location to location, so printer routing configurations will likely be in their own configuration type.
Prices	Locations usually belong to price groups that may have different prices for certain items even though they share the same item list.
Menus	When new items are being tested, some locations will have the item on their menu while others do not.
Taxes	Since taxes are determined by state, county, etc, taxes have their own configuration type.
Discounts	Some companies may be able to share the same list of discounts across all locations, in which case the discount data can be included in the Menu Items group. If not, a Discounts configuration type will be needed.
Sites	Site-specific information goes in this group. For example, in the IRIS POS system, <i>tbl_StoreInfo</i> would be in this group since two locations never share the same site settings, which include the address of the site.

The following steps describe how to:

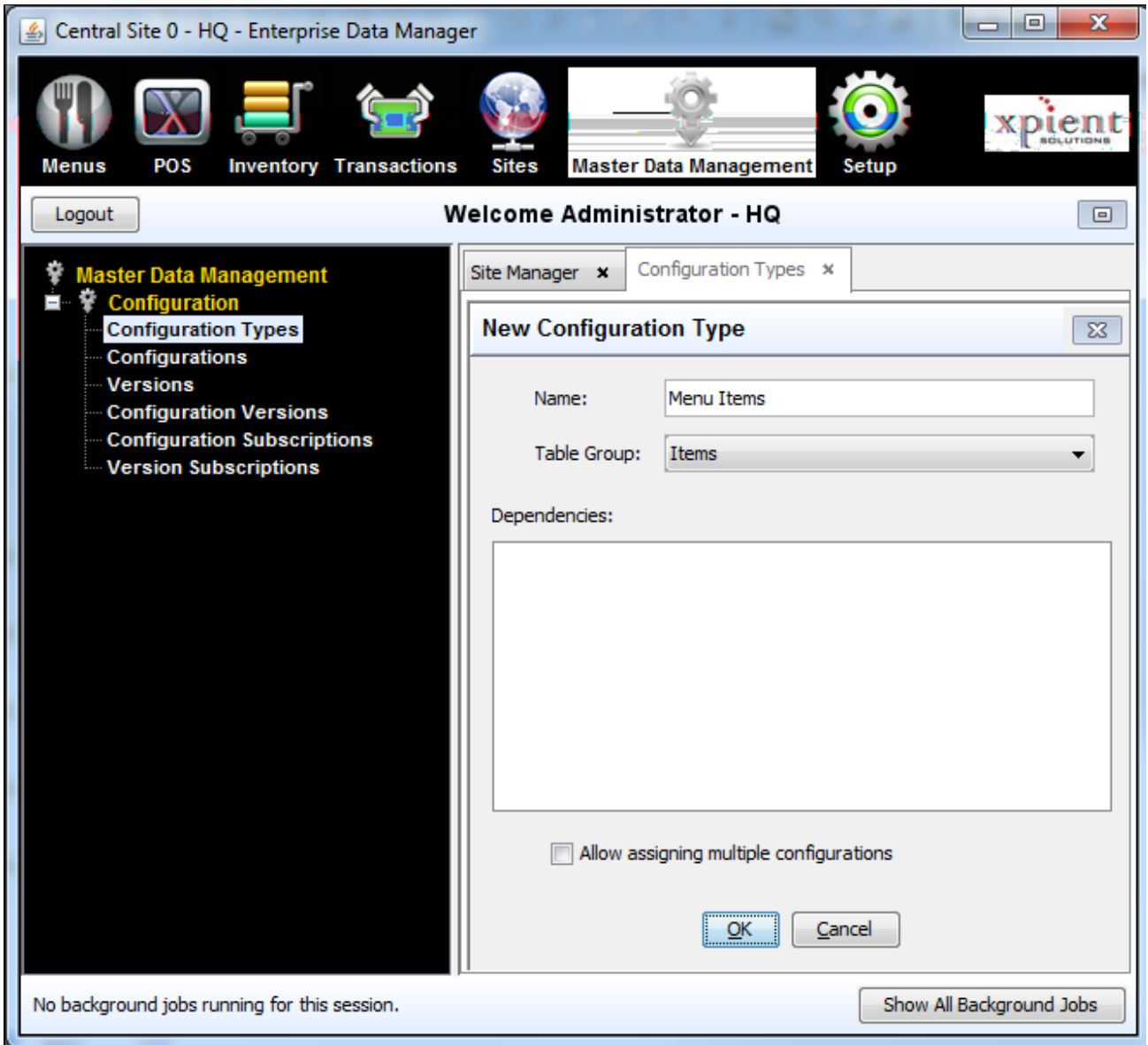
- Create a new *Configuration Type*
- Assign a *Shared Table Group* to the *Configuration Type*
- Identify dependencies for the *Configuration Type*

Create a Configuration Type

1. Select *Master Data Management* from the EDM menu bar.
2. From the pane on the left, expand *Configuration*, and then select *Configuration Types*. In the following sample screen, there are two existing *Configuration Types*: "Menu Items" and "Item Prices".



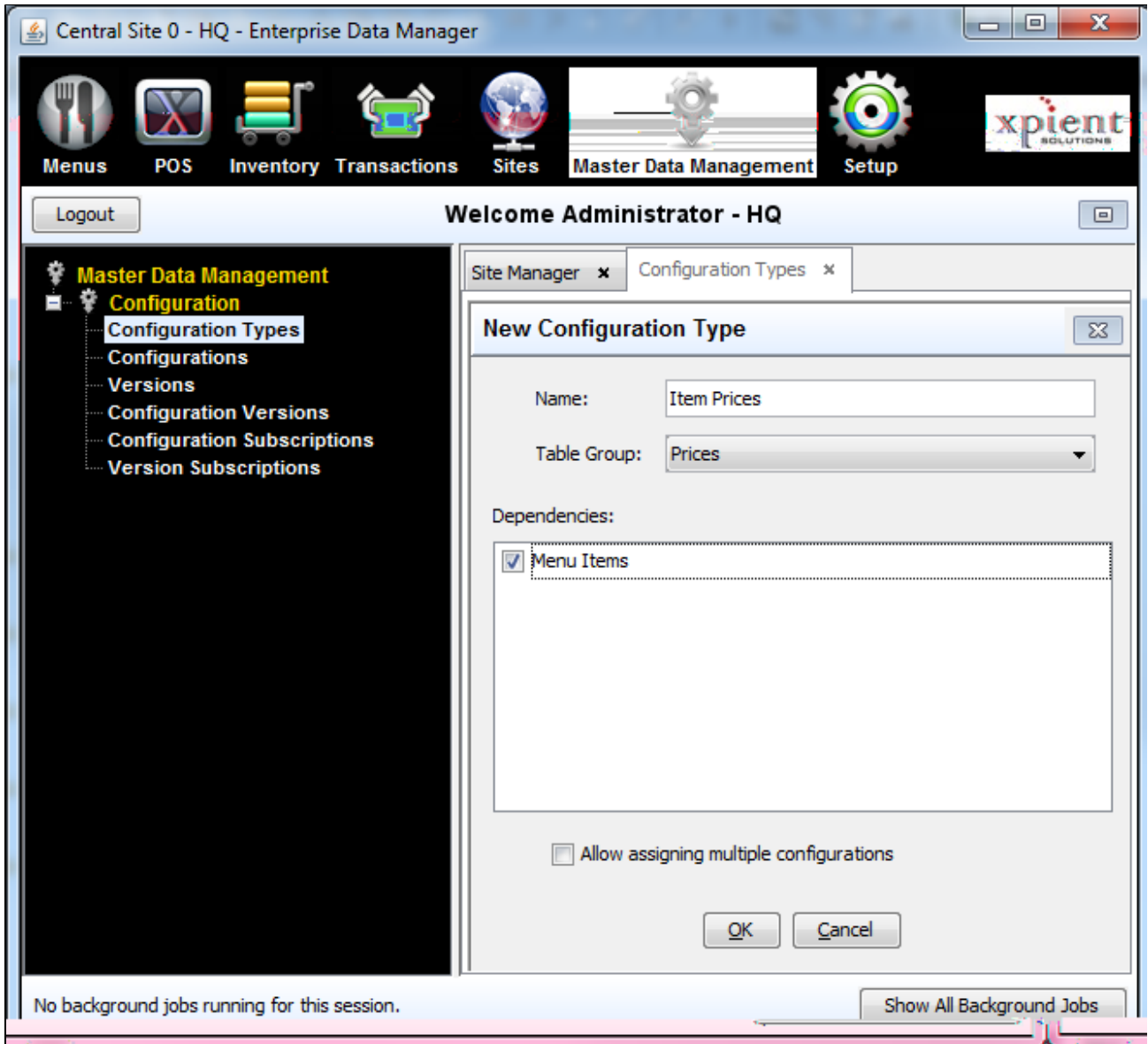
3. Click the *New* button.
4. In the *Name* field, type the name of the *Configuration Type*.
5. From the *Table Group* drop-down listbox, select the *Table Group* that this *Configuration Type* will update. The *Table Groups* that you created in Step 1 - Shared Table Groups are listed in this listbox.



6. In the *Dependencies* field, define the *Configuration Type*'s dependence on another *Configuration Type*. A *Configuration Type* dependency enforces a relationship between *Configuration Types*.

Note

Defining multiple dependencies for a *Configuration Type* is currently not supported.



7. Click **OK** to save the changes.

Note

The *Allow assigning multiple configurations* option is currently not supported.

Step 3 - Configurations

This is the third step to configure Master Data Management on EDM, creating *Configurations*.

You can navigate between each step using the Hierarchy Tree on the left of the page, or by choosing the desired step on the [How Does Master Data Management Work](#) page.

Introduction

A *Configuration* is used to identify the data that is intended to be shared. A *Configuration* is a specific instance of a *Configuration Type*. For example, the *Configuration Type* "Prices" may have three *Configurations*: "Price Tier 1", "Price Tier 2", and "Price Tier 3".

Sample Configurations for POS Data

Here are some sample configurations for the configuration types that are typical for POS configuration data.

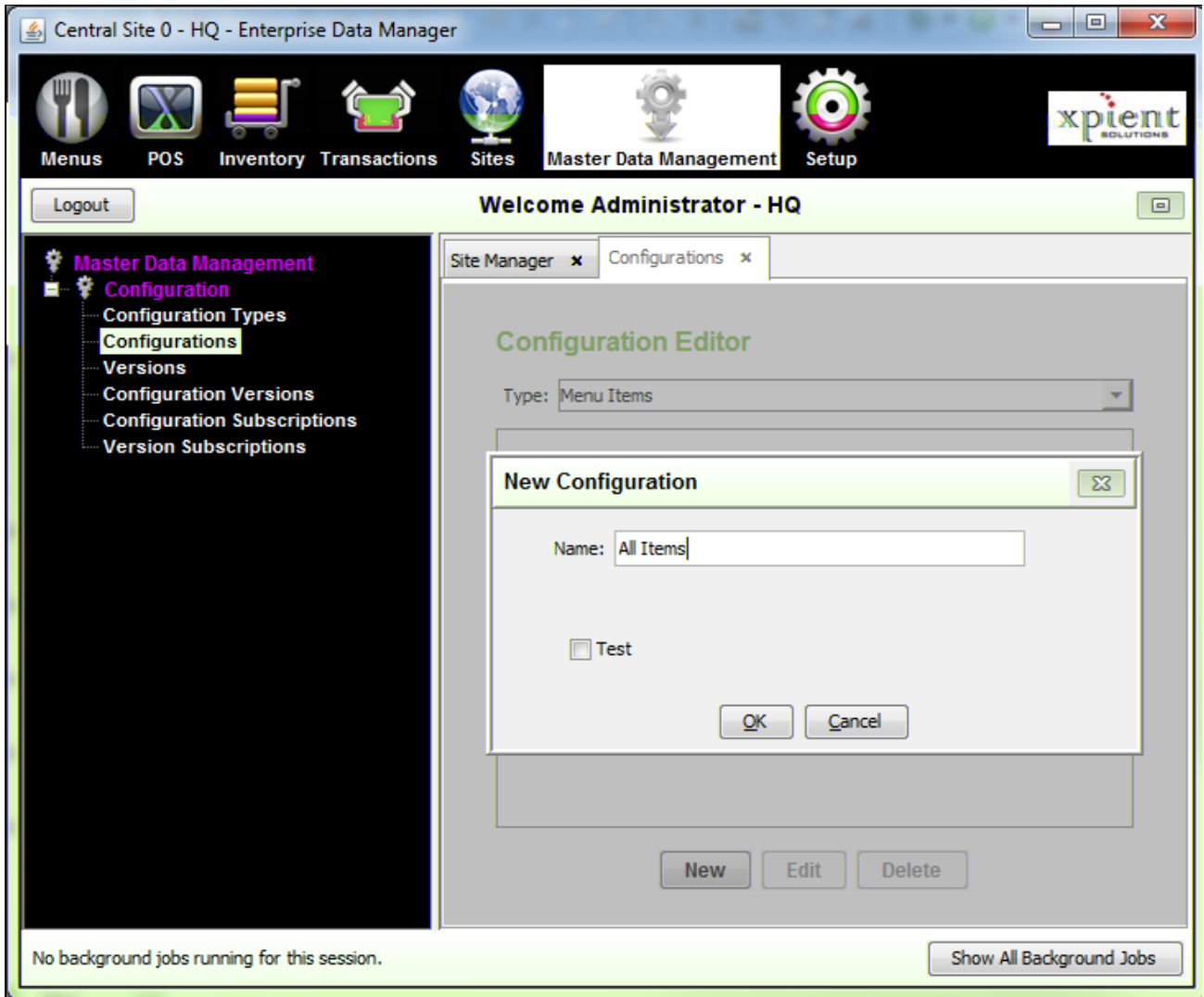
Configuration Type	Configuration	Description
Menu Items	Global	Global item data shared by all sites
Monitor Routing	3-Monitor Layout	Routing items to monitors in a site with 3 kitchen monitors
	5-Monitor Layout	Routing items to monitors in a site with 5 kitchen monitors
Printer Routing	2-Printer Layout	Routing items to printers in a site with 2 printers
	3-Printer Layout	Routing items to printers in a site with 3 printers
Prices	Price Tier A	Low price tier
	Price Tier B	Medium price tier
	Price Tier C	High price tier
Menus	Standard Menu	Menu used by most sites
	Standard Menu + Lunch Test Items	Standard menu plus some new lunch items being tested
Taxes	Austin, TX taxes	Tax tables used in Austin area
	Chicago, IL taxes	Tax tables used in Chicago area
Discounts	Corporate standard discounts	Discounts used at most corporate sites
	Franchisee Bob Smith discounts	Discounts used by franchisee Bob Smith
Sites	Site 1234	Site-specific information for site 1234
	Site 1235	Site-specific information for site 1235

Create a Configuration

The following steps describe how to create a *Configuration*.

1. Select *Master Data Management* from the EDM menu bar.
2. From the pane on the left, expand *Configuration*, and then select *Configurations*.
3. Click the *New* button.
4. In the *Name* field, type the name of the *Configuration*. In the following sample screen, you can see the creation of a new *Configuration* called *All Items*.
5. Click *OK*.

The *Test* option is not currently supported. Selecting this option has no effect.



Step 4 - Version

This is the fourth step to configure Master Data Management on EDM, creating *Versions*.

You can navigate between each step using the Hierarchy Tree on the left of the page, or by choosing the desired step on the [How Does Master Data Management Work](#) page.

Introduction

A *Version* is a set of properties that are used to identify a unique version of a *Configuration* that you intend to share. The properties of a *Version* include: its name, its purpose, the name of the person requesting the *Configuration*, and its *Planned Start Date*. A *Version* is assigned to a *Configuration* to create a *Configuration Version*. You can assign multiple *Versions* to a *Configuration* to create multiple *Configuration Versions*.

Sample Versions for POS Data

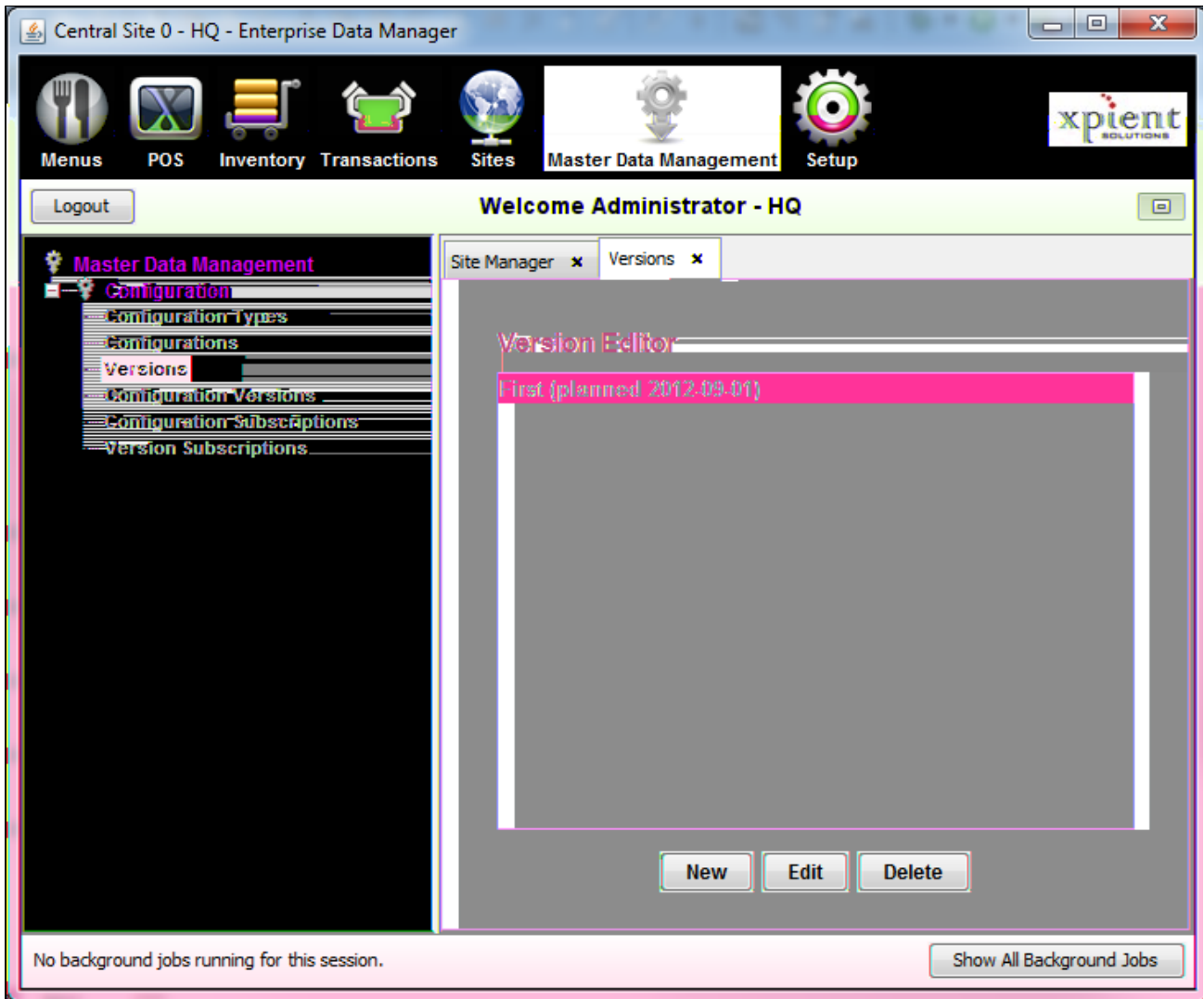
The following table shows sample versions for POS configuration data:

Name	Purpose	Requester	Planned Start Date
Marketing Window 1 - 2012	Introduce new items and prices changes	Marketing	1/1/2012
Marketing Window 2 - 2012	Introduce new items and prices changes	Marketing	4/1/2012
Fix Entree Names	Fix typos in receipt names of new items	Menu Management	4/15/2012
Chicago Tax Rate Change	Update taxes due to tax law changes	Accounting	6/1/2012

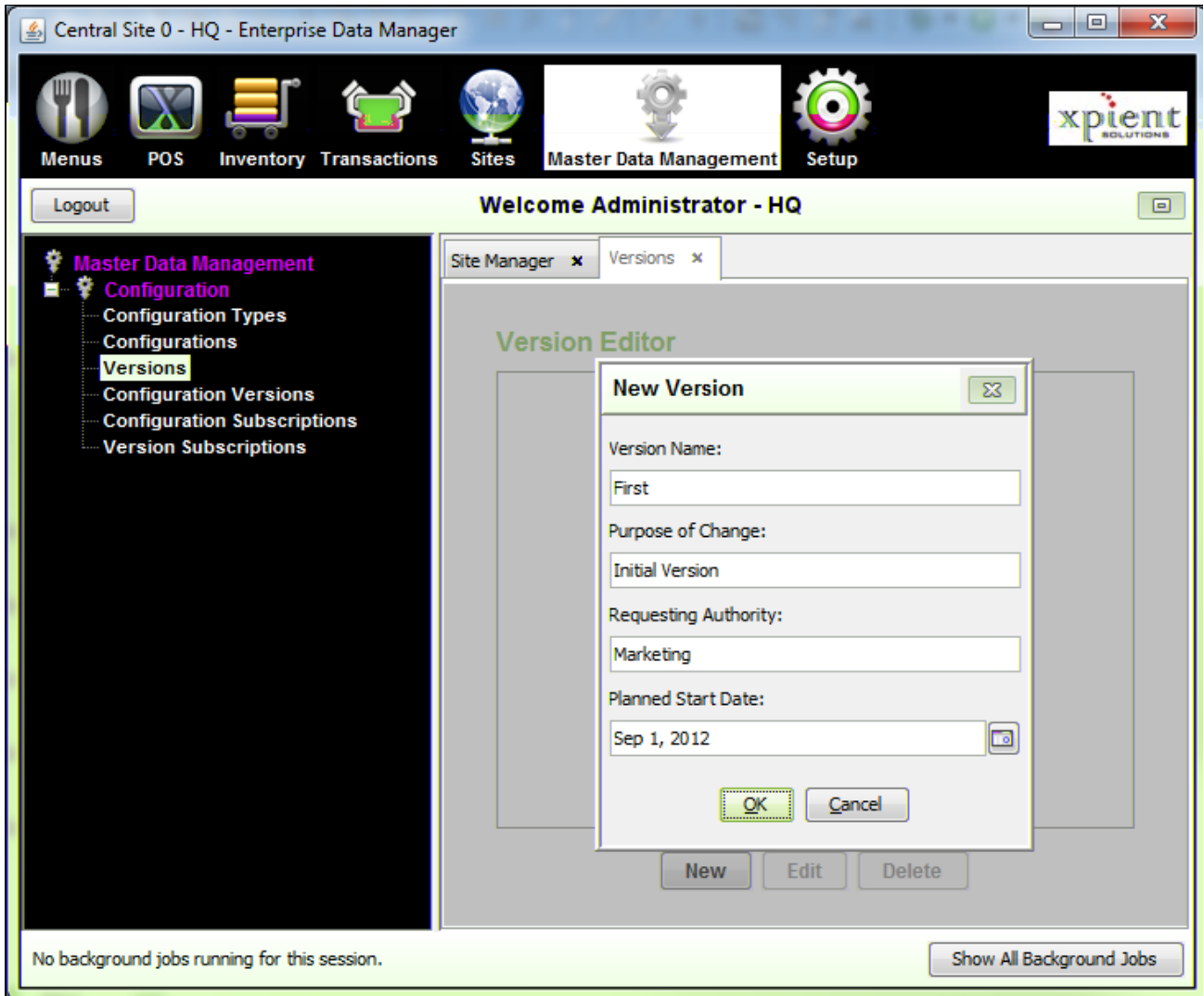
Create Version

The following steps describe how to create a *Version*.

1. Select *Master Data Management* from the EDM menu bar.
2. From the pane on the left, expand *Configuration*, and then select *Versions*.



3. Click the *New* button. The *New Version* form opens.



4. In the *Version Name* field, type the name of the *Version*.
5. In the *Purpose of Change* field, type the purpose of this *Version*.
6. In the *Requesting Authority* field, identify the party requesting the *Version*.
7. Click



to select the *Planned Start Date*.

8. Click *OK* to save your changes.

Step 5 - Configuration Versions

This is the fifth step to configure Master Data Management on EDM, creating *Configuration Versions*.

You can navigate between each step using the Hierarchy Tree on the left of the page, or by choosing the desired step on the [How Does Master Data Management Work](#) page.

Introduction

Once you have defined your *Versions* and your *Configurations*, you can now assign a *Version* to a *Configuration* to create a *Configuration Version*. You can select multiple *Configurations* for a *Version* to create multiple *Configuration Versions*.

Sample Configuration Versions

The following table shows some sample *Configuration Versions*.

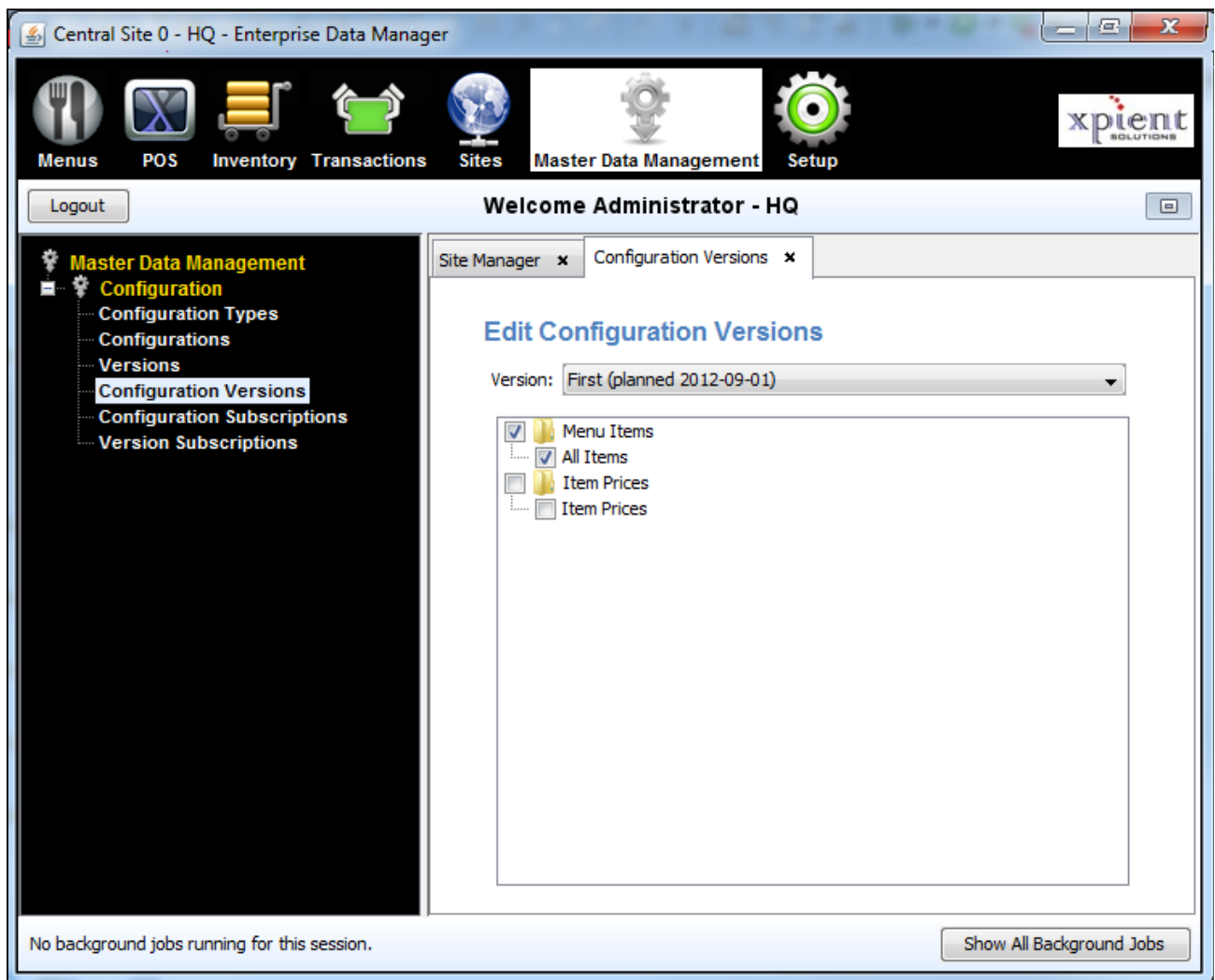
Configuration Type	Configuration	Version
--------------------	---------------	---------

Menu Items	Global	Marketing Window 1 - 2012
		Marketing Window 2 - 2012
		Fix Entree Names
Prices	Price Tier A	Marketing Window 1 - 2012
		Marketing Window 2 - 2012
	Price Tier B	Marketing Window 1 - 2012
		Marketing Window 2 - 2012
Taxes	Chicago, IL taxes	Chicago Tax Rate Change

Create Configuration Versions

The following steps describe how to create *Configuration Versions*.

1. Select *Master Data Management* from the EDM menu bar.
2. Expand *Configuration*, and then select *Configuration Versions*.
3. Select the desired *Version* from the *Version* drop-down listbox.
4. Select the check box next to the *Configurations* that you want to associate with the selected *Version*.



Step 6 - Configuration Subscriptions

This is the sixth step to configure Master Data Management on EDM, creating Configuration Subscriptions.

You can navigate between each step using the Hierarchy Tree on the left of the page, or by choosing the desired step on the [How Does Master Data Management Work](#) page.

Introduction

Subscriptions are persistent relationships that exist between restaurants (also called cafes) and configurations. When a cafe subscribes to a configuration, that cafe will receive future configuration updates (versions) as they are created and deployed in EDM. Some subscriptions can include multiple configurations of the same type. This depends upon the *Configuration Type*.

Sample Configuration Subscriptions

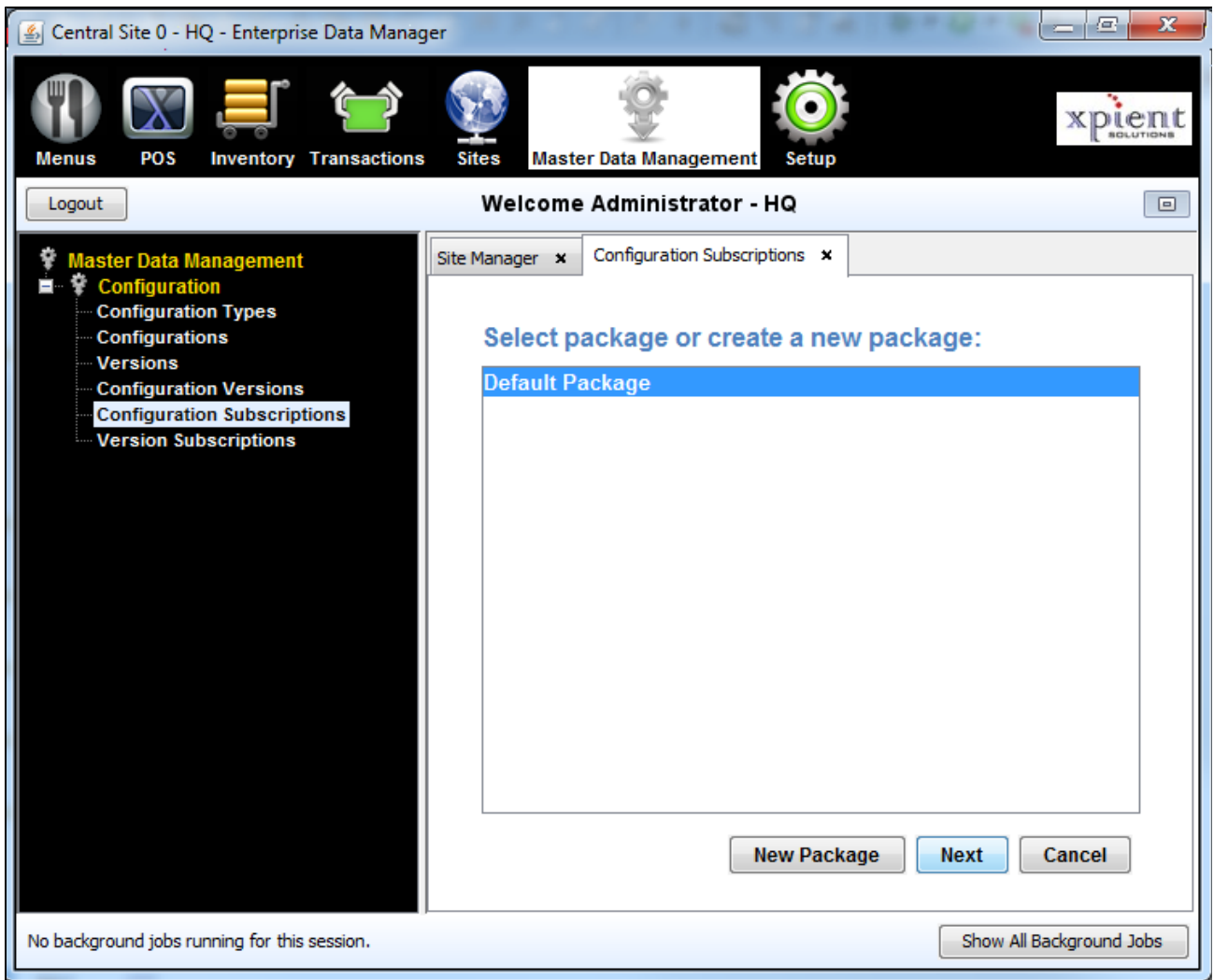
Subscribing sites to configurations is optional and is normally not needed. Some companies may find it useful to document the intended configurations for sites that are not open yet. Subscribing a site to a configuration does not deploy any data to the site. Data is deployed when a site subscribes to a specific version of the configuration.

Site	Configuration Type	Configuration
6789 - Elm St.	Menu Items	Global
	Monitor Routing	5 - Monitor Layout
	Printer Routing	3 - Printer Layout
	Prices	Price Tier C
	Menus	Standard Menu
	Taxes	Austin, TX taxes
	Discounts	Corporate standard discounts
	Sites	Site 6789

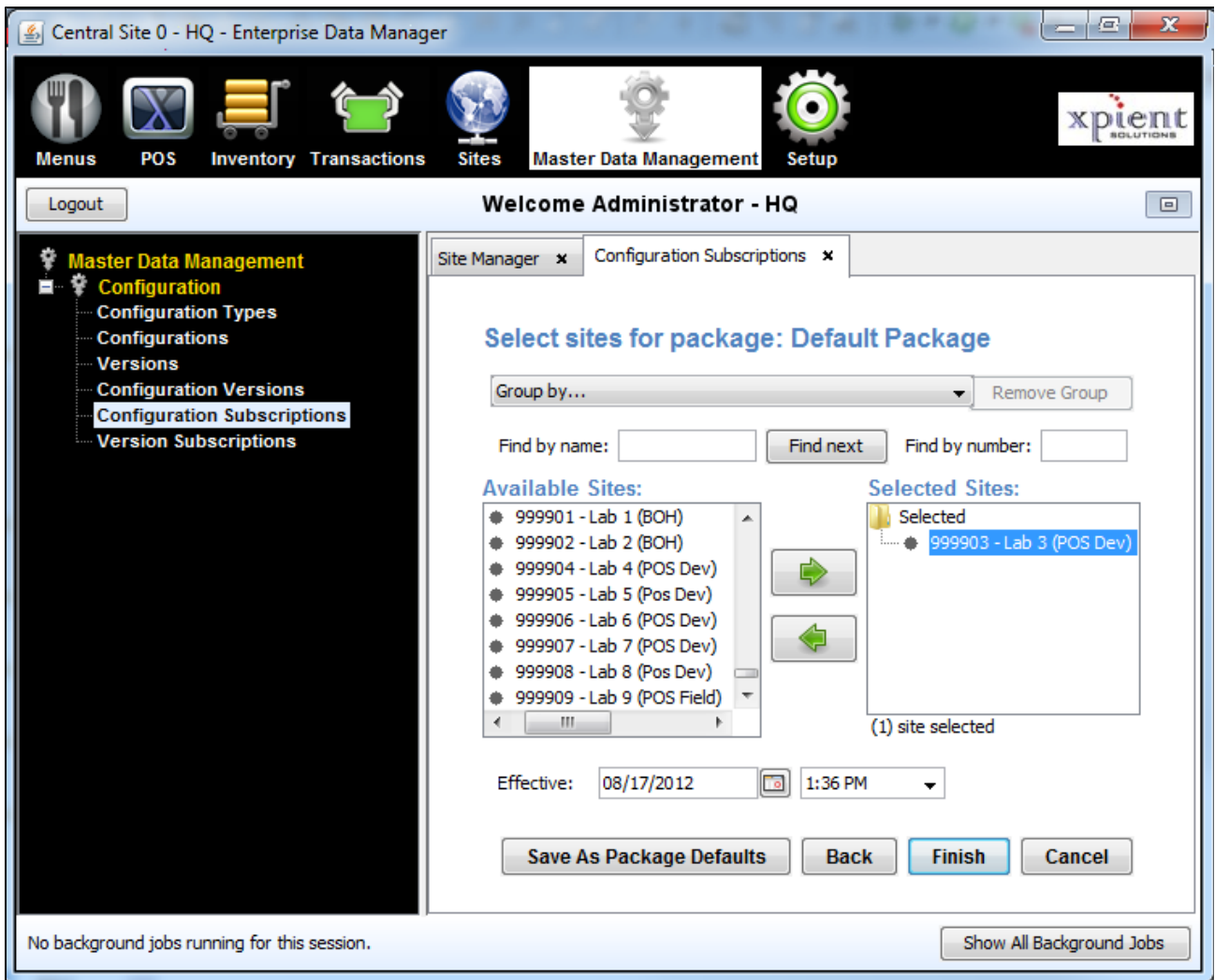
Create Configuration Subscriptions

The following steps describe how to create *Configuration Subscriptions*.

1. Select *Master Data Management* from the EDM menu bar.
2. From the pane on the left, expand *Configuration*, and then select *Configuration Subscriptions*.



3. Select the desired package from the list, and then click *Next* -OR- click *New Package* to create a new package. The *Select sites for package* screen opens.



4. From the *Group By* drop-down listbox, select the method for grouping the information.
5. From the *Available Sites* field, select the sites (restaurants) for the package, and then click



to move the sites to the *Selected Sites* field.

To remove a site from the *Selected Sites* field, select the site, and then click



You can search for sites by name or number using the *Find by name* and the *Find by number* fields, respectively.

6. Click



to select the effective date for the update. Select the time for this update from the drop-down listbox.

7. Click *Finish*. The *Subscribe to Configurations* screen opens.



- From the *Available* field, select the *Configuration* for which you want to create a *Subscription*, and then click



to move it to the *Subscribed* field.

To remove a *Configuration* from the *Subscribed* field, select it, and then click



Step 7 - Version Subscriptions

This is the seventh step to configure Master Data Management on EDM, creating *Version Subscriptions*.

You can navigate between each step using the Hierarchy Tree on the left of the page, or by choosing the desired step on the [How Does Master Data Management Work](#) page.

Introduction

During the normal course of business, *Configurations* will be change over time. EDM end users will organize these changes into *Configuration Versions*, which define the Data in a *Configuration* for a specific period of time. Existing Restaurants (or Cafes) that subscribe to a particular *Configuration* receive new *Configuration Versions* once they are updated on EDM.

Sample Version Subscriptions

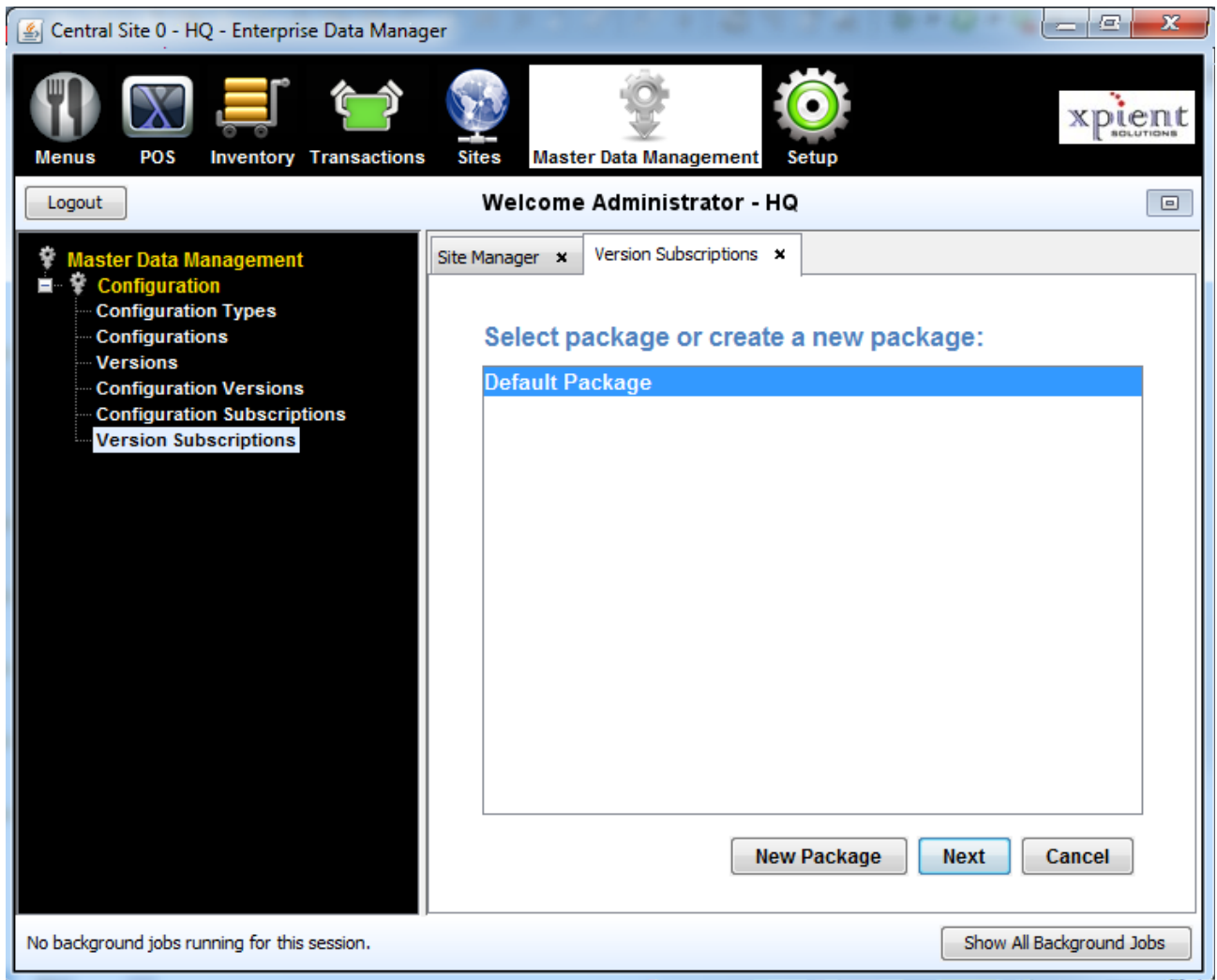
The following table shows some sample *Version Subscriptions*. The table contains all the information needed to determine which *Configuration Version* was active for each *Configuration Type* for a site on any given date.

Site	Configuration Type	Configuration Version	Start Date	
1234 - Main Street	Menu Items	Global - Marketing Window 1 - 2012	1/1/2012	
		Global - Marketing Window 2 - 2012	4/3/2012	
	Monitor Routing	3 - Monitor Layout - Initial	6/1/2011	
	Printer Routing	2 - Printer Layout - Initial	6/1/2011	
	Prices	Price Tier B - Marketing Window 1 - 2012	1/1/2012	
		Price Tier B - Marketing Window 2 - 2012	4/3/2012	
	Menus	Standard Menu - Marketing Window 1 - 2012	1/1/2012	
		Standard Menu - Marketing Window 2 - 2012	4/3/2012	
	Taxes	Chicago, IL taxes - Initial	6/1/2011	
		Chicago, IL taxes - Chicago Tax Rate Change	2/1/2012	
	Discounts	Corporate standard discounts - Initial	6/1/2011	
	Sites	Site 1234 - Initial	6/1/2011	
	1235 - Oak St.	Menu Items	Global - Marketing Window 1 - 2012	1/2/2012

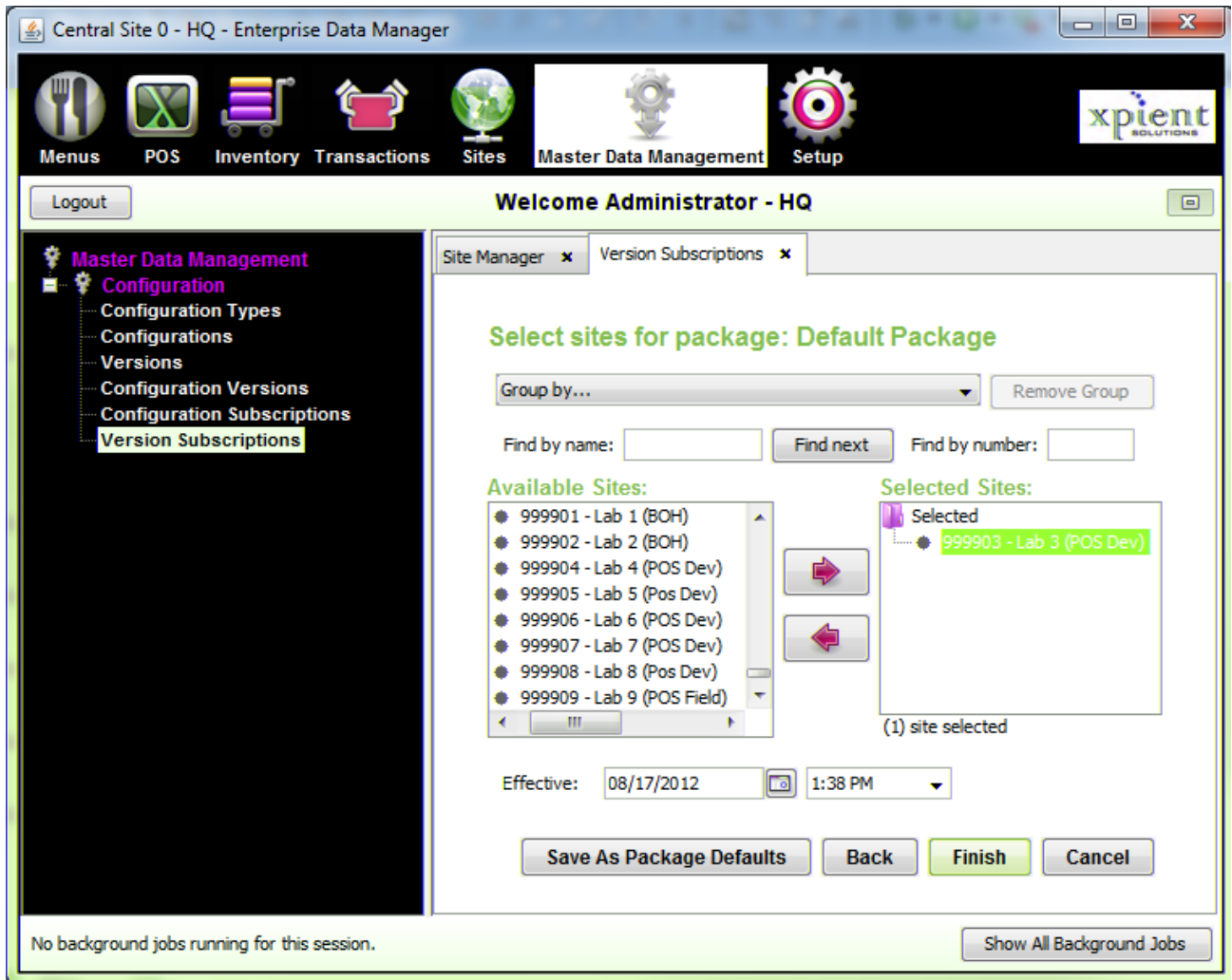
Create Version Subscriptions

The following steps describe how to create *Version Subscriptions*.

1. Select *Master Data Management* from the EDM menu bar.
2. From the pane on the left, expand *Configuration*, and then select *Version Subscriptions*.



3. Select the desired package from the list, and then click *Next* -OR- click *New Package* to create a new package. The *Select sites for package* screen opens.



4. From the *Group By* drop-down listbox, select the method for grouping the information.
5. From the *Available Sites* field, select the sites (restaurants) for the package, and then click



to move the sites to the *Selected Sites* field.

To remove a site from the *Selected Sites* field, select the site, and then click



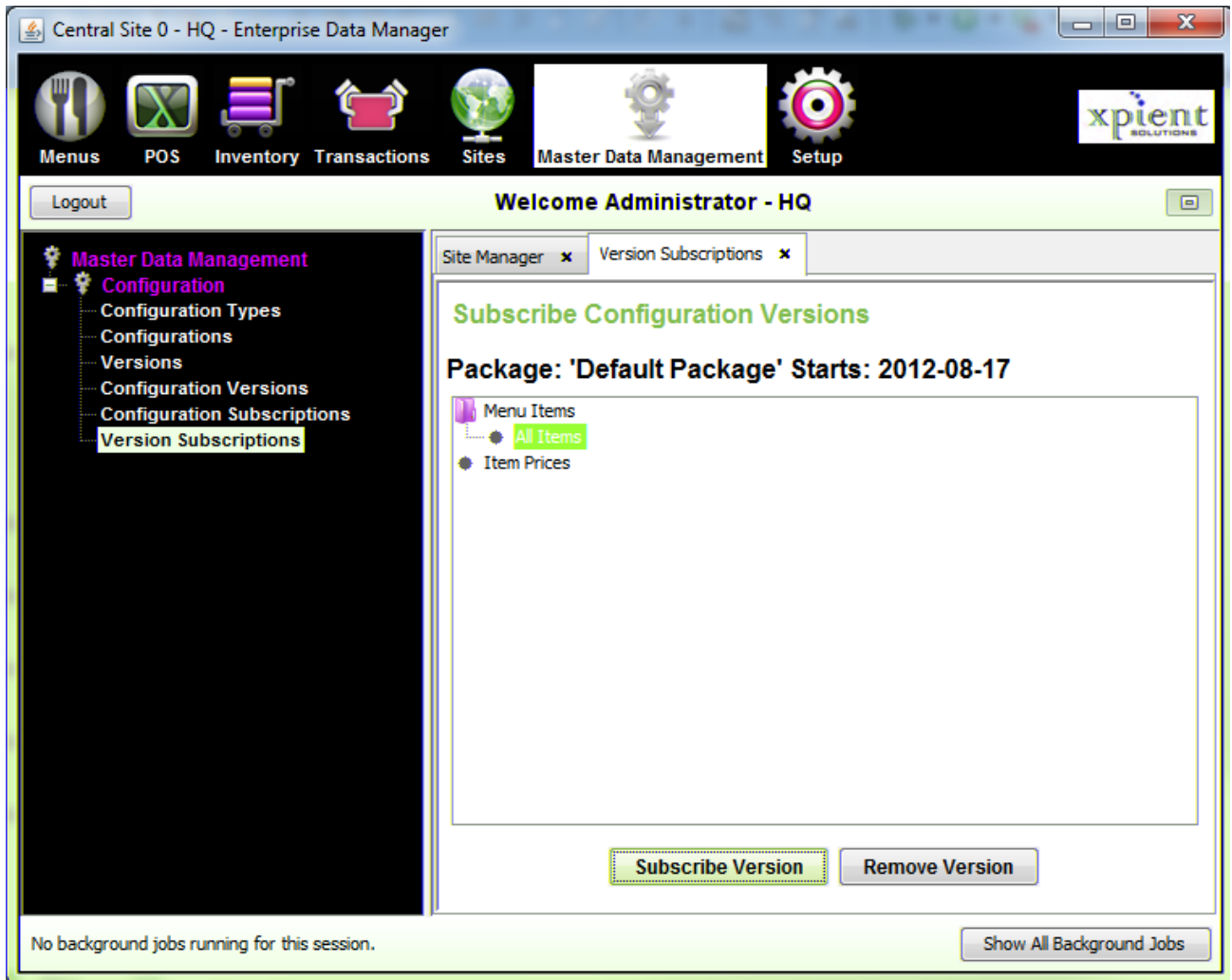
You can search for sites by name or number using the *Find by name* and the *Find by number* fields, respectively.

6. Click

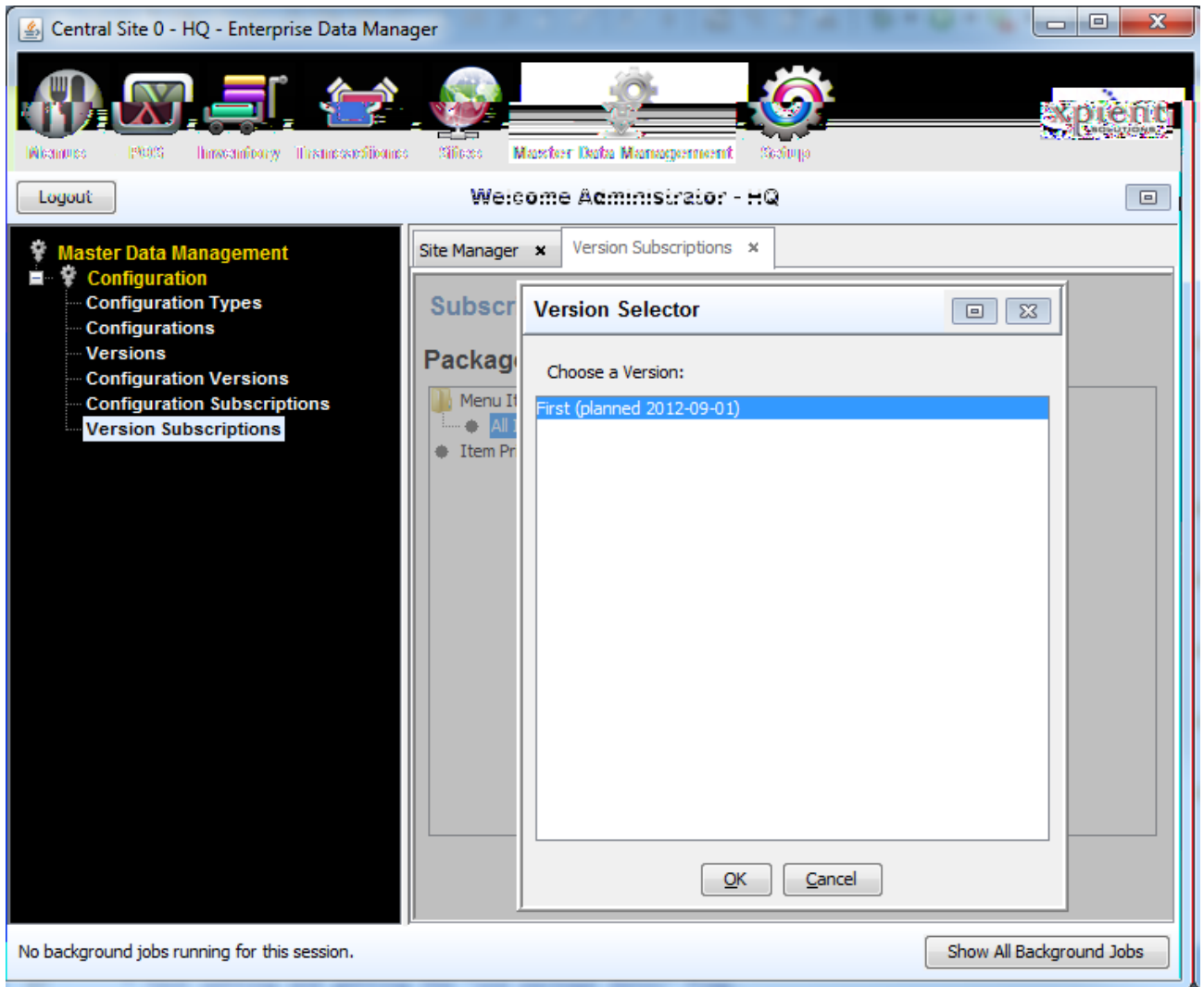


to select the effective date for the update. Select the time for this update from the drop-down listbox.

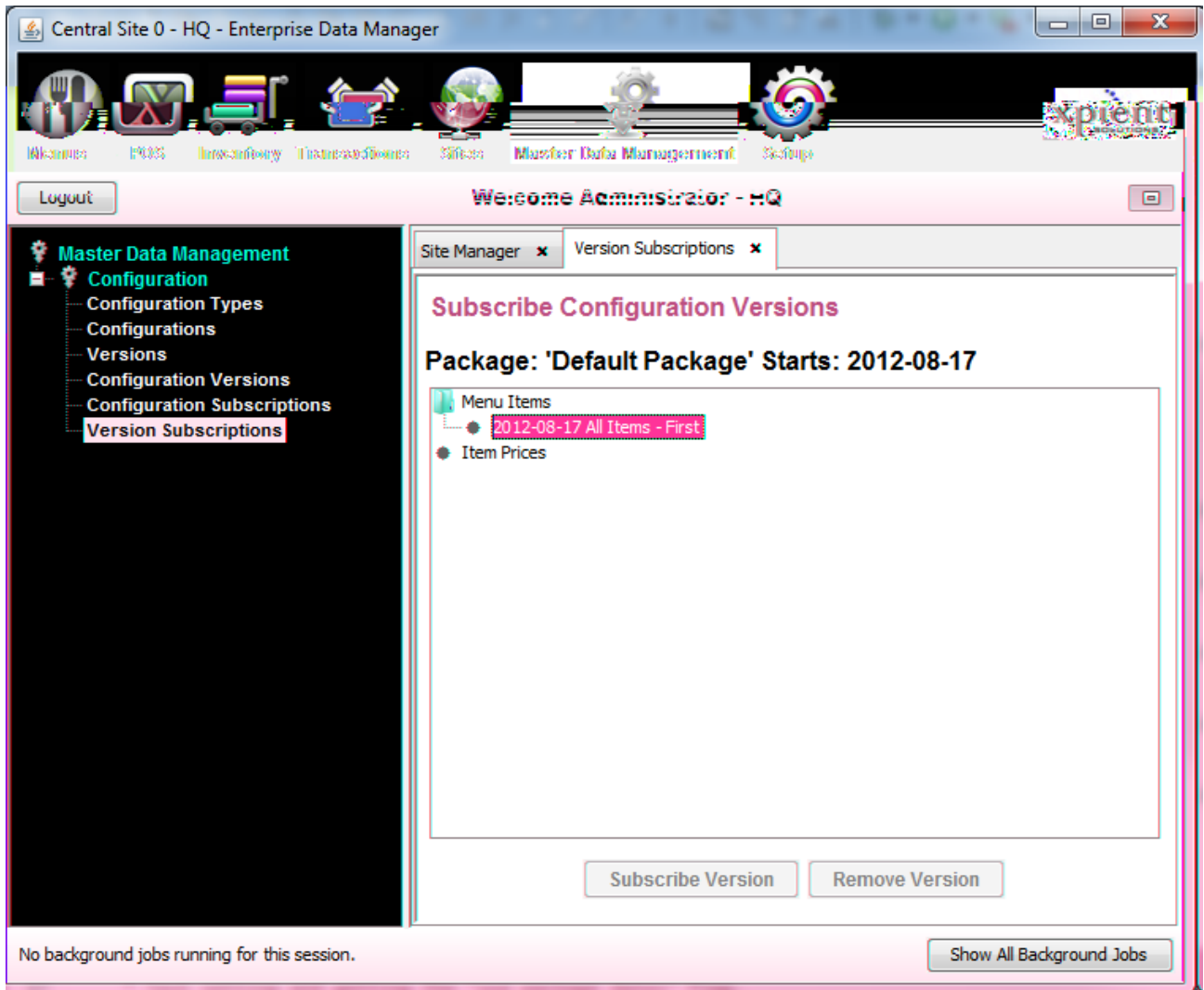
7. Click *Finish*. The *Subscribe Configuration Versions* screen opens.



8. Select the *Configuration* from the list, and then click *Subscribe Version*. The *Version Selector* window opens.



9. Select the *Version* of the *Configuration* to subscribe to the selected Restaurants (or Cafes), and then click *OK* to return to the *Subscribe Configuration Versions* screen.



Once the package is committed, the sites that were selected for the package will receive the data that is determined by the *Version* of the *Configuration*.

Note

This Data Site will not have any data associated with it at this point. You will need to "seed" the data to associate with the Data Site.

Step 8 - Identify Sites to Deploy

The administrator can set up multiple deployment types to test master data changes before releasing the changes to production. For example, the administrator may want to first deploy master data changes to a test system, then to a QA system when the testing is complete, and finally to production when QA testing is complete.

The administrator can also identify specific sites to deploy to each deployment type by defining properties for those sites.

Deploy to Specific Sites

To deploy to specific sites, take the following steps for each deployment type:

1. In EDM, select *Sites>Site Properties*.
2. Click + to add a new site property. Type a unique property ID and property name, such as *Test Site*.
3. Click + to add a new site property. Type a unique property ID and property name, such as *QA Site*.
4. Click + to add a new site property. Type a unique property ID and property name, such as *Production Site*.
5. Close the *Site Properties* form.
6. Select *Sites>Site Property Values*.
7. Select the *Test* sites.
8. Type Y in the *Test Site* property value. Click *Save* and close the form.

9. Select *Sites>Site Property Values*.
10. Select the *QA* sites.
11. Type *Y* in the *QA Site* property value. Click *Save* and close the form.
12. Select *Sites>Site Property Values*.
13. Select the *Production* sites.
14. Type *Y* in the *Production Site* property value. Click *Save* and close the form.

Now when you call the deployment or version assignment scheduling functions, you can pass in the site property and value that is used to determine which sites are affected. For example, when deploying to test sites, you would pass in the property name *Test Site* and the property value *Y*.

Selecting the Seed Data

This page explains how to select the "seed" data for your initial Data Site, as described on [Step 7 - Version Subscriptions](#).

Click [here](#) to return to the main EDM Master Data Management page.

About the Seed Data Selection Process

Once you have configured [Master Data Management](#), you will need to "seed" the data to associate with the Data Site you created in [Step 7 - Version Subscriptions](#).

By default, the name of the Data Site is the (*Configuration* name) + (*Version* name) you selected. For example, if you selected the "All Items" *Configuration* and the "Initial" *Version*, then the default name for the Data Site is "All Items - Initial".

The process of seeding data to the Data Site typically involves running queries against the database, such as IRIS or Aloha, and directing the data that is returned into the master tables that you selected when you created the [Shared Table Group](#) with which the *Configuration Type* is associated.

This will be the initial version of the data. When you commit the package that you created in [Step 7](#), this data is made available to the site that you included in the package.

Please contact your *XPIENT Solutions - Professional Services* representative for assistance with this process.

When you create the next version of data (which will have its own Data Site), you can use EDM's *Copy Location Data* functionality to copy the data from the "Initial" *Version's* Data Site to the new *Version's* Data Site. In a future implementation, this process of copying data from one Data Site to another will be automated.

Updating Application Data

This page explains how the Master Data on EDM must be set to update the application tables in the restaurants or other enterprise systems.

Click [here](#) to return to the main EDM Master Data Management page.

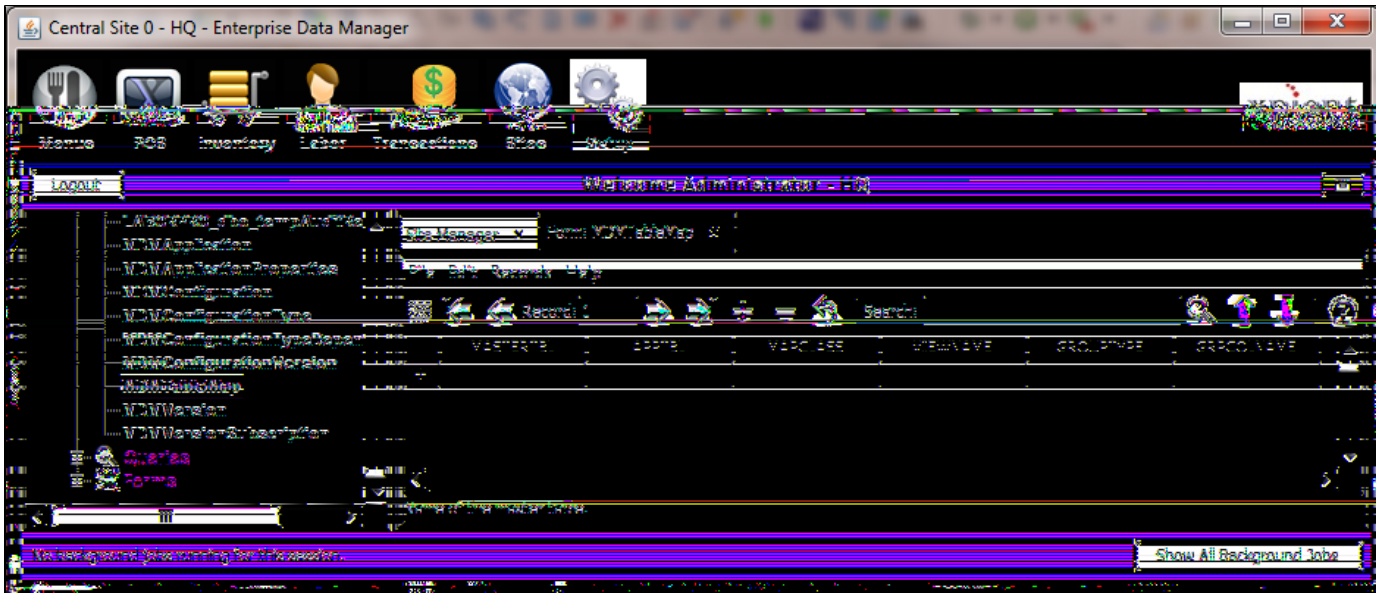
TableMap Table

This table defines how EDM updates each application table, from the data in the Master Data tables. The process is simple:

- The EDM administrator creates a view of the master data tables that has the same columns as the application table
- EDM compares the data in the master view with the data in the application table to determine what changes to make

Editing the TableMap Table

To edit the table map, navigate on EDM to the *MDMTableMap* page (inside *Setup - Applications - Tables*). Here is an example of the screen:



The following table describes the columns.

MASTERTBL	The name of the master table that is primarily used to update the application table
APPTBL	The application table whose data is to be updated from the master data
MAPCLASS	NOT CURRENTLY USED
VIEWNAME	The name of the view of the master data that has the same columns as the application table.
GROUPTYPE	Used for custom mappings where the application table contains data for more than one site. For example, if your online ordering system contains groups of data for PRICE_TIER and TAX_GROUP then you would enter PRICE_TIER as the group type for the price application table.
GROUPNAME	The name of the group column in the application table. For example, if the application's price table contains a group column called ORG_ID where the system should put the group number, then enter ORG_ID in this column.

Note

There should be one row in this table for each application table that needs to be updated.

Updating the different data tables

This page explains how the Master Data on EDM must be set to update the application tables in the restaurants.

Click [here](#) to return to the main EDM Master Data Management page.

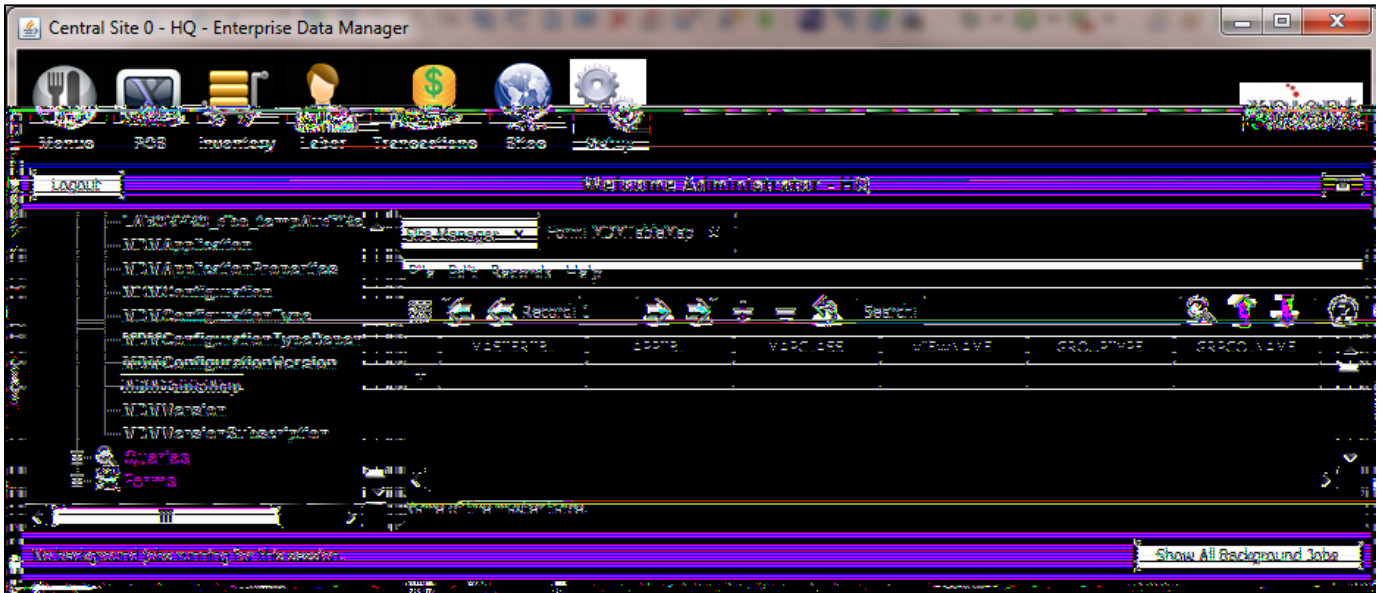
TableMap table

This table defines how EDM updates each application table, from the data in the Master Data tables. The process is simple:

- The EDM administrator creates a view of the master data tables that has the same columns as the application table
- EDM compares the data in the master view with the data in the application table to determine what changes to make

Editing the TableMap table

To edit the table map, navigate on EDM to the *MDMTableMap* page (inside *Setup - Applications - Tables*). Here is an example of the screen:



The following table describes the columns.

MASTERTBL	The name of the master table that is primarily used to update the application table
APPTBL	The application table whose data is to be updated from the master data
MAPCLASS	NOT CURRENTLY USED
VIEWNAME	The name of the view of the master data that has the same columns as the application table.
GROUPTYPE	Used for custom mappings where the application table contains data for more than one site. For example, if your online ordering system contains groups of data for PRICE_TIER and TAX_GROUP then you would enter PRICE_TIER as the group type for the price application table.
GROUPNAME	The name of the group column in the application table. For example, if the application's price table contains a group column called ORG_ID where the system should put the group number, then enter ORG_ID in this column.

Note

There should be one row in this table for each application table that needs to be updated.

Package Editor

Package statuses are defined in the *MDMPKGSTS* database table. The *Package Editor* is used to edit the following properties about packages used for [Master Data Management](#).

Property	Description
Name	The name of the package (this is required)
Description	A description of the package
Change Sponsor	The person who sponsored the change
Change Manager	The person who manages the change
Version	The version of the data in the package
Plan Date	The date to deploy the package

MDM for EDM Users

For users who already use EDM to manage the data for all their remote sites, [Master Data Management](#) can be implemented in two phases to minimize the impact on existing personnel and processes.

MDM Implementation Phase 1

Phase 1 uses the EDM central database for all master data. Users can add objects and attributes to the central database to support the data needs of additional applications. For example, users can add a table to the POS data tables called *MobileItemProperties* to store the image URL and mobile ordering item description for each of the items in the item table.

Benefits

- All data objects and attributes that are needed to support multiple applications can be managed
- Eliminates duplicate data entry when more than one application needs the same data object
- Current data can be published for external applications to consume
- All applications that use the master data can stay synchronized
- Existing data management processes remain the same except for managing the additional data objects and attributes

MDM Implementation Phase 2

Phase 2 moves all master data to a completely new set of tables. To support all the applications supported by MDM, the new tables can include any desired columns and relations.

Benefits

- Master table can have any desired columns and relationships instead of using the central table structures, which must match the remote table structures
- Restructuring tables can make it possible to share data that could not be shared using the existing table structures if there were column differences
- Data management is simplified using versioned data

Subscribing Multiple Configurations to a Site

The following describes subscribing multiple configurations of the same type to a site. This topic is related to [Master Data Management](#).

Certain configuration types may be flagged to a site to subscribe to multiple configurations of that type. For example, when defining item availability for a site, one configuration could contain the Core items and another configuration could contain some additional test items and even a "not available" flag on specific Core items. Then, instead of having multiple large configurations for every possible combination of Core and test items, a site can subscribe to the Core configuration and also subscribe to whatever test configurations it is participating in.

Combining the data from the configurations may require different algorithms for different tables or customers. For example, an item availability table could contain an item number and a boolean availability flag and when two availability configurations are combined, only the items that exist in one or both tables and do not have the availability flag set to false in either table are included in the combined data set. Other algorithms might be the union or intersection of the groups.

Users will be in control of how the data is combined across multiple configurations by creating the view for each table in the configuration type to combine the data for all data sites in the "Configuration Combinations" table (MDMCFGCOMB) which contains a list of CDMLOCID's that are to be combined by the view.

For example, the Item Availability master table shown below identifies which items are available in different configurations of the configuration type. CDMLOCID 1 is the Core configuration which only contains rows for items that are generally available at all sites and does not contain records for test items. CDMLOCID 2 is the Test A configuration which marks core item 2 as "not available" and adds test item 4. CDMLOCID 3 is the Test B configuration which marks core item 3 as "not available" and adds test item 5.

Item Availability Master Table

CDMLOCID	ItemNum	Available
1	1	1
1	2	1
1	3	1

2	2	0
2	4	1
3	3	0
3	5	1

CDMLOCID 1 represents the Core configuration of items that are available to all sites by default.

CDMLOCID 2 represents Test A configuration that removes item 2 and adds item 4

CDMLOCID 3 represents Test B configuration that removes item 3 and adds item 5

If a site is assigned to Core and Test B, the system will write rows for CDMLOCID 1 and 3 in the MDMCFGCOMB table then open the view to get the combined result set.

Configuration Combinations table (MDMCFGCOMB)

CDMLOCID
1
3

A sample view that combines all the items from the configuration data sites referenced in the MDMCFGCOMB table such that all items from each data site that don't have a 0 in the Available column at any of the data sites are in the result set is shown below:

```
-- Combine Availability row for all data sites included in MDMCFGCOMB
table
select Availability.ItemNum
  from EDMStoreHQ.dbo.MDM_dbo_Availability Availability,
EDMStoreHQ.dbo.MDMCFGCOMB CombineConfigs
 where Availability.CDMLOCID = CombineConfigs.CDMLOCID
 group by Availability.ItemNum
 having MIN(Availability.Available) = 1
```

If a site is assigned to Core and Test B, the system will write rows for CDMLOCID 1 and 3 in the MDMCFGCOMB table and the view would return items 1, 2, and 5. Item 3 would not be included because it was marked as not available in configuration 3, item 4 would not be available because it did not exist in the available list of either configuration 1 or 3.

Deployment Groups

Deployments may not be able to use existing configuration site ID's as group ID's when deploying multiple configurations to a single site because there may be more combinations of configurations subscribed than sites. For example, if the Availability configuration type has configurations Core, Test A, and Test B, site 1 may just use Core, site 2 may use Core and Test A, site 3 may use Core and Test B, and site 4 may use Core, Test A, and Test B which means that four different data groups are required to be deployed but only 3 exist within the master data.

To maximize deployment performance, if two sites use the same combination of configurations, they will both refer to the same data group so that the combination only has to be deployed once. The data group ID for the data from the combined configurations will be generated during deployment and whenever a site is switched from one data group to another, if the original data group is no longer used by any site then it will be deleted.

When a new combination of configurations that has never been deployed before is deployed, a new deployment group ID will be generated and saved in the deployment group table with the combination of configurations that were deployed.

Deployment Group Table

GroupID	ConfigType	Config	VersionID
1	1	1	5
1	1	2	5
2	1	1	5

2	1	3	5
3	1	1	5
3	1	2	5
3	1	3	5

MDM Data API

Master Data Management can operate either by pushing data to 3rd party applications or by providing an API that consumers can use to request data and/or changes to data. This document describes how EDM's Master Data Management component publishes data and changes that consumers can request via a RESTful web service API.

Requirements

The requirements for Master Data Management API are:

1. Enable users to add additional tables to the EDM central database to hold data required by other applications. For example, to support a mobile application that needs a "webImage" for each item, a new MobileItem table could be created with columns for the item number and the web image. Existing EDM central database tables being managed for remote sites may not be modified.
2. Publish current data values so consumers can access them via a RESTful web service call. For example, when the mobile application detects a new restaurant ID that it has not seen before, the mobile app can load the entire price list, tax list, etc. for the site.
3. Publish changes to data values so consumers can access them via a RESTful web service call. For example, once the data has been loaded for a site, the mobile app can simply check for price changes periodically instead of completely reloading the price data.
4. Provide option for larger customers to minimize the impact on the performance of the production EDM central database when publishing data and changes by having the web service API use a database other than the production EDM central database.

Supporting Additional Tables

Users can create additional tables in the EDM central database, using SQL Server Enterprise Manager or other tools, to support the data elements needed by other applications. Additional tables must meet the following requirements:

1. Tables are created in the central EDM database with "MDM_dbo_" as the prefix to all table names.
2. Tables are added to the schema file with the remote data source set to "MDM" and the version set to "1".
3. The convert file must include a conversion of the MDM data source from version 0 to version 1 that adds all the tables.
4. Records are added to the remote sources table for each site with the data source name "MDM" and the version "0" to prevent transactions for the additional tables from being sent to the remote sites.

Requesting Changes

So that MDM data consumers do not have to completely reload the MDM data periodically, consumers can call a RESTful web service to obtain a list of the changes that have made to the MDM database since the last time the changes were requested.

To request changes, consumers pass in the latest effective date/time of the transactions that have received in the past. Users request changes periodically and the system returns a list of changes where: (effectiveDate is null and tranDate >= lastEffectiveDate) or (effectiveDate >= lastEffectiveDate and effectiveDate <= now). Since there is a very small chance that multiple transactions have the same effective date/time down to the millisecond and that not all the changes with the same timestamp may have been received last time changes were requested, consumers should ignore cases where a row is being inserted that already exists or updated to values it already has or deleted when it does not exist. Since the changes will always be applied in order the final result will be correct even if the changes are applied more than once.

Changes are queried via RESTful web service call to: /edm/rest/audit/changes

See [RESTful Web Service API](#) for details.

Requesting Data

Consumers can request all the data for specific tables and sites by calling a RESTful web service.

Current data values will also be published so that consumers can retrieve entire data sets for one or more sites. Data will be published to a database on a separate server from the EDM/MDM database so that consumers querying the database will not negatively impact system performance for users of the EDM/MDM system. Web services will be available to read data from the published database using simple row filtering.

For example, to query the prices for site 17, make a RESTful web service call to: /edm/rest/currentRowValues

See [RESTful Web Service API](#) for details.

Maximizing Performance

For users with large number of remote locations that need to minimize the performance impact on the EDM central database of publishing data and processing calls to the API, SQL Server replication can be used to replicate the central EDM database to one or more slave servers. The web application that processes incoming calls to the web service API will read data and/or changes from the slave database so that the queries do not impact the production EDM central database.

Regression Tests

In order to validate that the MDM publishing functionality is working properly, the following tests are to be run as part of the automated regression test suite.

Supporting Additional Tables

- Add a new "MobileItem" table with CDMLOCID, ItemNum, and WebName columns.
- Add table to schema as MDM_dbo_MobileItem (attachCentralTable function)
- Add remote table records for the MobileItem table with version 1 for sites 1, 2, and 3.
- Add remote source records for the MDM data source with version 0 for sites 1, 2, and 3.
- Open the MobileItem form, select sites 1, 2, and 3 and insert a record, update a record, and delete a record and close the form and commit all transactions.
- Validate that the insert, update, and delete transactions for the MobileItem table exist in the audit table for sites 1, 2, and 3.
- Validate that sites 1, 2, and 3 do NOT have outgoing transactions for the MobileItem table.

Requesting Changes

This test validates that only changes for which the effective date has arrived since the last requested effective date are returned.

- Create a set of price change transactions as follows (columns are Transaction ID, Transaction Date, Old Price, New Price, and Effective Date):
 - 100, 1/11/2010, 1.99, 2.99, none
 - 101, 1/13/2010, 2.99, 3.99, 1/23/2010
 - 102, 1/15/2010, 3.99, 4.99, 1/23/2010
 - 103, 1/16/2010, 4.99, 5.99, 1/17/2010
 - 104, 1/19/2010, 5.99, 6.99, none
 - 105, 1/19/2010, 6.99, 7.99, 1/1/2099
- Request changes with last effective date 1/2/2010 and validate that response includes transaction ID's: 100,101,102,103,104
- Request changes with last effective date 1/12/2010 and validate that response includes transaction ID's: 101,102,103,104
- Request changes with last effective date 1/15/2010 and validate that response includes transaction ID's: 101,102,103,104
- Request changes with last effective date 1/18/2010 and validate that response includes transaction ID's: 101,102,104
- Request changes with last effective date 1/20/2010 and validate that response includes transaction ID's: 101,102
- Request changes with last effective date 1/30/2010 and validate that response includes NO tran ID's

Request site data for a table

This test validates that data can be retrieved for all the rows in a table for a single site and that if there are open or future transactions on some of the rows, they are rolled back.

- Create MobileItem table (see steps in Support Additional Tables)
- Add rows to MobileItem table for sites 1 and 2 for items 1 and 2 with WebNames set to Site1Name1, Site1Name2, Site2Name1, and Site2Name2 respectively.
- Request all items for site 2 (URL: rest/data/MDM_dbo_MobileItem?CDMLOCID=2) and validate that items 1 and 2 are returned for site 2 only.
- Add a transaction to change the WebName of item 1 at site 2 to Site2Name1Update1 but do not commit the transaction.
- Request all items for site 2 and validate that items 1 and 2 are returned for site 2 only and the web names is Site2Name1, Site2Name2 respectively.
- Commit the transaction.
- Request all items for site 2 and validate that items 1 and 2 are returned for site 2 only and the web names is Site2Name1Update1, Site2Name2 respectively.
- Change the transaction's effective date to tomorrow.
- Request all items for site 2 and validate that items 1 and 2 are returned for site 2 only and the web names is Site2Name1, Site2Name2 respectively.
- Change the transaction's effective date to yesterday.
- Request all items for site 2 and validate that items 1 and 2 are returned for site 2 only and the web names is Site2Name1Update1, Site2Name2 respectively.

Request data performance

This test validates acceptable performance for requesting 1000 rows from a table for 1 site to ensure that it completes in 2 seconds with no

transactions to roll back and 4 seconds with 100 transactions to roll back.

- Create MobileItem table (see steps in Support Additional Tables)
- Add rows to MobileItem table for sites 1 through 100 for items 1 through 1000 with WebNames set to the item number appended to the site number..
- Request all items for site 2 and validate that all 1000 items are returned for site 2 only and the data is returned in less than 2 seconds.
- Add 100 transaction to change the WebName of 100 of the items at site 2 to append the word "update" to the WebName but do not commit the transaction.
- Request all items for site 2 and validate that all 1000 items are returned for site 2 only and the web names do not contain "update" and the data is returned in less than 4 seconds.

Basic User Guide

The pages in this section are intended for a non-administrative user of EDM who simply needs to understand how to access and use existing forms and reports. For more technical details relating to the configuring and setting up EDM, please click [here](#).

Topics

Editing Data Using EDM

Depending on where changes are being made, users can edit data for one or many locations. One of the features of EDM is the fact that data for multiple locations can be changed at the same time. This saves time and reduces errors. Because POS systems and base configuration options vary, the particulars of forms and their use will differ, but basic use principals remain the same.

This section provides information that will clarify the use of forms. Click [here](#) to return to the Basic User Guide Home.

Topics

[Editing Data for Multiple Stores](#)
[Versioning](#)

Editing Data for Multiple Stores

When connected to a central server, the data for multiple stores can be edited at in one operation. For example, you can change a menu item for multiple stores simultaneously.

Selecting Stores

Select the stores to change in one of these ways:

- Hold down the Ctrl key on your keyboard, and then click the stores to select
- Press the double green arrows to select all stores
- Double-click each store
- Click each store and click the single arrow

Once you have selected the desired stores, click *OK*.

Store Selection Screen Functions

There are several functions on the selection screen that allow you to edit the data. In some cases, buttons are removed if the particular function is restricted.

New	The <i>New</i> button opens a blank form where a new record can be created.
Delete	The <i>Delete</i> button removes the record from all the selected stores. This button is often not present because deleting records can be dangerous. Select a record and click <i>Delete</i> to remove the selected record from all selected stores.
Show Locations	The <i>Show Locations</i> button displays a list of locations that have the selected record. Select a record and click <i>Show Locations</i> to see the list.
Copy Record	The <i>Copy Record</i> button makes an exact duplicate of the selected record with a new ID number at all selected stores. This simplifies new record creation by allowing you to copy a similar item.

Copy to Sites

The *Copy to Sites* button copies the selected record to any selected stores that do not already have that record.

Finding a Menu Item

Note

The location where menu item records are stored varies by POS system type.

Find an item to be edited from the list in one of the following ways:

- Type any part of the name or number (any columns listed on the screen) in the search box at the top right. The list will reduce as you type characters.
- Click a column header to sort the list by this column. Click the header again to reverse the order.

Editing the Menu Item

Edit the item in one of these ways:

- Double-click the desired item
- Highlight the desired item, and then click *View/Change*

At this point, if the selected item is identical at all stores, the form containing the item information will appear.

Any changes made on the screen are saved when the *OK* button is selected.

Versioning

If the items differ between stores, a *Versioning* screen will appear. This screen lists one line for each version of the selected item. If the item is the same at multiple stores, they are listed on the same line. After selecting the desired version, the form containing the item information will appear. Any changes made on the screen are saved when the *OK* button is selected.

See [Versioning](#) for more information.

Versioning

When [Editing Data for Multiple Stores](#), if the data records between stores differ, a *Versioning* screen will appear. This screen lists one line for each version of the selected item. If the item is the same at multiple stores, they are listed on the same line.

Versioning Screen Functions

There are several functions on the *Versioning* screen that allow you to see what the differences are and which stores have what version.

Show Locations	The <i>Show Locations</i> button displays a list of stores that have the same version of the record. Choose one version and click <i>Show Locations</i> to see the list.
Show Differences	The <i>Show Differences</i> button compares two versions and shows only the values that are different. Hold down the Ctrl key on your keyboard, and click the two versions. Click <i>Show Differences</i> to see the differences.
View/Change	The <i>View/Change</i> button opens the form for the selected version. Only one version may be selected at a time.
Back	The <i>Back</i> button returns you to the prior screen where all of the items are listed.

Any changes made on the screen are saved when the *OK* button is selected. After editing a version, you will be returned to the *Versioning* screen where you may select another version to edit.

Java Message Service (JMS) Queues for Transaction Transfer

EDM supports transferring transaction files between remote and central installations via JMS queues. JMS queues provide fast, reliable transport of transaction files between central and remote sites and JMS brokers can be clustered behind a load balancer for high volume installations. Transaction files can be transferred at specific times or can be triggered by user actions or the system can be configured to automatically send transactions as soon as they are committed and to receive transactions as soon as they are received for near-real-time data transfer. JMS offers guaranteed delivery and also guarantees that the messages are delivered in the order they were sent and will not be delivered more than once.

We will document and publish the details pertaining to Java Message Service (JMS) configuration here. You can navigate to each page using the side bar to your left or the links below::

ActiveMQ Setup

Overview

Apache [ActiveMQ](#) is an open sourced implementation of JMS. EDM can be configured to use ActiveMQ's messaging capabilities. We will document how to install and configure ActiveMQ so that it works with regard to EDM messaging here. You can navigate to each section using the links below:

- [Example](#)
- [Example](#)

Install ActiveMQ on EDM Central

The current supported version of ActiveMQ is 5.8.0.

1. Click [here](#) to download the installation zip file.
2. Download the installation zip file to the server.
3. Extract the contents of the zip file to the root of the C: drive.

Configure ActiveMQ for EDM Central

To configure ActiveMQ for the EDM Central Server, perform the following steps:

1. Rename C:\ActiveMQ\conf\activemq_central.xml to C:\ActiveMQ\conf\activemq.xml
2. Run the following from an administrator command prompt to install the ActiveMQ service:

```
cd\
cd c:\activemq\bin\win32
InstallService.bat
```

Configure the EDM Central Server to support ActiveMQ

1. Open the config.xml file from C:\EDMServer\webapps\EDMWEB-INF\classes.
2. Add the *onStartup* attribute inside the *company* tag using the following as a template:

```
onStartup = "addProcessReceivedTransactionsListener() + addJMSCommitListener( '*',
'com.xpient.EDM.<CustomerID>.outgoing', 'tcp://localhost:61616', 'system', 'manager' ) + addJMSReceiveListener(
'com.xpient.EDM.<CustomerID>.incoming', 'tcp://localhost:61616', 'system', 'manager' )"
```

3. Replace *<CustomerID>* with the customer's provided Customer ID.

Example

(CustomerID: CustomerIDExample1234)

```
onStartup = "addProcessReceivedTransactionsListener() + addJMSCommitListener( '*',
'com.xpient.EDM.CustomerIDExample1234.outgoing', 'tcp://localhost:61616', 'system', 'manager' ) + addJMSReceiveListener(
'com.xpient.EDM.CustomerIDExample1234.incoming', 'tcp://localhost:61616', 'system', 'manager' )"
```

Configure EDM Remote

To configure EDM to work with the ActiveMQ instance on EDM Central, perform the following steps:

1. Open the config.xml file from C:\EDMWeb.

If you are configuring an EDMServer Remote, this file is located at C:\EDMServer\webapps\EDM\WEB-INF\classes.

2. Add the *onStartup* attribute inside the *company* tag using the following as a template:

```
onStartup = "addProcessReceivedTransactionsListener() + addJMSCommitListener( '*',  
'com.xpient.EDM.<CustomerID>.incoming', 'tcp://<Server Name>:61616', 'system', 'manager' ) + addJMSReceiveListener(  
'com.xpient.EDM.<CustomerID>.outgoing', 'tcp://<Server Name>:61616', 'system', 'manager' )"
```

3. Replace *<CustomerID>* with the customer's provided Customer ID.
4. Replace *<Server Name>* with the URL of the EDM Central server instance.

Example

(CustomerID: CustomerIDExample1234 and Server Name: mwb.xpientxpress.com)

```
onStartup = "addProcessReceivedTransactionsListener() + addJMSCommitListener( '*',  
'com.xpient.EDM.CustomerIDExample1234.incoming', 'tcp://mwb.xpientxpress.com:61616', 'system', 'manager' ) +  
addJMSReceiveListener( 'com.xpient.EDM.CustomerIDExample1234.outgoing', 'tcp://mwb.xpientxpress.com:61616', 'system',  
'manager' )"
```

Automated Setup for New Customers

Executive Summary

Customers using XPRESS POS, Wayne's Nucleus POS, and other POS systems can easily install, use, and pay for EDM remote management without any assistance from XPIENT personnel. Remote management provides great value for customers with a single site and even more value for customers with hundreds or thousands of sites. By providing the software as a service and making it easy to install at remote sites and a risk-free experience, users all over the world can enjoy the benefits with just a few clicks of the mouse.

Benefits to Customers

- Remotely manage prices, items, images, documents, and other data from any browser or mobile device connected to the internet
- Access sales and labor reports remotely
- Eliminate evening and weekend work by making changes with effective dates
- Eliminate need to remotely connect to sites
- Reduce data entry when managing multiple sites
- Improve data consistency across multiple sites

Risk-free for Customers

- Try the system for free for 30 days for up to 3 sites
- Low monthly payment with discount for paying 6 months or a year in advance
- Web site is secure with HTTPS access
- Data is secure in a private database for each customer

Key Components

The key components are:

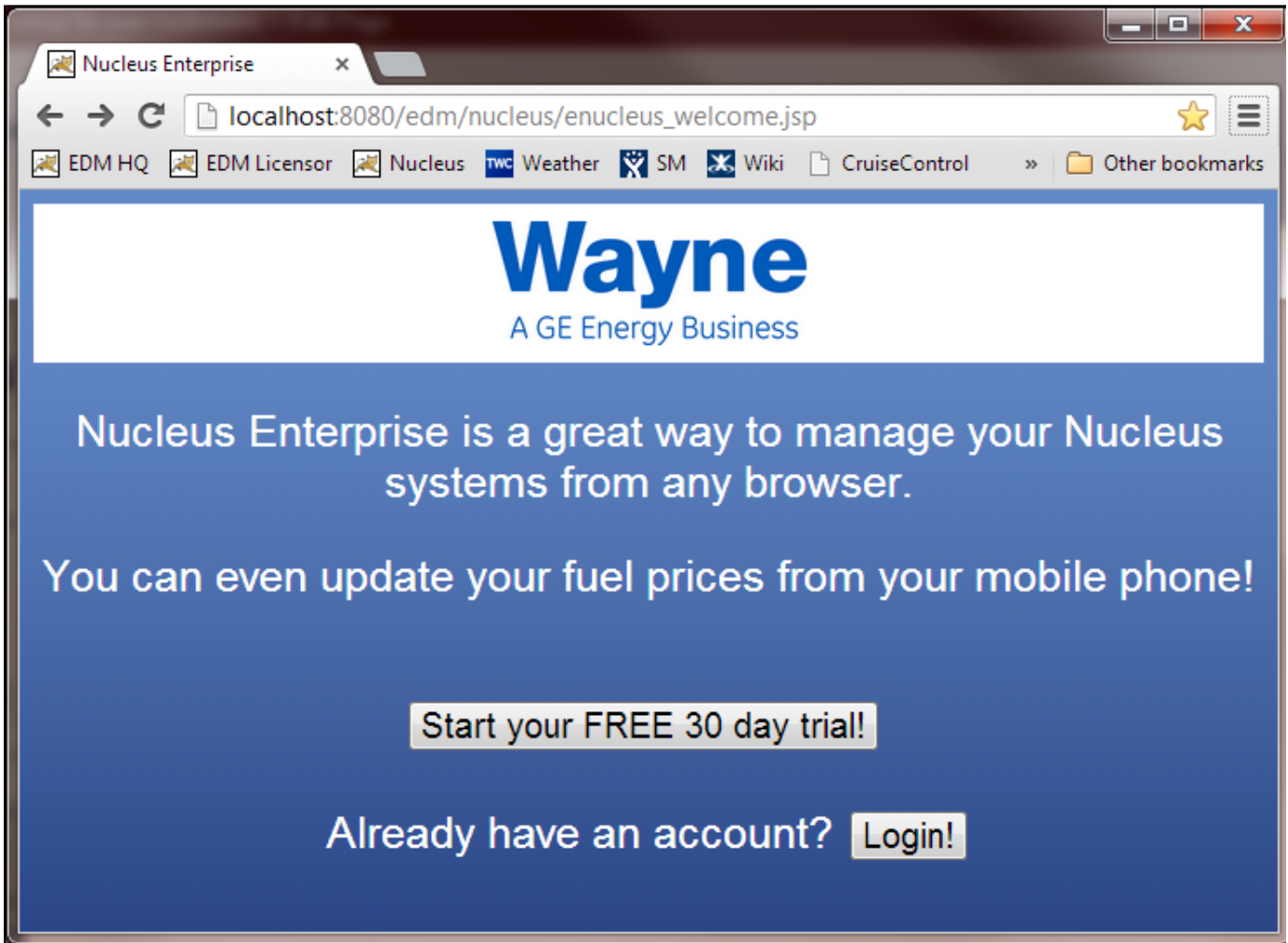
- Hosted central web and database servers for SaaS
- Self-service web page for creating new accounts
- Self-service web page for adding a new remote site
- Remote app installer program
- EDM web application to access forms and reports for managing site data
- Automatic upgrades
- Self-service payments
- Online demos, training, tutorials, documentation, and tech support

Hosted Software as a Service (SaaS)

EDM web and database servers will be hosted in XPIENT's data center so that customers don't have to concern themselves with setting up a

central server and the XPIENT Professional Services team doesn't have to navigate each customer's networking and security systems to help them get the systems up and running.

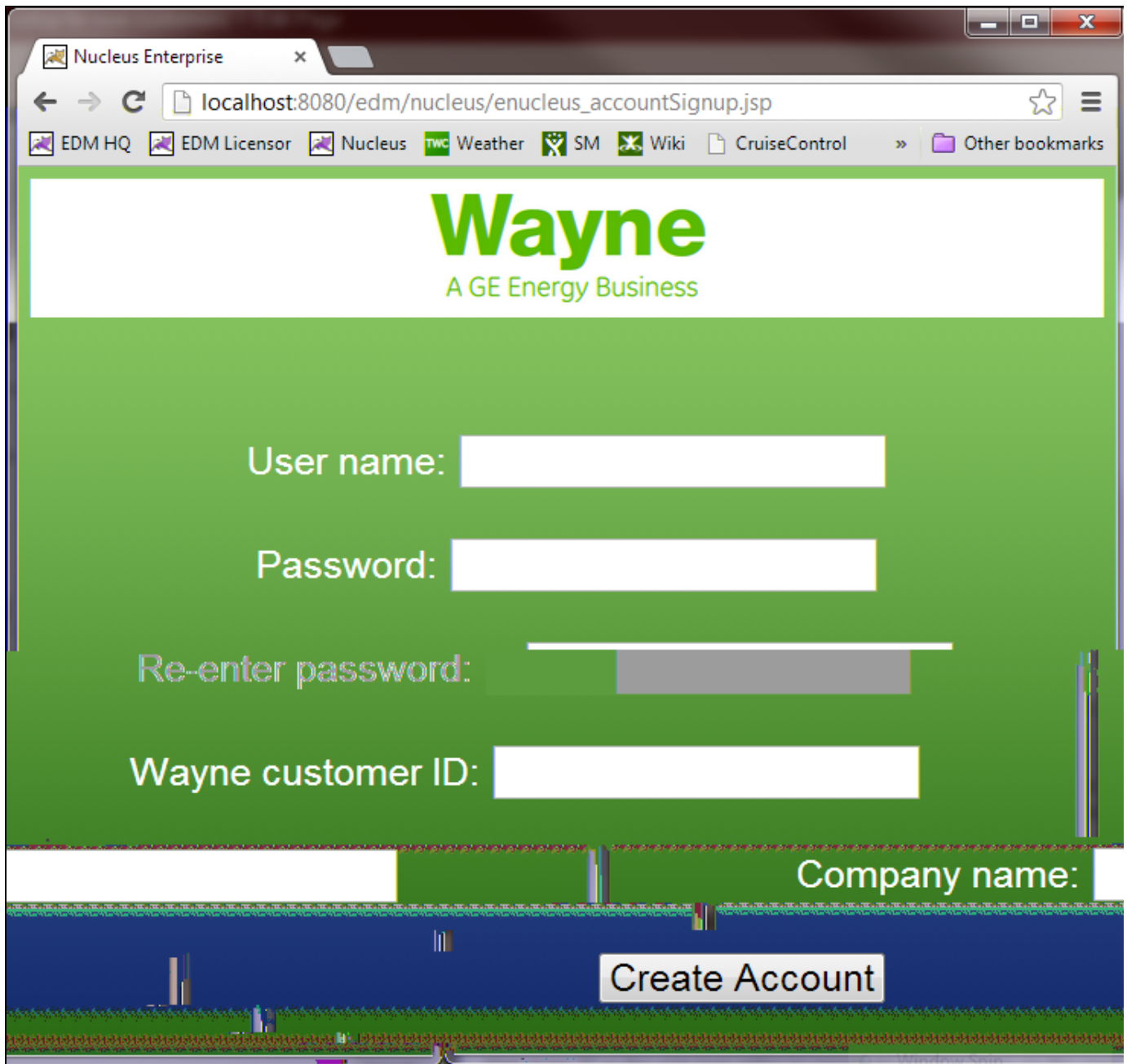
The system is multi-tenant so multiple customers can share the same web and database servers while still maintaining privacy for each customer's data.



Self-Service Web Page for Creating New Accounts

Users can create their own account in the system using a web page. The system performs the following tasks when a new account is created:

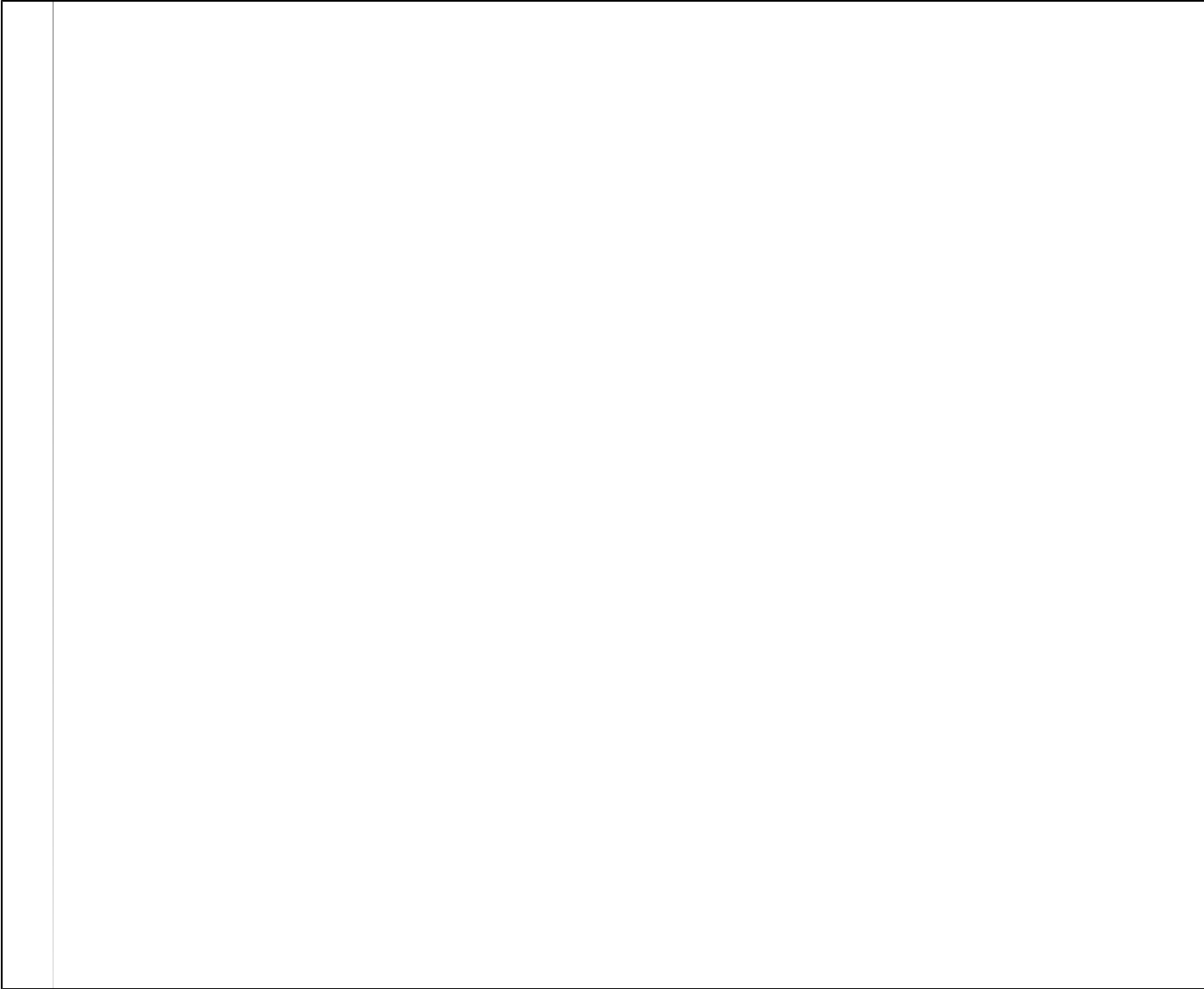
- Verify that the new user ID is unique
- Creates a new database for the customer's site data
- Creates the new user in the company database with a "Company Administrator" role
- Generates license for the company that expires when their next payment is due
- Captcha validation to prevent automated new account setup which could indicate a denial-of-service attack



Self-Service Web Page for Adding a New Remote Site

Users can start managing a site by downloading the remote app installer.

- To add a remote site, customers log in from the appropriate computer at the remote site
- Clicking the "Add a new site" link on the main menu takes them to a web page where they can download the remote app installer program to the remote PC



•
•
•

An upgrade service, installed by the installer program, checks for upgrades to the remote app periodically and installs them when they are available

Self-service Payments

Customers can either pay for the use of the system directly or can enter an ID such as the Wayne Customer ID if a reseller is paying the bill for them. Payment options can include e-checks or credit cards. No collections activities should be needed since the system will simply stop working for them if their payments are not made.

- New customers can try the system for free for 30 days
- A few days before a payment is due, an email will be sent to the customer with a link to the payment page where payment can be made via e-check or credit card.
- If payments are not received when due, the license expires and customers can make a payment to renew the license
- If an account has an expired license for a year, the account and all site data will be deleted from the system

Online Support

- Online demos help users understand the benefits of the system
- Online training and tutorials help new users learn to use the system efficiently
- Online documentation provides in-depth reference information for the system
- Web-based tech support and user forum makes it easy for users to get their questions answered quickly.

Automated Setup of New Remote Sites

When a new XPRESS or Nucleus EDM client is installed at a remote site have it automatically communicate with central to set up the site there. If the central site already exists, the system does nothing since the user is obviously setting up the site data themselves and we don't want to make assumptions about the procedure. If the central site does not exist, create the central site, if it is the first site in the company then send the schema file to create the central database, and send all configuration and transaction data to central so the user can begin managing the data immediately.

Overview

To install EDM at a new remote site as easily as possible, the user will perform the following steps:

1. Browse to the EDM company and login from the computer where the EDM remote software is to be installed.
2. Click the "Install Remote Site" menu option on the EDM menu which reminds the user that they must be using the browser on the PC where the software should be installed and prompts them to enter the new site ID and name (since the site ID and name can't be found in the Nucleus database).
3. Enter the new site number and name and the system will verify that the ID is not already in use and click "Goto Download Page".
 - a. EDM verifies that the site number is not in use
 - b. Create a site record
 - c. Create a installation executable that
 - i. Prompts the user to select an install directory
 - ii. Reads the site number, name, zip file URL, central URL, central user ID, and central password from the CompileInfo.txt file.
 - iii. Downloads the zip file to the installation directory and unzips it
 - iv. Installs EDM as a service and starts the service.
 - d. Display the download page
4. User clicks the "Download Remote App" button on the web page to download the installation file.
5. Run the downloaded installation file.

After performing the steps above, the system will automatically send all the site's data to the central database and the user can begin managing the data immediately.

Configuring the System

In order to simplify the setup of new remote sites, take the following steps to configure the system.

1. Install a lab system in the same way that the remote sites would be set up including setting the specific URL of the central web server, the company name and company ID, etc.
2. Zip the remote installation from the lab system and place the .zip file in the webapps/EDM/company/[companyId]/RemoteInstaller directory. The companyId is the company ID from config.xml that was calculated when the company was set up.
3. Copy the directory used to create pre-configured remote installation exe's from the AccuRev workspace in the

ExJet/EDM-Auto-Remote-Installer directory to the webapps/EDM/WEB-INF/EDMInstaller directory.

4. Edit the download.html file and the CompileInfo.txt file to have the correct settings for the company and copy them to the webapps/EDM/company/[companyId]/RemoteInstaller directory.
5. Add a menu item to the company's menu with the name = "Create Remote Site Installer" and the action = "createRemoteInstaller()" and click the menu option to create the remote installer.
6. Copy the URL of the displayed browser window to email to onsite managers or installers at the remote sites and have them browse to the URL from the PC that can connect to the database to be managed and download and run the installer.

Installing a New Version

When a new version of the software is released, repeat the steps above to update the remote installation creator.

Tests

Test 1: Install remote site when no sites exist in the central company

1. Install EDM on an XPRESS or Nucleus remote test site with an unused store ID
2. Log in to EDM at the remote test site
3. Select Connect to Central on the menu
4. Enter the URL of the central site (such as <https://XPRESS.XPIENT.com/EDM>) and the userId and password given to you by the company administrator for connecting to the central system.
5. Click OK to connect the site to the central system.
6. Verify that the site was created at central
7. Verify that the central database has all the configuration and transaction tables created
8. Verify that central has the correct XPRESS or Nucleus data source version
9. Verify that the remote site configuration and transaction data was loaded into the central tables

Test 2: Install another remote site with a new ID and the same version of the database structure

1. Install EDM on an XPRESS or Nucleus remote test site with an unused store ID
2. Start EDM on the remote test site
3. Verify that the site was created at central
4. Verify that central has the correct XPRESS or Nucleus data source version
5. Verify that the remote site data was loaded at central

Test 3: Install another remote site with a new ID and a different, newer version of the database structure

1. Install EDM on an XPRESS or Nucleus remote test site with an unused store ID
2. Start EDM on the remote test site
3. Verify that the site was created at central
4. Verify that central has the correct XPRESS or Nucleus data source version
5. Verify that central has been converted to the newer version of the database structure and that the table conversions are correct
6. Verify that the remote site data was loaded at central

Test 4: Install with site ID that already exists at central

1. Install EDM on an XPRESS or Nucleus remote test site with ID that already exists at central.
2. Start EDM on the remote test site
3. Verify that no remote data was sent to central

Design

1. Add a new transaction type INIT_SITE that is sent to central any time the system is started up and has to attach a remote data source. This happens on initial installation, on site reset, when a data source version is modified in config.xml and when a new data source is added to config.xml. The transaction data includes the site ID, name, custom ID, transaction version, EDM version number, and the version number of each of the data sources in config.xml (this comes from the DBVERSION table in XPRESS). When central receives the transaction it performs the following operations:
 - a. Accept the transaction even though the site may not exist yet (how will it know to receive the file or JMS message? with web transfer we could do it or with a web service call, with JMS we would have to scan for new site ID's in the incoming queue or a different message type or other message property selector)

- b. Create the site if it does not exist
 - c. Request a table list if this is the first site in the company or the sites data source versions are not registered in the remote sources table ???
 - d. Convert the central database if the site has a newer version of any data sources than any existing sites (each data source version and its conversion must be known to central before accepting the site's table list)
 - e. Request a refresh of configuration and transaction tables if the site data has not been loaded
 - f. Save the site's EDM version number in a site property called EDM_VERSION_NUMBER
 - g. Save the sites data source versions in the remote sources table (in XPRESS they come from the active row in the dbversion table, find out how to identify them for Wayne)
2. If central site must exist before it can receive the INIT_SITE transaction then, add a new config property called initSiteWebService that contains the URL of the central web service WSDL and the user id and encrypted password (or OAuth token) to used to setup the central site. The user ID and password should be for the company administrator (secure?).
 3. EDM web services use Amazon-style security where each request is signed using the secret key of the user. The user installing the software at the remote site must have a credentials file (like awscredentials) which contains the encrypted access key and secret key used to sign requests sent to the central web service. The access key is the user ID of the central user whose permissions will be used for processing the call. EDM provides a function to create the awscredentials file with a user-entered access key and secret key. EDM clients will use the QueryStringSigner class in AWS library to sign requests. Third parties clients can also use the AWS libraries or implement their own signer based on the available AWS source code.

RESTful Web Service API

The system automated regression test tests the RESTful web services of the system after each build. This page shows the URL's and parameters of each of the requests along with sample response data from the test database.

For information about creating a signature, see [Creating a Signature for the RESTful API](#).

Common Request Headers

All REST requests must have a set of standard request headers for the server to authenticate and authorize the request.

- x-EDM-company: The name of the company to which the request is being directed (a single server can support multiple companies)
- x-EDM-timestamp: The date and time of the request.
- x-EDM-userid: ID of the user making the request
- x-EDM-signature-method: The algorithm used to calculate the request signature (currently, only HmacSHA256 is supported)
- x-EDM-signature: The request signature which the server will validate to ensure that the request is valid.

The following list shows examples of the common request headers with values assigned:

- x-EDM-company=HQ
- x-EDM-timestamp=2013-09-13T15:29:45.920Z
- x-EDM-userid=admin
- x-EDM-signature-method=HmacSHA256
- x-EDM-signature=sd208sjdf20hvfsh02glsryt-29hvwv028n0ghs0934=

Request Parameters

Some requests require parameters to be included in the body of the request. The parameters must be formatted as URL-encoded form parameters.

Service: GET eval

Evaluate the given expression and return the results.

Parameters

expr - Expression to evaluate.

Returns

Result of evaluating expression.

Example:

GET /EDM/rest/eval?expr=3+2

Response:

```
{  
  "result" : "OK",  
  "response" : "5"  
}
```

Service: GET config

Get the value of the config property value with the given name.

Parameters

propertyName - Name of the property.

Returns

Value of the property or null if not found.

Example

GET /EDM/rest/config/autoCommit

Response

```
{  
  "result" : "OK",  
  "response" : "Y"  
}
```

Service POST remoteDataSource

Add/update a remote data source.

Parameters

siteId - ID of the remote site with the data source.

dataSourceName - Name of data source.

version - Version of the data source at the remote site.

Returns

Remote data source that was created or null if it already exists.

Example

POST /EDM/rest/remoteDataSource?dataSourceName=IRIS&siteId=0&version=Ver%207.7

Response

```
{
  "result" : "OK",
  "response" : { "siteId" : 0, "dataSourceName" : "IRIS", "version" : "Ver
7.7" }
}
```

Service: GET remoteDataSource

Get a remote data source.

Parameters

siteId - ID of the remote site with the data source.
dataSourceName - Name of data source.

Returns

Remote data source that was created or null if it already exists.

Example

GET /EDM/rest/remoteDataSource/{siteId}/{dataSourceName}

Response

```
{
  "result" : "OK",
  "response" : {
    "siteId" : 0,
    "dataSourceName" : "IRIS",
    "version" : "Ver 7.7"
  }
}
```

Service: DELETE remoteDataSource

Parameters

siteId - ID of the remote site with the data source.
dataSourceName - Name of data source.

Returns

Null

Example

DELETE /EDM/rest/remoteDataSource/{siteId}/{dataSourceName}

Response

```
{
  "result" : "OK",
  "response" : null
}
```

Service: GET /EDM/rest/remoteDataSources/count

Get the number of rows in the CDMRemoteDataSources table.

Parameters

None

Returns

Number of rows in the CDMRemoteDataSources table.

Example

GET /EDM/rest/remoteDataSources/count

Response

```
{
  "result" : "OK",
  "response" : 3
}
```

Service: GET /EDM/rest/remoteDataSources/known

Get a list of the known, unique data source versions.

Parameters: None

Returns

List of known data source versions as string of the form "dataSourceName,version".

Example

/EDM/rest/remotedatasources/known

Response

```
{
  "result" : "OK",
  "response" : [ "IRIS,3.7.4 PR10", "LABORPRO,2.6.2", "LABORPRO,2.6.2(2)" ]
}
```

Service: POST /EDM/rest/siteProperty

Add/update a site property.

Parameters

name - Property name.

description - Property description.

sortType - 0 = sort alphabetically, 1 = sort numerically.

Returns

Site property that was saved.

Example

POST /EDM/rest/siteProperty?description=a%20description&name=Site%20Property%20Name&sortType=0

Response

```
{
  "result" : "OK",
  "response" : {
    "id" : 4,
    "name" : "Site Property Name",
    "description" : "a description",
    "type" : 0,
    "source" : "TABLE"
  }
}
```

Service GET /EDM/rest/siteProperty/{propertyName}

Parameters

propertyName - Name of the site property.

Returns

Value of the property or null if not found.

Example

GET /EDM/rest/siteProperty/Site%20Property%20Name

Response

```
{
  "result" : "OK",
  "response" : {
    "id" : 4,
    "name" : "Site Property Name",
    "description" : "a description",
    "type" : 0,
    "source" : "TABLE"
  }
}
```

Service: DELETE /EDM/rest/siteProperty/{propertyName}

Delete the site property with the given name.

Parameters

propertyName - Name of the site property.

Returns

Value of the deleted property or null if not found.

Example

DELETE /EDM/rest/siteProperty/Site%20Property%20Name

Response

```
{
  "result" : "OK",
  "response" : {
    "id" : 4,
    "name" : "Site Property Name",
    "description" : "an updated description",
    "type" : 0,
    "source" : "TABLE"
  }
}
```

Service: GET /EDM/rest/sitePropertyValue/{siteId}/{propertyName}

Get the value of the site property with the given site id and name.

Parameters

siteId - ID of the site whose property is to be returned.

name - Name of the property.

Returns

Value of the property or null if not found.

Example

/EDM/rest/sitepropertyvalue/123/EDM_version

Response

```
{
  "result" : "OK",
  "response" : {
    "SITEID" : 0,
    "PROPID" : 3,
    "PROPVAL" : "Version 7.1.9.6"
  }
}
```

Service: GET /EDM/rest/site/{siteId}

Get a site by its ID.

Parameters

id - ID of the site.

Returns

Site with the given ID or null if not found.

Example

/EDM/rest/site/123

Response

```
{
  "result" : "OK",
  "response" : {
    "id" : 123,
    "name" : "Main Street Store",
    "customId" : "",
    "hostName" : "",
    "userId" : "",
    "password" : "",
    "company" : "",
    "transactionVersion" : 9
  }
}
```

Service: POST /EDM/rest/site

Add/update a site.

Parameters

id - Unique site id.
name - site name.
customId - Customer assigned site id.
hostName - Host name for transaction file transfers.
userId - User id for transaction file transfers.
password - Password for transaction file transfers.
company - Company to log into for transaction file transfers.
transactionVersion - Transaction version at this site.
reportTransactionErrors - True if transactions errors are reported for this site.

Returns

Site that was created or null if it already exists.

Example

POST
/EDM/rest/site?company=&customId=&hostName=&id=-123&name=test&password=&reportTransactionErrors=1&transactionVersion=9&userId=

Response

```
{
  "result" : "OK",
  "response" : {
    "id" : 123,
    "name" : "test",
    "customId" : "",
    "hostName" : "",
    "userId" : "",
    "password" : "",
    "company" : "",
    "transactionVersion" : 9
  }
}
```

Service: DELETE /EDM/rest/site/{siteId}

Delete the site with the given ID.

Parameters

id - ID of the site.

Returns

Delete site or null if not found.

Example

/EDM/rest/site/123

Response

```
{
  "result" : "OK",
  "response" : {
    "id" : 123,
    "name" : "test",
    "customId" : "",
    "hostName" : "",
    "userId" : "",
    "password" : "",
    "company" : "",
    "transactionVersion" : 9
  }
}
```

UNIT TEST: testRemoteTablesGet

REST request: GET /EDM/rest/remoteTables/maintained/{siteId}

Parameters: None

REST response:

```
{
  "result" : "OK",
  "response" : [{
    "tableName" : "IRIS_dbo_tbl_ItemMaster",
    "tableVersion" : "3.7.4 PR10",
    "remoteSite" : 0,
    "dataSite" : 0,
    "override" : "centralOnly",
    "ignoreInsert" : false,
    "ignoreUpdate" : false,
    "ignoreDelete" : false,
    "shareGroup" : null
  }, {
    "tableName" : "IRIS_dbo_tbl_ItemPricing",
    "tableVersion" : "3.7.4 PR10",
    "remoteSite" : 0,
    "dataSite" : 0,
    "override" : "central",
    "ignoreInsert" : false,
    "ignoreUpdate" : false,
    "ignoreDelete" : false,
    "shareGroup" : null
  }, {
    "tableName" : "IRIS_dbo_tbl_ItemPricing_Child",
    "tableVersion" : "3.7.4 PR10",
    "remoteSite" : 0,
    "dataSite" : 0,
    "override" : "central",
    "ignoreInsert" : false,
    "ignoreUpdate" : false,
    "ignoreDelete" : false,
    "shareGroup" : null
  }, {
    "tableName" : "LABORPRO_dbo_tblLsPositioningGuide",
    "tableVersion" : "2.6.2",
    "remoteSite" : 0,
    "dataSite" : 0,
    "override" : "central",
    "ignoreInsert" : false,
    "ignoreUpdate" : false,
    "ignoreDelete" : false,
    "shareGroup" : null
  }, {
    "tableName" : "LABORPRO_dbo_tblLsTimeInterval",
    "tableVersion" : "2.6.2",
    "remoteSite" : 0,
```



```
"dataSite" : 0,  
"override" : "central",  
"ignoreInsert" : false,  
"ignoreUpdate" : false,  
"ignoreDelete" : false,  
"shareGroup" : null
```

```
} ]  
}
```

UNIT TEST: testRemoteTableGet

REST request: GET /EDM/rest/remoteTable/{tableName}/{siteId}

Parameters: None

REST response:

```
{  
  "result" : "OK",  
  "response" : {  
    "tableName" : "IRIS_dbo_tbl_ItemMaster",  
    "tableVersion" : "3.7.4 PR10",  
    "remoteSite" : 0,  
    "dataSite" : 0,  
    "override" : "centralOnly",  
    "ignoreInsert" : false,  
    "ignoreUpdate" : false,  
    "ignoreDelete" : false,  
    "shareGroup" : null  
  }  
}
```

UNIT TEST: testGetSiteList

REST request: GET /EDM/rest/sites

Parameters: None

REST response:

```

{
  "result" : "OK",
  "response" : [{
    "id" : 0,
    "name" : "Burger King HQ",
    "customId" : "",
    "hostName" : "",
    "userId" : "",
    "password" : "",
    "company" : "",
    "transactionVersion" : 9
  }, {
    "id" : 1,
    "name" : "Main Street",
    "customId" : "",
    "hostName" : "",
    "userId" : "",
    "password" : "",
    "company" : "",
    "transactionVersion" : 9
  }, {
    "id" : 2,
    "name" : "Lake Shore",
    "customId" : "",
    "hostName" : "",
    "userId" : "",
    "password" : "",
    "company" : "",
    "transactionVersion" : 8
  }, {
    "id" : 3,
    "name" : "East Side",
    "customId" : "",
    "hostName" : "",
    "userId" : "",
    "password" : "",
    "company" : "",
    "transactionVersion" : 7
  } ]
}

```

UNIT TEST: testGetTableList

REST request: GET /EDM/rest/tableList/{tableListName}

Parameters: None

REST response:

```
{
  "result" : "OK",
  "response" : {
    "name" : "Menu Editor",
    "tableNames" : [
      "IRIS_dbo_tbl_MenuBtns",
      "IRIS_dbo_tbl_MenuCmds",
      "IRIS_dbo_tbl_MenuMaster",
      "IRIS_dbo_tbl_MenuMacros",
      "IRIS_dbo_tbl_MenuMacroDetails",
      "IRIS_dbo_tbl_MenuBtnTypes",
      "IRIS_dbo_tbl_MenuObj",
      "IRIS_dbo_tblMenuResolutionCurrent"
    ]
  }
}
```

UNIT TEST: testError

REST request: GET /EDM/rest/error

Parameters: None

REST response:

```
{
  "result": "ERROR",
  "response": "EDMcommon.error.NotImplementedError: Method not implemented (Yet)."
```

UNIT TEST: testGettingChangesAndCurrentRowValues

REST request: GET /EDM/rest/audit/changes

Parameters:
since=2010-01-02
site=1
table=IRIS_dbo_tbl_UpsizeType

REST response:

```
{
  "result" : "OK",
  "response" : [{
    "type" : "updateRow",
    "version" : 9,
    "FromLoc" : 0,
```

```

    "ToLoc" : 1,
    "Id" : 7085,
    "PkgId" : 0, "EffDate" : "2010-01-11 00:00:00.000",
    "TranDate" : "2010-01-11 00:00:00.000",
    "StatusId" : "Committed",
    "StatusDate" : "2010-01-11 00:00:00.000", "EffDate" : "2010-01-11 00:00:00.000",
    "TableNames" : "IRIS dbo.tbl_UpsizeType",
    "Version" : "3.7.4 PR10",
    "Override" : "None",
    "User" : "admin",
    "Force" : false,
    "Conflict" : "",
    "Undo" : "",
    "Fields" : ["CDMLOCID", "UpsizeType", "-1492", "-1492"],
    [ "Description", "testA", "testB" ] ]
  }, {
    "type" : "updateRow",
    "version" : 9,
    "FromLoc" : 0,
    "ToLoc" : 1,
    "Id" : 7088,
    "PkgId" : 0,

```

"EffDate" : "20

```
"Undo" : "",  
"Fields" : [ [ "CDMLOCID", "1", "1" ], [ "UpsizeType", "-1492", "-1492"  
], [ "Description", "testC", "testD" ] ]
```

```
} ]  
}
```

REST request: GET /EDM/rest/currentRowValues

Parameters:

asOf=2010-01-09

site=1

table=IRIS_dbo_tbl_UpsizeType

REST response:

```
{  
  "result" : "OK",  
  "response" : [  
    [ "CDMLOCID", "UpsizeType", "Description" ]  
  , [ 1, 14, "test" ]  
  , [ 1, 13, "Only" ]  
  , [ 1, 12, "Add" ]  
  , [ 1, 11, "Heavy" ]  
  , [ 1, 10, "No" ]  
  , [ 1, 9, "Lite" ]  
  , [ 1, 8, "Small" ]  
  , [ 1, 7, "Senior" ]  
  , [ 1, 6, "Big Kids" ]  
  , [ 1, 5, "Sandwich Only" ]  
  , [ 1, 4, "King" ]  
  , [ 1, 3, "Large" ]  
  , [ 1, 2, "Medium" ]  
  , [ 1, 1, "atest1-a" ]  
  , [ 1, \-1492, "test" ]  
  ]  
}
```

Service: GET audit/getChangedTables

Get map of lists of site IDs that have had changes since the effective date/time keyed by table name.

Parameters:

sites - Comma-separated list of site IDs to include in list of changed tables ("*" means all sites)

table - Table name for to check for changes (null or blank means all tables).

since - Earliest effective date of transactions to be include.

Returns:

Map of lists of site IDs that have had changes since the effective date/time keyed by table name.

Example:

GET /EDM/rest/audit/changedTables?sites=12&table=IRIS_dbo_tbl_ItemMaster&since=2013-08-15

Response

```
{
  "result" : "OK",
  "response" : {
    "IRIS_dbo_tbl_ItemMaster" : [ 1 ],
    "IRIS_dbo_tbl_ItemPricing" : [ 1 ]
  }
}
```

Creating a Signature for the RESTful API

The legacy EDM RESTful Web Service API requires the passing of a x-EDM-signature header parameter. While this signature is based on AWS authentication, it uses a custom implementation unique to EDM which does not require standard AWS parameters. Additional details of the AWS Authentication methodology may be found in Amazon's AWS documentation, but please note their documentation details a standard AWS implementation.

This page describes the necessary steps to create a signature for the legacy API. Click [here](#) to go to the RESTful Web Service API home page.

Calculating the String to Sign

Include all request and query parameters in the provided list of parameters. The following table describes the request header parameters.

Parameter	Set the parameter value to
x-edm-userid=	User ID for the request
x-edm-timestamp=	Timestamp for the request
x-edm-signature-method=	HmacSHA256
x-edm-company=	Company against which the request is run

Request Example

In the following example, the request is to run the "eval" service with the parameter "expr" set to "3 + 2" with sample headers.

```
x-edm-userid=admin
x-edm-timestamp=2015-10-23T19:59:26.329+0000
x-edm-signature-method=HmacSHA256
x-edm-company=IRIS
with:
URI http://localhost:8080/edm/rest/eval?expr=3+%2B2
```

Creating the URI-Encoded String

Append a Canonicalized URI to the request parameters. To form the canonicalized query string:

1. Sort all the query string parameters.
2. URI-encode both the key and value and then join them in the desired sequence, separating the key value pairs with the ampersand symbol (&).

This is computed as an RFC 2104-compliant HMAC signature.

URI-Encoded String Example

In the following example, the parameters are ordered alphabetically and URI-encoded.

```
GET\n/edm/rest/eval\nexpr=3%20%2B%202&x-edm-company=IRIS&x-edm-signature-m  
ethod=HmacSHA256&x-edm-timestamp=2015-10-23T19%3A59%3A26.329%2B0000&x-edm-  
userid=admin
```

Creating the Encoded Signature

The URI-encoded string is encrypted using HmacSHA256 and returned as a Base64-encoded string; the result is the signature.

```
hz1TIbxChRkuN2Ywm5HSYfHxfb0AYkD/gIM3Zbqq/8Q=
```

This entire process produces the following request headers:

```
{Accept=[application/json], x-edm-userid=[admin],  
x-edm-timestamp=[2015-10-23T19:59:26.329+0000],  
x-edm-signature-method=[HmacSHA256], x-edm-company=[IRIS],  
x-edm-signature=[hz1TIbxChRkuN2Ywm5HSYfHxfb0AYkD/gIM3Zbqq/8Q=]}
```

RESTful Web Service for Authentication

This page shows the URLs and parameters of each of the authentication based API requests, along with sample response data from the test database. All requests are POSTs.

- EDM RESTful Web Service Calls and Responses
- Calling the Web Services
 - Service Call Response Objects
 - Request Header
 - Request Parameters
 - Status Codes
 - Permissions
- Authentication Service
 - Method login
 - Method loginToCompany
 - Method logout
 - Method setCompany
 - Method getCompanyInfo
 - Method getCompaniesConnectionStrings
 - Method isValidToken
- User Administration
 - Method getRoles
 - Method getACL
 - Method getRoles

EDM RESTful Web Service Calls and Responses

The EDM web services API may be accessed by appending `/api/service/method` to the URL of the desired EDM host. Logging into a default local instance would be performed at <http://localhost:8080/edm/api/auth/login>.

EDM expects a form POST - `Content-Type application/x-www-form-urlencoded` - and provides `Content-Type`

[application/json](#). The methods of the currently available services, *Authentication Service* and *User Administration*, are documented in subsequent sections below.

Calling the Web Services

To call the web services, post the parameters defined in the service documentation to the appropriate URL. Each session with EDM will be initialized with a call to `login`, passing `userId` and `password` parameters and retrieving a list of companies the user is able to administer and a token to track the user's visit.

Once the user has selected a company, the `setCompany` method will be called passing back the token as the "`token`" header parameter and the selected company name as a `companyName` form parameter. The application will then be able to use the token to call all available services.

The token will expire if the application does not access a web service for a predetermined period of time, specific to each company. The expiration time of the token will be reset with each call. The application can reset the expiration time by calling `getCompanyInfo` or `isValidToken`. If an attempt is made to call any service with an outdated token, EDM will provide a response of status 1 (`expired token`).

Service Call Response Objects

Each service call will return a response object consisting of:

- Numeric status code ("`status`")
- Optional error message ("`message`")
- Generic data Object ("`data`")

The data object will vary based upon the contents – some data will be simple lists while others require more complex structures - but the API will present each type of data as simply as is practical. EDM errors will be captured and a failure response should be returned to the application with the appropriate code and the thrown error (or more likely its message) as the message.

Request Header

After the initial connection, all REST requests must have the token request headers for the server to authenticate and authorize the request. To accommodate sessionless environments, the only state maintained will be via these tokens.

- `token`: The token for the User/Company association of the current user and visit (a single EDM server can support multiple simultaneous visits and tokens for a user).

An examples of the token request headers with value assigned:

- `token=228a51d5-97e8-419b-976b-3897525aaa61`

Request Parameters

Some requests require parameters to be included in the body of the request. The parameters must be formatted as URL-encoded form parameters.

Status Codes

Code	What it indicates
0	Success
1	Expired token
2	Invalid token
3	Access denied
4	Invalid credentials - used for login, not user management
5	Invalid userId - used for user management, not login
6	Invalid company
7	Unexpected error – this will be returned with the Error message if an Error is thrown that is not directly relatable to more explicit statuses

Permissions

Bitmask	What it indicates
0	NONE
1	VIEW
2	ADD
4	EDIT
8	DELETE
16	EXECUTE

NONE permissions will take precedence when multiple roles' permissions clash.

Authentication Service

Method login

Log user into the system. The 2 login methods are the sole API calls to not include `token` in the header.

Parameters

`userId` - username of the user initiating the "session"

`password` - The user's password.

Returns

Token and list of users' companies, if the users' credentials are valid, in a custom object with Token value and List of Company names.

Example

POST /edm/api/auth/login?userId=remote&password=remote

Response

```
{
  "status": 0,
  "message": "Optional error message",
  "data": {
    "token": "228a51d5-97e8-419b-976b-3897525aaa61",
    "companies": [
      "IRIS"
    ]
  }
}
```

Method loginToCompany

Log user into the system and immediately assign the user's company. The 2 login methods are the sole API calls to not include `token` in the header.

Parameters

`userId` - username of the user initiating the "session"

`password` - The user's password.

`companyName` - Name of user's company

Returns

Token, if the users' credentials are valid, in a hashmap containing the Token value.

Example

POST /edm/api/auth/loginToCompany?userId=remote&password=remote&companyName=IRIS

Response

```
{
  "status": 0,
  "message": "",
  "data": {
    "token": "ad9a2287-57bc-4a6c-b5ea-ea9f1ce8ee5f"
  }
}
```

Method logout

Log user out of the system. This method immediately expires the token.

Parameters

Token

Returns

Status.

Example

POST /edm/api/auth/logout

Response

```
{
  "status": 0,
  "message": "Optional error message",
  "data": null
}
```

Method setCompany

Select a company and get Connection information

Parameters

Token

companyName - Name of user's company, from company list returned by login

Returns

Connection information, if company is valid.

HashMap of Connection data

Example

POST /edm/api/auth/setCompany?companyName=IRIS

Response

```
{
  "status": 0,
  "message": "",
  "data": {
    "dNSServerName": "localhost",
    "dNSUserName": "sa",
    "dNSdatabase": "EDMWebHQ",
    "dNSConnection": "",
    "dNSPassword": "thesapassword"
  }
}
```

Method getCompanyInfo

Get Connection information associated with the provided token

Parameters

Token

Returns

Connection information - HashMap of Connection data

Example

POST /edm/api/auth/getCompanyInfo

Response

```
{
  "status": 0,
  "message": "",
  "data": {
    "dNSServerName": "localhost",
    "dNSUserName": "sa",
    "dNSdatabase": "EDMWebHQ",
    "dNSConnection": "",
    "dNSPassword": "thesapassword"
  }
}
```

Method getCompaniesConnectionStrings

Get Connection information associated with the provided token

Parameters

`userId` - sysAdmin user name

`password` - sysAdmin user's password.

Returns

Database Connection strings - HashMap of all Companies and Connection strings for their central databases

Example

POST /edm/api/auth/getCompaniesConnectionStrings

Response

```
{
  "status": 0,
  "message": "",
  "data": {
    "COMP1":
      "jdbc:sqlserver://localhost;User=user;Password=thepassword;DatabaseName=EDMWebHQComp1;responseBuffering=adaptive",
    "HQ":
      "jdbc:sqlserver://localhost;User=hquser;Password=theHQsapassword;DatabaseName=EDMwebhq;responseBuffering=adaptive",
    "ALOHA":
      "jdbc:sqlserver://localhost;User=alohaUser;Password=alohapassword;DatabaseName=EDMwebalohahq;responseBuffering=adaptive"
  }
}
```

Method isValidToken

Get Connection information associated with the provided token

Parameters

Token

Returns

Status code - either Success (for a valid token) or the reason for failure (normally an expired or otherwise invalid token).

UserId

Example

POST /edm/api/auth/isValidToken

Response

```
{
  "status": 0,
  "message": "",
  "data": {
    "userId": "remote"
  }
}
```

User Administration

Method getRoles

Get roles for either userId associated with the provided token or for an optionally provided userId

Parameters

Token

userName - Optional id of user for whom to retrieve roles

Returns

List of role (name)s for current/selected user

Example

POST /edm/api/user/getRoles

POST /edm/api/user/getRoles?userName=admin (with optional userName parameter)

Response

```
{
  "status": 0,
  "message": "",
  "data": [
    "Remote User"
  ]
}
```

Method getACL

Get all permissions this user has for any role

Parameters

Token

`userId` - Optional id of user for whom to retrieve roles

Returns

Collection of Maps of permissions for current user.

NONE permissions will override other permissions when Roles clash.

Example

POST /edm/api/user/getACL

Response

```
{
  "status": 0,
  "message": "",
  "data": [
    {
      "name": "All",
      "type": "company",
      "value": 1
    },
    {
      "name": "CDMAudit",
      "type": "data",
      "value": 1
    },
    {
      "name": "CDMPackage",
      "type": "data",
      "value": 3
    },
    {

```



```
"name": "CDMRemoteFiles",
"type": "data",
"value": 5
},
{
"name": "CDMUserCompany",
"type": "data",
"value": 1
},
{
"name": "EVENTS",
"type": "data",
"value": 31
},
{
"name": "ExportPositouchData",
"type": "function",
"value": 16
},
{
"name": "getCompanyName",
"type": "function",
"value": 16
},
{
"name": "getPropertyFileValue",
"type": "function",
"value": 17
},
{
"name": "updateLocalDirToMatchAmazonDir",
"type": "function",
"value": 17
},
{
"name": "updateLocalDirToMatchWebDir",
"type": "function",
"value": 17
},
{
"name": "updateWebDirToMatchLocalDir",
"type": "function",
"value": 17
},
{
"name": "All",
"type": "table",
"value": 0
},
{
"name": "Events__EVENTS",
"type": "table",
"value": 31
```



```
]
}
```

Method getRoles

Get this user's permissions for provided role.

Parameters

Token

`type` - of permission to find

`name` - of permission to find

Returns

Map of permissions for current user.

Example

```
POST /edm/api/user/hasPermission?type=data&name=CDMPackage
```

Response

```
{
  "status": 0,
  "message": "",
  "data": {
    "permission": "CDMPackage",
    "permissionType": "data",
    "role": null,
    "value": "3"
  }
}
```

EDM File Transactions

This section describes the EDM File Transactions feature. This feature is was implemented as part of the IRIS2W project. Click [here](#) to go to the IRIS2W home page. Click [here](#) to return to the EDM Home page.

Overview

The *EDM File Transactions* may be used to send or request files from the EDM Server or a remote site. In addition, files may be synchronized, such that when changes are made to a Central file, the revised file will be sent to the remote site, replacing the file with which it is being synchronized. Synchronization can also be performed from a remote site to the central site. A file may also be managed. This allows multiple versions of a file to be synchronized from the Central location, and a single version of a file can be synchronized with multiple remote sites.

User Actions

The following table describes the available user actions.

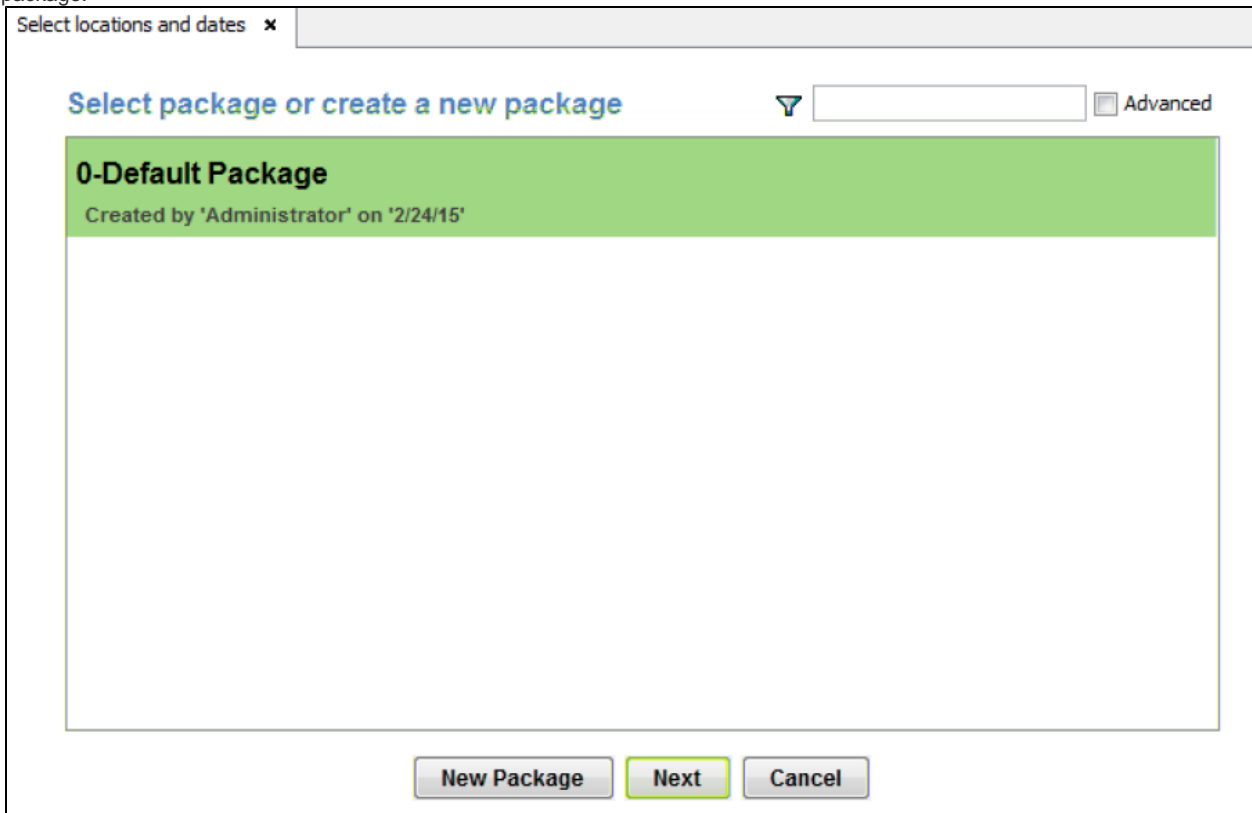
Action	What it does
Send File	Allows the user to select a local file to send to an external location
Request File	Allows the user to select an external file to copy to a local directory
Send All Remote Files	Copies all synchronized files to the external sites
Send Updated Remote Files	Copies any modified synchronized files to the external sites
Edit File Sharing	Allows the user to define multiple sites that will be synchronized from a single file on the Central location

Send File

This page describes the *Send File* action that is part of the *EDM File Transactions* feature. Click [here](#) to return to the EDM File Transactions home page.

How to Use the Send File Feature

1. On new installations, the *Send File* form is opened by clicking *Transactions – Manage Remote PC – Send File*. If this is not found on your menu, you can add it by creating a menu item where the action is 'SendFile(", ", ", 0, ", ", ")'.
2. Opening the *Send File* form will create a new tab. This may be performed at a Central or Remote site. From here, select or create a package.



The screenshot shows a web browser window titled "Select locations and dates". The main content area has a heading "Select package or create a new package" and a search filter input field with a dropdown arrow and an "Advanced" checkbox. Below this, a green box highlights the "0-Default Package" with the text "Created by 'Administrator' on '2/24/15'". At the bottom of the window, there are three buttons: "New Package", "Next" (highlighted in green), and "Cancel".

3. Specify the sites where the file will be sent. When sending from a remote site, this step is skipped as it is assumed you are sending the file to the central site.

Select locations and dates x

Select sites for package: 0-Default Package

Group by... No Groups Selected Remove Group

Find by name: Find next Find by number: Find next

Available Sites:

- Available
 - Share Groups
 - Location Groups
 - Locations

➔

➜

Selected Sites:

- Selected
 - Share Groups
 - Location Groups
 - ☑ Locations

(2) sites selected

Effective: 📅 ▼

Terminates: 📅 ▼

Force processing

Save As Package Defaults Back Finish Cancel

4. Click *Finish*. The *Send File* dialog opens.

Send File

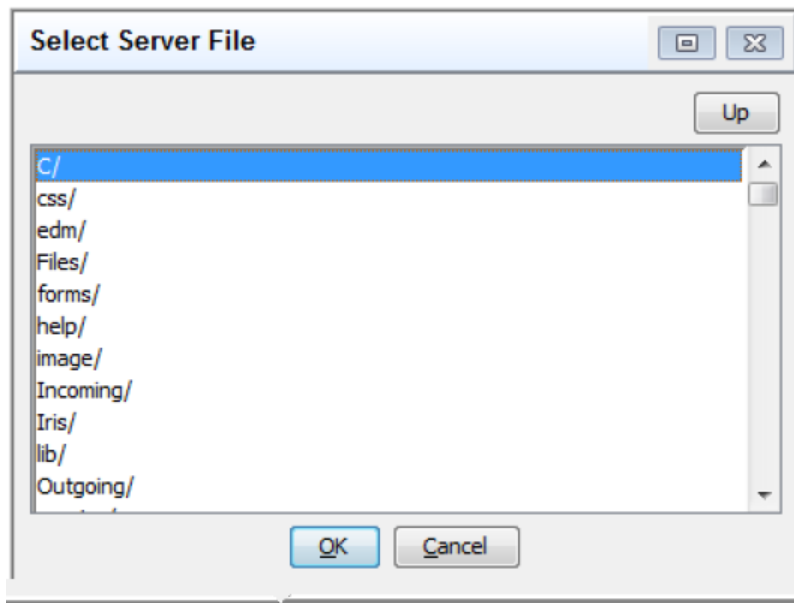
From file: 📁

To file:

Overwrite
 Synchronize

OK Cancel

5. In the *From File* field, type the file path of the file to be sent OR click the file icon to open the *Select Server File* window to select the file.



6. In the *To file* field of the *Send File* dialog, type the file path of the destination of the sent file.
7. If the file already exists at the 'To' site, the transaction will be rejected, and no copying will occur. Selecting the *Overwrite* checkbox will override this restriction, and the file will be copied regardless.
8. Selecting the *Synchronize* checkbox will cause the *From* and *To* files to be synchronized. This means that when the *Send All Remote Files* or *Send Updated Remote Files* action is performed (either manually or as a scheduled action), the 'From' file will be sent again to the 'To' site, according to the conditions of the action.
9. Click *OK* to send the file.

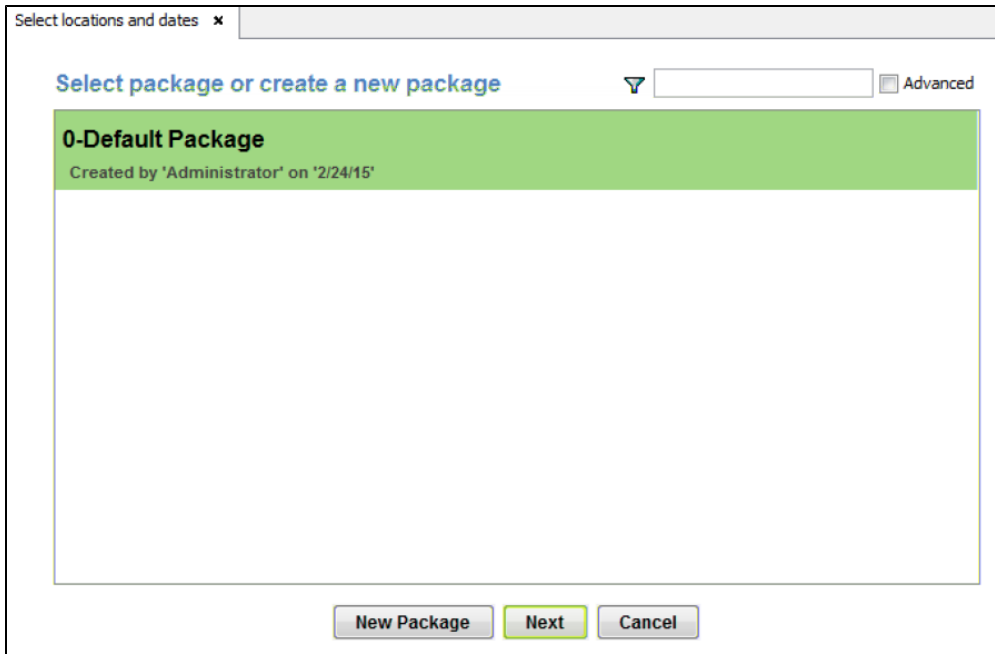
The *From file* and *To file* paths are relative to the application directory. For the central site, this is generally 'C:\Apps\Apache\Tomcat6_0\webapps\EDM'. At a remote site, it would be 'C:\EDMWeb'. A full file path may be entered explicitly, (such as 'C:\EDMWeb\css\EDM.css'). On the central site, any path that is outside the application directory, or is under the WEB-INF directory, will be rejected as inaccessible.

Request File

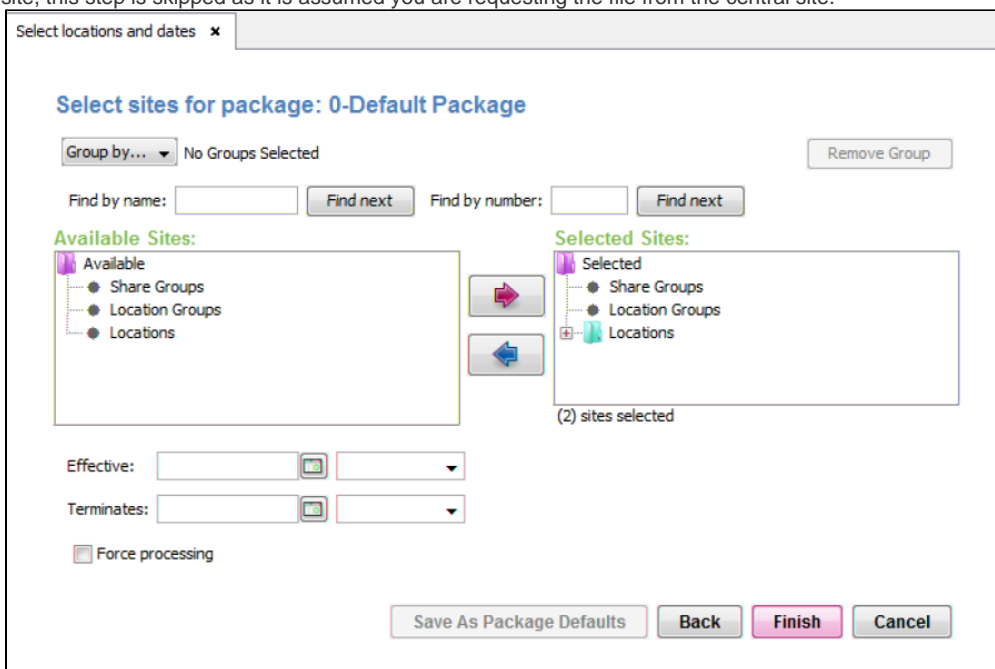
This page describes the *Request File* action that is part of the *EDM File Transactions* feature. Click [here](#) to return to the EDM File Transactions home page.

How to Use the Request File Feature

1. On new installations, the *Request File* form is opened by clicking *Transactions – Manage Remote PC – Request File*. If this is not found on your menu, you can add it by creating a menu item where the action is 'RequestFile("", "", 0, "", "", "")'.
2. Opening the *Request File* form will create a new tab. This may be performed at a Central or Remote site. From here, select or create a package.




3. Click *Next*. From the next page, specify the sites where the file from which the file will be requested. When requesting a file from a remote site, this step is skipped as it is assumed you are requesting the file from the central site.



4. Click *Finish*. The *Request File* dialog opens.

Request File

From file:

To file: 

Overwrite

Synchronize

5. In the *From file* field, type the file path of the file to be requested.
6. In the *To file* field, type the file path destination for the requested file. Selecting the file icon on the right will open a window from which you can select the desired destination.
7. If the file already exists at the 'To' site, the transaction will be rejected, and no copying will occur. Selecting the *Overwrite* checkbox will override this restriction, and the file at the site from which the file is requested will be updated.
8. Selecting the *Synchronize* checkbox will cause the *From* and *To* files to be synchronized. This means that when the *Send All Remote Files* or *Send Updated Remote Files* action is performed (either manually or as a scheduled action), the 'To' file will be sent again to the 'From' site, according to the conditions of the action. See [Managed Files](#) for more information.
9. Click *OK* to request the file.


The *From file* and *To file* paths are relative to the application directory. For the central site, this is generally 'C:\Apps\Apache\Tomcat6_0\webapps\EDM'. At a remote site, it would be 'C:\EDMWeb'. A full file path may be entered explicitly, (such as 'C:\EDMWeb\css\EDM.css'). On the central site, any path that is outside the application directory, or is under the WEB-INF directory, will be rejected as inaccessible.

Performing Request File from the Central Site

When *Request File* is performed at the central site, and there is more than one site in the selected package, the request will be refused unless the string '%REMOTESITEID%' is present in the *To file* field. What may be different versions of the same file at remote sites cannot be written to a single destination at the central site. When the string '%REMOTESITEID%' is used, it will be replaced by the site number of the remote site from which the file is requested.

Request File

From file:

To file: 

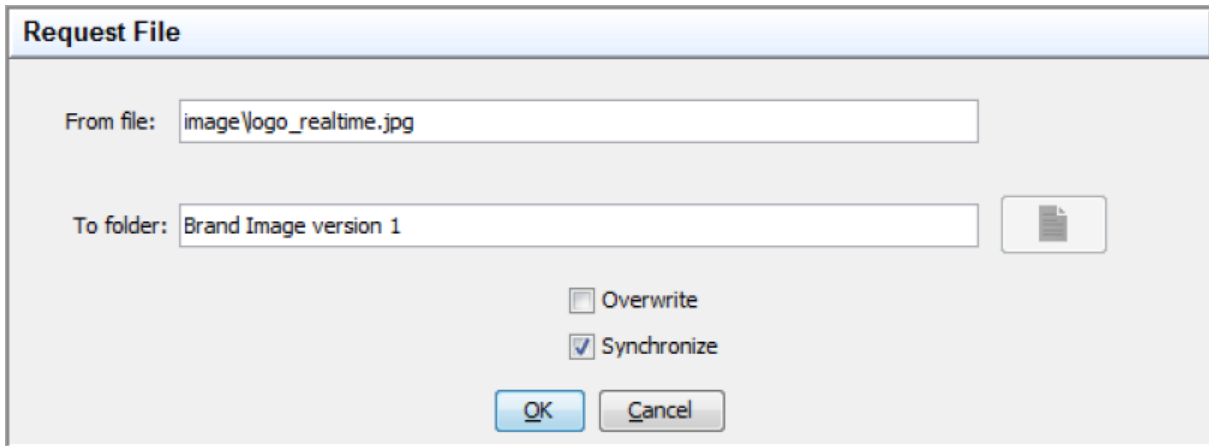
Overwrite

Synchronize

For example, suppose the selected package for this *Request File* action includes sites 1, 2, and 3. The following table specifies where the C:\EDMWeb\image\logo_xpient.jpg file that is requested from each site will be written to at the central site.

Site	Where the file is written
1	C:\Apps\Apache\Tomcat6_0\webapps\EDM\Iris\Images\test 1\logo_xpient.jpg
2	C:\Apps\Apache\Tomcat6_0\webapps\EDM\Iris\Images\test 2\logo_xpient.jpg

The following is an invalid request.



The image shows a dialog box titled "Request File". It contains two text input fields: "From file:" with the value "image\logo_realtime.jpg" and "To folder:" with the value "Brand Image version 1". To the right of the "To folder:" field is a small icon of a document. Below the input fields are two checkboxes: "Overwrite" (unchecked) and "Synchronize" (checked). At the bottom are two buttons: "OK" and "Cancel".

Configuration

Managed files are maintained in a dedicated file folder on the central site. This folder may be defined by the user. By default, this folder is a sub-folder of the folder where the application configuration file is kept. It is named "ManagedFiles". This can be changed by setting the 'managedFileFolder' property in the system configuration file to the desired folder. The property value should be the full path of the desired folder.

File Sharing

This page is related to the *Request File* action that is part of the *EDM File Transactions* feature. Click [here](#) to go the *Request File* page. Click [here](#) to return to the EDM File Transactions home page.

Edit File Sharing

Creating **Managed Files** allows for a single version of a file that is managed at the central location to be shared among multiple sites. File sharing can be defined for multiple sites with the *Edit File Sharing* action.

Opening the Edit File Sharing Form

1. On new installations, the *Edit File Sharing* form is opened by clicking *Sites – Data Sharing – Edit File Sharing*. If this is not found on your menu, you can add it by creating a menu item where the action is 'editFileSharing()'.
2. Opening the *Edit File Sharing* form will create a new tab. This may only be performed at a Central site. From here, select or create a package.

Select locations and dates x

Select package or create a new package Advanced

0-Default Package
Created by 'Administrator' on '2/24/15'

3. Click *Next*. The *Edit File Sharing* form opens. In this example, the following **Request File** action has been performed at the central site with a package for sites 1, 2 and 3.

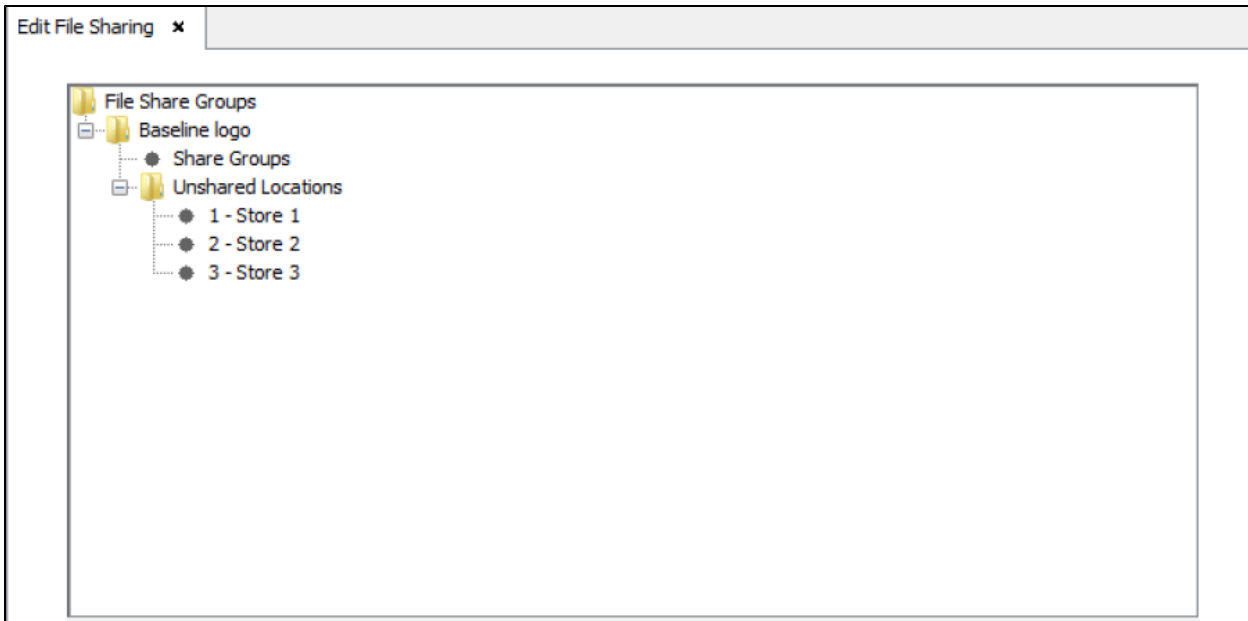
Request File

From file:

To folder:

Overwrite
 Synchronize

4. In this example, the *Edit File Sharing* form would look like this.



The file descriptors of all **Managed Files** will be listed. For each descriptor, every site that has a file synchronized to that descriptor, and was included in the selected package, will be listed. Initially, all remote files are unshared. That is, there is a one-to-one correspondence between the files at central and the files at the remote sites. The files themselves are not necessarily the same.

In this example, the Baseline logo of sites 1 and 3 looks like this:



While the site 2 logo looks like this:

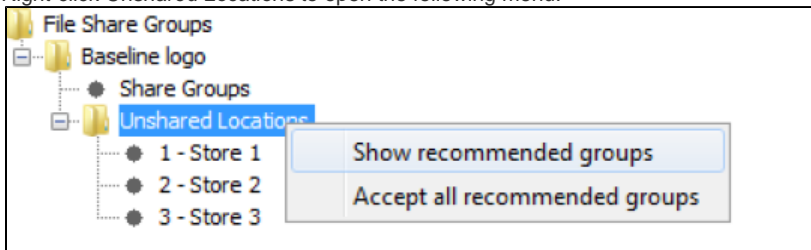


Creating Share Groups

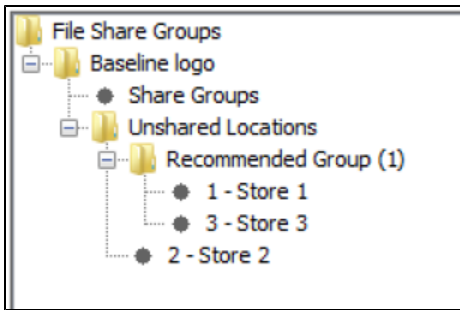
At this point, you can create Share Groups, where a single file at the central site is shared among all the remote sites in the group as long as the remote files match. This section describes the different methods for creating Share Groups.

Accepting Recommended Share Groups

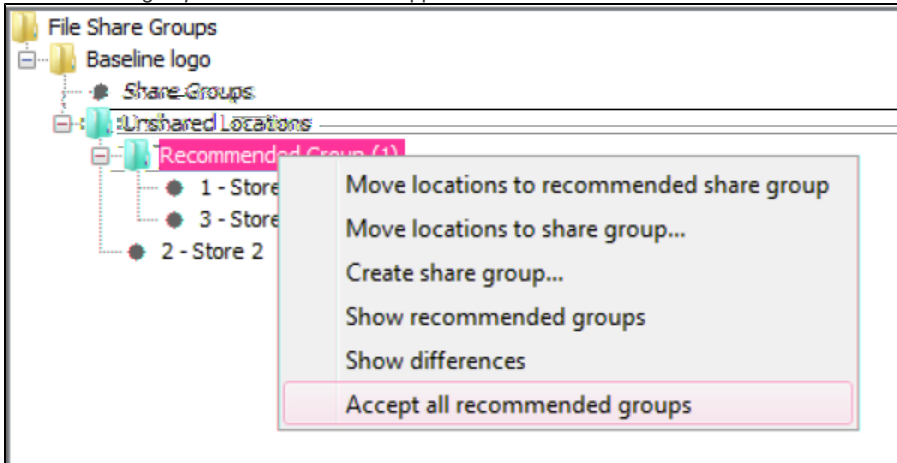
1. Right-click *Unshared Locations* to open the following menu.



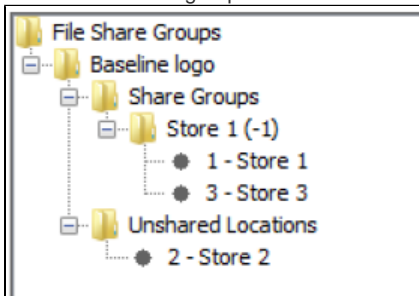
2. Select *Show recommended groups*. The recommended groups are listed.



3. In this example, remote sites 1 and 3 can be made into a group because the files are the same. Site 2 cannot be added to the group on account its file is different. To accept the recommended group, right-click *Recommended Group*, and then select *Accept all recommended groups* from the menu that appears.



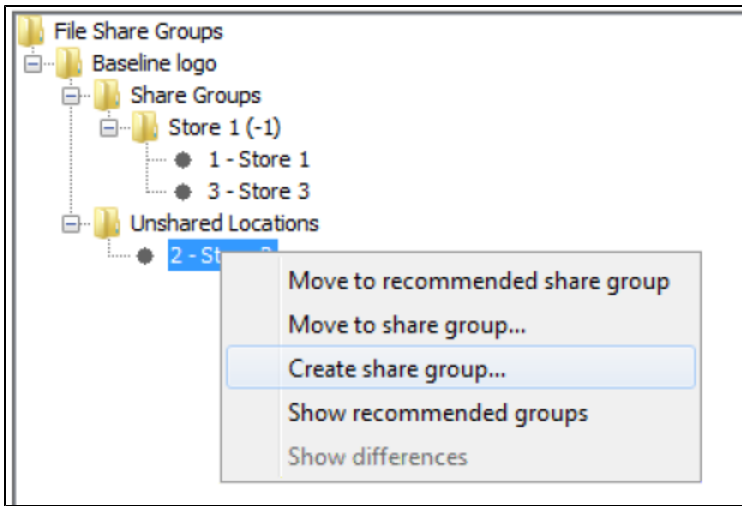
4. The recommended groups are moved to the *Share Groups* folder.



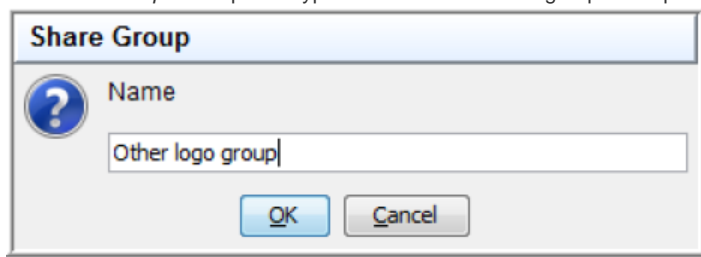
Manually Creating Share Groups

Share Groups can also be created manually. Initially, a Share Group consists of only one site. In the following example, site 2 in the *Unshared Locations* folder can be made to form a new Share Group.

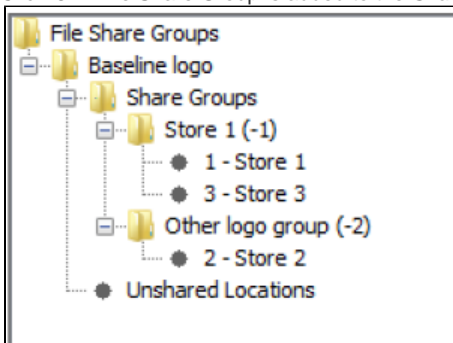
1. Right-click the site and select *Create share group* from the menu that appears.



2. The *Share Group* form opens. Type a name for the share group in the provided field.



3. Click *OK*. The Share Group is added to the *Share Groups* folder.



Working with Share Groups

This section describes the available actions you can perform when working with the items in the *File Share Groups* folder. Right-clicking an item opens a menu listing the respective actions.

File Descriptor

Show recommended groups	List the possible share groups recommended by the application
Accept all recommended groups	Accept the share groups recommended by the application

Share Groups

Show recommended groups	This only applies to existing groups. It lists groups that can be merged into a single group.
--------------------------------	---

Share group ('Store 1' in the example above)

Move locations to recommended group	Moves the locations in this group to the recommended group (this is only valid after the <i>Share Groups: Show recommended groups</i> action has been performed).
Move locations to another group	Moves the locations in this group to another existing group (so long as there are no differences in the managed files). You select the group from a pop up menu of existing groups.
Unshare locations	Moves all the locations in this group onto the unshared locations list. The group will then be deleted.
Rename share group	Allows you to enter a new name for the group.
Delete share group	Removes the share group
Show recommended groups	List the possible share groups recommended by the application
Show differences	This is currently unavailable.

Shared Location (such as 'Store 1' above)

Move to recommended share group	Moves the locations in this group to the recommended group (this is only valid after the <i>Share Groups: Show recommended groups</i> action has been performed).
Move to share group	Moves the location in this group to the recommended group (this is only valid after the <i>Share Groups: Show recommended groups</i> action has been performed).
Unshare	Moves the location to the unshared locations list
Create share group	Manually creates a share group as described above

Unshared Locations

Show recommended groups	List the possible share groups recommended by the application
Accept all recommended groups	Accept the share groups recommended by the application

Recommended Group (Recommended Group (1) above)

Move locations to recommended share group	This will move the locations to an existing recommended group.
Move locations to share group	This will move the sites in this recommendation to a selected existing group.
Create share group	This will create a share group with a user entered name, containing the recommended locations.
Show recommended groups	This will refresh the recommended group item list.
Show differences	This is currently unavailable.
Accept all recommended groups	Accept the share groups recommended by the application

Unshared location

Move to recommended share group	This will move the location to an existing recommended group.
Move to share group	This will move the location to a selected existing share group

Create share group	This will create a share group with a user entered name, containing the recommended locations.
Show recommended groups	This will refresh the recommended group item list.
Show differences	This is currently unavailable.

Synchronization

This page is related to the *EDM File Transactions* feature. Click [here](#) to return to the EDM File Transactions home page.

About Synchronization

Ordinarily, synchronized files are updated by an automatic task that is scheduled to run at regular intervals. Synchronization can also be performed manually. Two options are available when synchronizing files:

- Send All Remote Files
- Send Updated Remote Files

Send All Remote Files

On new installations, the synchronizing of all files may be performed by clicking *Transactions – Manage Remote PC – Send All Remote Files*. If this is not found on your menu, you can add it by creating a menu item where the action is 'SendRemoteFiles(false, "", "-Remote File Package", "", false, true)'.

This action may be performed at either central or remote sites. When this action is performed at the central site, all synchronized files (i.e. [Send File](#) or [Request File](#) packages for which the *Synchronize* option was selected) will be sent to the remote sites overwriting the existing files. This includes [Managed Files](#).

When the action is performed at a remote site, all synchronized files will be sent to the central site overwriting the files there. This does not include [Managed Files](#).

Send Updated Remote Files

When performing this action, only those files that have been modified since the last synchronization action will be sent.

On new installations, the synchronizing of updated files may be performed by clicking *Transactions – Manage Remote PC – Send Updated Remote Files*. If this is not found on your menu, you can add it by creating a menu item where the action is 'SendRemoteFiles(true, "", "-Remote File Package", "", false, true)'.

This action will behave in a similar fashion to the *Send All Remote Files* action described above except that only files that have been modified since the last synchronization (specifically, those files whose 'Date modified' timestamps have changed) will be sent to the synchronized site.

The EDM Build System

Building EDM is a multi step process that, when completed, results in a fully regression tested and installable package.

Jenkins

The build process is controlled by [Jenkins](#), an extensible open source continuous integration build server.

Step-by-step guide

1. Make changes to source files
 - a. If a new JAR file has been added that will be downloaded with the Applet, additional actions must be taken:
 - i. The JAR file must have a META-INF folder at the top level
 - ii. The META-INF folder must contain a MANIFEST.MF file
 - iii. The MANIFEST.MF file must contain the line "Trusted-Library: true" in order for the applet to run
 - iv. The **EDM\bat\SCM_SignJars.bat** batch file must be updated to sign the new JAR file

MANIFEST.MF

Here is an example MANIFEST.MF file:


```
Manifest-Version: 1.0
Ant-Version: Apache Ant 1.7.1
Created-By: 10.0-b22 (Sun Microsystems Inc.)
Trusted-Library: true
```

2. Check in changes to source and/or JAR files
3. The INT build will kick off automatically when it detects that changes have been checked in
 - a. The INT build will just ensure that the source files in the EDM workspace can be built
 - b. Email will be sent indicating success or failure of the INT build
 - c. If failure, return to step 1, fix the problem and try again
4. If there were changes made during the day, then the TST build will run around 2am
 - a. The TST build will run the EDM Regression Tests, which is a collection of around 250 tests that verify EDM's functionality.
 - b. Email will be sent indicating success or failure of the TST build
 - c. If success, then the TST build can be considered "releasable", as long as all of the appropriate tests have been written
5. There are 2 INT and TST build systems:
 - a. [32Bit Build](#)
 - b. [64Bit Build](#)

EDM Client Package Structure Changes

Sub packages were created in the EDMClient package to help better organize our client-related classes.

Here is the list of classes that have been moved in EDMClient and their new package structure. Click [here](#) to return to the EDM Home page.

- /edmclient/columnversioneditor/columnversioneditor.java
- /edmclient/columnversioneditor/valueupdatelistener.java
- /edmclient/ctl/updatelistener.java
- /edmclient/email/emaileditcard.java
- /edmclient/email/emaillistform.java
- /edmclient/form/addmultiplerowsdialog.java
- /edmclient/form/clientformdatasource.java
- /edmclient/form/clientformdatasourcelistener.java
- /edmclient/form/commitwizard.java
- /edmclient/form/controlwiniterator.java
- /edmclient/form/currentrowtracker.java
- /edmclient/form/findoptionsdialog.java
- /edmclient/form/formdatasourceinterface.java
- /edmclient/form/formeventlistener.java
- /edmclient/form/formwin.java
- /edmclient/form/gridwin.java
- /edmclient/form/locationpopup.java
- /edmclient/form/objectnamedialog.java
- /edmclient/form/objectwindow.java
- /edmclient/form/openobject.java
- /edmclient/form/openobjects.java
- /edmclient/form/sectionwin.java
- /edmclient/form/sitevaluemanager.java
- /edmclient/form/toolbar.java
- /edmclient/form/valuepopup.java
- /edmclient/form/valuepopupcolumnupdater.java
- /edmclient/formdesign/quickformdialog.java
- /edmclient/formplayer/formplayerdatasource.java
- /edmclient/packages/changepackagestatusdialog.java
- /edmclient/packages/editpackagecard.java
- /edmclient/packages/packagedatetimepanel.java
- /edmclient/packages/packagerenderer.java
- /edmclient/packages/routingdialog.java
- /edmclient/packages/routingwizard.java
- /edmclient/packages/selectpackagecard.java
- /edmclient/packages/selectpackagecardcreator.java
- /edmclient/packages/selectpackagetable.java

- /edmclient/packages/selectsessionpackagetab.java
- /edmclient/plugin/panasonic/keyboardeditor/multiselectlistcellitem.java
- /edmclient/plugin/panasonic/keyboardeditor/multiselectlistcellrenderer.java
- /edmclient/rowset/datarowlisttablemodel.java
- /edmclient/rowversioneditor/tablecolumnadjuster.java
- /edmclient/scheduledtask/scheduledtaskcard.java
- /edmclient/scheduledtask/scheduledtaskeditor.java
- /edmclient/scheduledtask/scheduledtasktable.java
- /edmclient/securityeditors/ldaproertyform.java
- /edmclient/sharedtableeditor/sharedtablegroupcard.java
- /edmclient/sharedtableeditor/sharedtablegroupsdialog.java
- /edmclient/site/selectsitescard.java
- /edmclient/site/simplesitesselectionpanel.java
- /edmclient/site/siteselection.java
- /edmclient/site/siteselectionlistener.java
- /edmclient/site/siteselectionpanel.java
- /edmclient/sitemanager/daterangechooser.java
- /edmclient/sitemanager/sitemanager.java
- /edmclient/sitemanager/sitemanagerconfigurationdialog.java
- /edmclient/sitemanager/sitemgrdata.java
- /edmclient/sitemanager/sitemgrsitedata.java
- /edmclient/transactionviewer/filteredlocationlist.java
- /edmclient/transactionviewer/transactionviewer.java
- /edmclient/ui/animatedbutton.java
- /edmclient/ui/appmenupanel.java
- /edmclient/ui/autocompletejcombobox.java
- /edmclient/ui/boundlistcellrenderer.java
- /edmclient/ui/buttontabcomponent.java
- /edmclient/ui/chartpanel.java
- /edmclient/ui/clientlayeredpane.java
- /edmclient/ui/Clientmodalpanel.java
- /edmclient/ui/clientpanel.java
- /edmclient/ui/clienttab.java
- /edmclient/ui/colorchooserpanel.java
- /edmclient/ui/coloredtable.java
- /edmclient/ui/coloredtablemodel.java
- /edmclient/ui/datasettablemodel.java
- /edmclient/ui/doscolor.java
- /edmclient/ui/draw.java
- /edmclient/ui/formpanel.java
- /edmclient/ui/gradientjpanel.java
- /edmclient/ui/imagecanvas.java
- /edmclient/ui/jmultilinelabel.java
- /edmclient/ui/msgbox.java
- /edmclient/ui/okcancellistener.java
- /edmclient/ui/roundedjbutton.java
- /edmclient/ui/selectfromlistdialog.java
- /edmclient/ui/selectsitespanel.java
- /edmclient/ui/simpleimageobserver.java
- /edmclient/ui/smoothjlabel.java
- /edmclient/ui/smoothjpanel.java
- /edmclient/ui/springutilities.java
- /edmclient/ui/tablefilterpanel.java
- /edmclient/ui/tableselector.java
- /edmclient/ui/updatebutton.java
- /edmclient/ui/wraplayout.java
- /edmclient/ui/beanedit/beaneditcard.java
- /edmclient/ui/beanedit/beaneditpanel.java
- /edmclient/ui/beanedit/beanform.java
- /edmclient/ui/beanedit/beanlistpanel.java
- /edmclient/ui/beanedit/beanselectiontablepanel.java
- /edmclient/ui/beanedit/beantablemodel.java
- /edmclient/ui/card/clientcard.java
- /edmclient/ui/card/listcard.java
- /edmclient/ui/card/sequencecard.java
- /edmclient/ui/card/tablecard.java
- /edmclient/ui/checklist/checkboxjlist.java
- /edmclient/ui/checklist/checklistmanager.java
- /edmclient/ui/checktree/checktreecellrenderer.java
- /edmclient/ui/checktree/checktreemanager.java
- /edmclient/ui/checktree/checktreeselectionmodel.java
- /edmclient/ui/checktree/tristatecheckbox.java

- /edmclient/ui/controller/controller.java
- /edmclient/ui/controller/controllevent.java
- /edmclient/ui/controller/controlleventtype.java
- /edmclient/ui/controller/controllerlistener.java
- /edmclient/ui/document/limiteddocument.java
- /edmclient/ui/document/limitedintegerdocument.java
- /edmclient/ui/document/limitedrangedocument.java
- /edmclient/ui/filteredlist/dualfilteredlistselectorpanel.java
- /edmclient/ui/filteredlist/filtereditemselector.java
- /edmclient/ui/filteredlist/filteredjlist.java
- /edmclient/ui/filteredlist/filteritem.java
- /edmclient/ui/listeners/focuswhenvisiblelistener.java
- /edmclient/ui/listeners/msgactionlistener.java
- /edmclient/ui/optionprompt/optionpromptpanel.java

How-to articles

Add how-to article

Title	Creator	Modified
EDM to Xenial Integration	Michael Doyle	Sep 22, 2017

EDM to Xenial Integration

Xenial is a new web-based POS system that consists of an in-store solution using web technologies and a cloud solution that acts as a central point of management system, providing services that have previously resided in the in-store back-office system. Many existing customers will migrate to the Xenial system by first changing their IRIS-based stores to connect to the Xenial cloud services, and then later replacing the in-store IRIS POS with the Xenial POS.

EDM aids this interim configuration by copying data updates made to the IRIS database to Xenial cloud services when configured to use the Xenial Connector as described in the [Configuring the Xenial Connector](#) section of the installation instructions. Administrators continue using EDM Central to edit data as they have previously; no software updates, configuration, or process changes are required on the server.

EDM Update Process with Xenial

When a Xenial client uses the EDM IRIS Menu Editor on EDM Central it updates the tables in the snapshots file under snapshot name="Menu Editor"; item changes update the snapshot name="ItemMaster" tables. When a package containing those updates is committed, transactions are created in the outgoing folder as described in the documentation for [Transactions](#).

Updates may also be sent by refreshing an entire snapshot set of tables' data on Remote utilizing the same process used when [Recovering Remote Application Data](#). Those transactions will be stored in the server's outgoing directory until the store level EDM Remote system connects to Central and receives transactions, at which time those updates are transferred to EDM Remote for processing.

On EDM Remote, a scheduled task connects to Central and pulls down all transactions waiting to be sent to the site. These transactions are then read into the CDMAudit table, where they will stay until the effective date assigned to them on Central arrives and they are processed.

When the transactions are processed, they are read back out of the CDMAudit table and any changes they provide are applied to the database; all changes successfully applied to the database are cached to be sent to Xenial once transaction processing is complete.

When all transactions have been processed, the transactions cached for Xenial are filtered to remove all transactions for tables not included in the Xenial snapshot. Each transaction is then written to a JSON file of message type dm.edm or dm.edm.tablerefresh and that file is placed into the Xenial Cloud Connector (XCC) "out" folder (defined in the configXENIALFile) for transfer to the cloud by the [XENIAL Cloud Connector](#).