

SIEMENS EDA

Xpedition™ EDM Library Guide for Administrators

Release Xpedition Enterprise 2504 - Classic
Document Revision 11.4

SIEMENS

Unpublished work. © 2024 Siemens

This Documentation contains trade secrets or otherwise confidential information owned by Siemens Industry Software Inc. or its affiliates (collectively, "Siemens"), or its licensors. Access to and use of this Documentation is strictly limited as set forth in Customer's applicable agreement(s) with Siemens. This Documentation may not be copied, distributed, or otherwise disclosed by Customer without the express written permission of Siemens, and may not be used in any way not expressly authorized by Siemens.

This Documentation is for information and instruction purposes. Siemens reserves the right to make changes in specifications and other information contained in this Documentation without prior notice, and the reader should, in all cases, consult Siemens to determine whether any changes have been made.

No representation or other affirmation of fact contained in this Documentation shall be deemed to be a warranty or give rise to any liability of Siemens whatsoever.

If you have a signed license agreement with Siemens for the product with which this Documentation will be used, your use of this Documentation is subject to the scope of license and the software protection and security provisions of that agreement. If you do not have such a signed license agreement, your use is subject to the Siemens Universal Customer Agreement, which may be viewed at <https://www.sw.siemens.com/en-US/sw-terms/base/uca/>, as supplemented by the product specific terms which may be viewed at <https://www.sw.siemens.com/en-US/sw-terms/supplements/>.

SIEMENS MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS DOCUMENTATION INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF INTELLECTUAL PROPERTY. SIEMENS SHALL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, CONSEQUENTIAL OR PUNITIVE DAMAGES, LOST DATA OR PROFITS, EVEN IF SUCH DAMAGES WERE FORESEEABLE, ARISING OUT OF OR RELATED TO THIS DOCUMENTATION OR THE INFORMATION CONTAINED IN IT, EVEN IF SIEMENS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

TRADEMARKS: The trademarks, logos, and service marks (collectively, "Marks") used herein are the property of Siemens or other parties. No one is permitted to use these Marks without the prior written consent of Siemens or the owner of the Marks, as applicable. The use herein of third party Marks is not an attempt to indicate Siemens as a source of a product, but is intended to indicate a product from, or associated with, a particular third party. A list of Siemens' Marks may be viewed at: www.plm.automation.siemens.com/global/en/legal/trademarks.html. The registered trademark Linux® is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a world-wide basis.

About Siemens Digital Industries Software

Siemens Digital Industries Software is a global leader in the growing field of product lifecycle management (PLM), manufacturing operations management (MOM), and electronic design automation (EDA) software, hardware, and services. Siemens works with more than 100,000 customers, leading the digitalization of their planning and manufacturing processes. At Siemens Digital Industries Software, we blur the boundaries between industry domains by integrating the virtual and physical, hardware and software, design and manufacturing worlds. With the rapid pace of innovation, digitalization is no longer tomorrow's idea. We take what the future promises tomorrow and make it real for our customers today. Where today meets tomorrow. Our culture encourages creativity, welcomes fresh thinking and focuses on growth, so our people, our business, and our customers can achieve their full potential.

Support Center: support.sw.siemens.com

Send Feedback on Documentation: support.sw.siemens.com/doc_feedback_form

Revision History ISO-26262

Revision	Changes	Status/Date
11.4	<p>Modifications to improve the readability and comprehension of the content. Approved by Kevin Chupp.</p> <p>All technical enhancements, changes, and fixes listed in the <i>Xpedition Enterprise Flow Release Notes</i> for this product are reflected in this document. Approved by Mike Bare.</p>	Released April 2025
11.3	<p>Modifications to improve the readability and comprehension of the content. Approved by Kevin Chupp.</p> <p>All technical enhancements, changes, and fixes listed in the <i>Xpedition Enterprise Flow Release Notes</i> for this product are reflected in this document. Approved by Mike Bare.</p>	Released September 2024
11.2	<p>Modifications to improve the readability and comprehension of the content. Approved by Kevin Chupp.</p> <p>All technical enhancements, changes, and fixes listed in the <i>Xpedition Enterprise Flow Release Notes</i> for this product are reflected in this document. Approved by Mike Bare.</p>	Released March 2024
11.1	<p>Modifications to improve the readability and comprehension of the content. Approved by Kevin Chupp.</p> <p>All technical enhancements, changes, and fixes listed in the <i>Xpedition Enterprise Flow Release Notes</i> for this product are reflected in this document. Approved by Mike Bare.</p>	Released November 2023

Author: In-house procedures and working practices require multiple authors for documents. All associated authors for each topic within this document are tracked within the Siemens documentation source. For specific topic authors, contact the Siemens Digital Industries Software documentation department.

Revision History: Released documents include a revision history of up to four revisions. For earlier revision history, refer to earlier releases of documentation on Support Center.

Table of Contents

Chapter 1

EDM Library Administration Overview.....**9**

Responsibilities of an EDM Administrator.....	9
Prerequisite Knowledge for an EDM Administrator.....	10
Command Line Syntax Conventions.....	11

Chapter 2

Software Installation and Deployment.....**13**

Tiers of the Client/Server Model.....	13
Software and Version Compatibility.....	15
Installing EDM Library Software.....	15
Deploying EDM Library Services.....	18
Post-Installation Administration.....	20

Chapter 3

User Environment.....**23**

Software Directory Structure.....	23
User Administration.....	26
Default User Groups.....	26
Access Permission Levels.....	27
Showing User Group Permissions.....	28
Local Xpedition EDM Library Cockpit Cache.....	30
Cache Contents.....	30
Cache Location.....	30
Reinitializing the Cache.....	31
Preferences Storage.....	31

Chapter 4

Core Object Class Administration.....**33**

Admin System Classes.....	33
Default Versus Customized Data Model.....	36
Object Class Administration.....	40
Database Concepts.....	41
Relational Databases.....	41
Object-Oriented Databases.....	42
Data Modeling Basics.....	42
EDM Library Data Model Naming Conventions.....	44
Object Classes Class.....	46
Opening an Object Class Information Window.....	46
Object Classes Object - Top Tab.....	48
Object Classes Object - Status Tab.....	52
Object Classes Object - Dialog Tab.....	61
Object Classes Object - Java Macros Tab.....	65

Table of Contents

Object Classes Object - Macros Tab.....	66
Object Classes Object - History Tab.....	70
Object Class Creation.....	72
Finding an Open Object Class Number and Catalog Key.....	72
Creating an Object Class.....	74
Catalog Group Administration.....	78
Opening a Catalog Group Information Window.....	80
Opening a Catalog Group Information Window From the Classification Pane.....	80
Opening a Catalog Group Information Window From the Search Results List.....	82
Catalog Group Key.....	84
Catalog Group Object - Top Tab.....	86
Catalog Group Object - Generator Tab.....	90
Catalog Group Object - Admin Tab.....	92
Catalog Group Object - GUI Tab.....	93
Catalog Group Object - Rights Tab.....	96
Catalog Group Object - History Tab.....	100
Characteristic Administration.....	103
Classes, Class Number, and Class Tables.....	104
Management Characteristics.....	105
Class Characteristics and Dynamic Characteristics.....	108
Maximum Number of Characteristics in a Table.....	109
Characteristic Types.....	112
Standard Characteristic Type.....	112
Action Button Characteristic Type.....	113
Body Property and Pin Property Characteristic Types.....	114
Bit Status Characteristic Type.....	115
List Frame Characteristic Type.....	116
Text Frame Characteristic Type.....	119
BLOB Characteristic Type.....	119
Status Administration.....	121
Default Status Values.....	121
Characteristic Visibility Through the Status Value.....	122
Hierarchical Status Management Rules.....	125
User-Dependent Status Administration.....	126
Release and Revision Control.....	129
Versioning Types.....	129
Characteristics Required for Release and Revision Control.....	130
Object Key Values For Different Version Types.....	133
Versioned Objects in a Process Workflow.....	135
Aggregated Object Handling.....	137
Key Reference Attributes.....	137
Reference Characteristics.....	138
Using Compose Mode to Edit Characteristic Definitions.....	143
Opening a Characteristic Information Window.....	146
Opening a Characteristic Information Window From Another Information Window.....	146
Opening a Characteristic Information Window From the Advanced Search Pane.....	148
Opening a Characteristic Information Window by Searching the Characteristics Object	
Class.....	150
Characteristic Object - Top Tab.....	151

Table of Contents

Characteristic Object - Search Ref Tab.....	158
Characteristic Object - Table Tab.....	161
Characteristic Object - Placement Tab.....	162
Characteristic Object - Status Tab.....	166
Characteristic Object - Rights Tab.....	185
Characteristic Object - Other Tab.....	187
Characteristic Object - History Tab.....	192
Creating a Standard Characteristic.....	193
Creating a Characteristic to View Data From a Foreign Class.....	204
Creating a Characteristic With a Value That is an Arithmetic Function.....	212
Creating an Object Reference Characteristic.....	216
Creating an Action Button Characteristic.....	218
Creating a List (List Frame Characteristics and List Columns).....	226
Creating the List Frame Characteristic.....	226
Creating the Line Key List Column.....	232
Creating Additional Columns in a List.....	237
Creating Hierarchical Lists (List Frames Within List Frames).....	240
Setting User-Based or Group-Based Characteristic Editing Permissions.....	241
Search Preset Administration.....	245
Input Pattern Checking.....	246
How EDM Library Applies Pattern Checks.....	246
Regular Input Pattern Syntax.....	247
Java Regular Expression Syntax.....	248
Opening an Input Pattern Check Information Window.....	248
Input Pattern Check Object - Top Tab.....	250
Unit Administration.....	252
Opening a Units Information Window.....	252
Units Object - Top Tab.....	253
Label Customization.....	256
Opening a Labels Information Window.....	256
Labels Object - Top Tab.....	257
Defining Hot Keys for Menu Items.....	258
The Toolbox.....	260
Default Toolboxes and Hierarchy.....	261
General Toolboxes.....	261
Tools Toolboxes.....	262
Information on Default Toolbox Tabs.....	262
Identification of a Toolbox for Program Usage.....	263
Opening a Toolbox Information Window.....	264
Toolbox Object - Top Tab.....	266
Toolbox Object - IPC Tab.....	271
Toolbox Object - Menu Tab.....	274
Toolbox Object - MetaDataMap Tab.....	275
Toolbox Object - History Tab.....	277
Part Replacement Toolboxes.....	279
Part Replacement Toolbox - Part Replacement Tab.....	279
Part Replacement Toolbox - Replacement Restrictions Tab.....	281
Databook Toolboxes.....	285
Databook Toolbox - MetaDataMap Tab.....	286

Table of Contents

Databook Toolbox - Databook Tab.....	291
Creating the Databook Toolbox Object.....	295
Quick Search Toolbox.....	301
Quick Search Toolbox - QuickSearch Tab.....	301
Creating a Quick Search Toolbox.....	303
BOM Extraction Toolbox.....	305
BOM Extraction Toolbox - Filtering Tab.....	305
BOM Extraction Toolbox - Top Tab.....	307
Update Component Info Toolbox.....	309
Compliance Management Toolboxes.....	310
Component Synchronization Toolbox.....	311
Component Synchronization Toolbox - Component Synchronization Tab.....	311
Content Provider Toolboxes.....	314
Application Configuration Toolbox - Content Provider Tab.....	314
Content Provider Configurations Toolbox - MetaDataMap Tab.....	316
Content Provider Configurations Toolbox - Content Provider Tab.....	318
Assignment Rows.....	319
History Tracking.....	323
Enabling History Tracking.....	324
Creating a History Tracking Tab.....	326
Finding Characteristics Recorded by History Tracking.....	327
Opening a History Tracking Information Window.....	328
History Tracking Object - Top Tab.....	329
MailGate Events.....	332
MailGate Events Objects.....	333
Opening a MailGate Events Information Window.....	333
Default MailGate Events Objects.....	334
MailGate Events Object - Top Tab.....	335
Filtered MailGate Events Objects.....	337
Custom Events From EDM Library Applications.....	339
Registering Users and Groups to MailGate Events.....	341
Plugin Bundles.....	346
keygen Command.....	347
Running keygen Using the GUI.....	348
Plugin Bundles Object - Top Tab.....	348

Chapter 5

Administrative Programs 351

Tools Menu Administration Programs.....	352
Unlock Manager Window.....	353
PathQuery Dialog Box - Migration Tab.....	355
PathQuery Dialog Box - Path Builder Tab.....	359
The batchadmin Program.....	362
About Class Init Files.....	362
Default Class Init Files.....	363
batchadmin Syntax.....	364
data_model_checker.....	369
domain_model_checker.....	376

Appendix A

ascl2dms Import Manager.....	377
ascl2dms Import Manager Overview.....	377
ascl2dms Input Files.....	377
Configuration File Structure.....	379
Example Configuration and Data Files.....	379
Scanners and Tokens.....	381
.CONFIG_GLOBAL Section.....	382
.CONFIG_CONVERT Section.....	383
Starting the ascl2dms Program.....	385
ASCII Loader Import Wizard Screens.....	387
ASCII Loader Import - Introduction Screen.....	387
ASCII Loader Import - Select Session Transcript File Screen.....	388
ASCII Loader Import - Settings Screen.....	389
ASCII Loader Import - Select Catalog Files Screen.....	391
ASCII Loader Import - Select Key Property Screen.....	392
ASCII Loader Import - Create Catalog Groups Screen.....	393
ASCII Loader Import - Map Characteristics Screen.....	395
ASCII Loader Import - Verify Source Data Screen.....	396
ASCII Loader Import - Import Settings Screen.....	397
ASCII Loader Import - Save Session Transcript File Screen.....	398
ASCII Loader Import - Creating Catalog Groups and Characteristics Screen.....	399
ASCII Loader Import - Importing Data Screen.....	400
ASCII Loader Import - Final Summary Screen.....	401
ascl2dms Command Line Syntax.....	402

Appendix B

XML Console	407
xml-console Command.....	408
XML Query File Format.....	415
Required XML Query File Tags.....	415
Search Restrictions in the XML Query File.....	417
Field Identifiers in the XML Query File.....	418
Example XML Query Files.....	419
XML Data File Format.....	421
General XML Data File Structure.....	421
Key Attributes for the <object> Tag (Data File).....	422
The <delete> Tag.....	423
Editing Characteristic Values in the XML Data File.....	423
Bulk Loading Documents With xml-console.....	427
Loading Documents in Bulk.....	427
Associating Documents to Components.....	431

Chapter 1

EDM Library Administration Overview

An EDM Library administrator accesses the core object classes found in the Admin module of the classification hierarchy, and uses EDM Library administrative programs.

For most administrative tasks, your user account must have library administrator group and role rights, and you must acquire a Librarian or Developer license when connecting to an EDM Server.

[Responsibilities of an EDM Administrator](#)

[Prerequisite Knowledge for an EDM Administrator](#)

[Command Line Syntax Conventions](#)

Responsibilities of an EDM Administrator

An EDM Library administrator is a privileged user that works with representatives of other user groups to make sure the EDM Library software is configured correctly. Administrators maintain system security and make sure security is not compromised by unnecessary or unrestricted user access to data objects. Because data model changes made by a library administrator can affect all users, administrator account login names and passwords should be highly guarded and not shared with other users.

An EDM Library administrator works with users and uses features found within the Core module to perform these types of tasks:

- Manage the EDM Library Services configuration on an EDM Server.
- Assist with configuring the user environment for EDM Library applications.
- Use the input of component engineering and librarian representatives to group objects into hierarchies, create and modify characteristics, create characteristic groupings on tabs, assign new characteristics to existing object classes, specify input formats, and establish rule checks.
- Manage the Toolbox objects that control the interface between the database and an external application.
- Update the server and client software trees with the latest changes, and preserve customizations so they can be carried forward from release to release.
- Assist in updating labels as part of localizing or customizing the user interfaces of EDM Library applications (if required at your company).

An EDM Library administrator who acquires a Developer license, in addition to being able to perform all administrative tasks, can also create new object classes (tables) in the database. The following are types of developer tasks:

- Create new classes and relationships, characteristics, buttons, and tabs.
- Create scripts to launch external programs and to pass arguments to those programs based on data in an EDM Library database.

- Use the API to develop custom programs and IPC interfaces to exchange data from a database with an external application.
- Test new data model extensions and interfaces.

Related Topics

Prerequisite Knowledge for an EDM Administrator

Prerequisite Knowledge for an EDM Administrator

As an EDM Library administrator, you should have a working knowledge of SQL and Oracle (or PostgreSQL) server administration concepts, but it is not usually necessary to have an Oracle database administrator (DBA) certification. You should know about general network administration procedures, Linux and Microsoft® Windows® system administration procedures, EDM Server utilities, and the EDM Web Portal.

Oracle or PostgreSQL Knowledge

If your company uses Oracle to host the database, you should be familiar with common Oracle database administration tasks and utilities, including

- Oracle architecture and terminology



CAUTION:

The EDM data model is proprietary. Unless specifically directed by a task or procedure in the documentation, never use SQL commands to directly alter table content in the data model. Tables in the data model often reference data in one or more other tables and often require a specific data format. Without a complete understanding of all the relationships between tables, rows, and columns, altering data using SQL commands could have unintended consequences, could lead to unpredictable system behavior, and could render the data model unusable.

- The structured query language (SQL), its syntax, and how to use it to create, alter, and retrieve database information
- Database partitioning and database startup and shutdown
- Oracle definition of users, roles, and privileges

If your company uses the embedded (PostgreSQL) database available with the X-ENTP software release rather than Oracle, you should be familiar with the corresponding database administration tasks and utilities as they apply to PostgreSQL. When using PostgreSQL, an EDM Library administrator should know the definition of users, roles, database owners, and database clusters.

Network Administration

You should be familiar with these general system administration concepts:

- Environment variables and startup files
- Internet protocols (in particular TCP/IP, HTTP, and HTTPS)
- Web server architecture and terminology
- Deployment and management of an EDM Server
- Creation and management of user accounts
- FLEXIm licensing server

EDM Library Knowledge

An EDM Library administrator must be familiar with the following information:

- The way EDM Library applications fit into the design and component management flows at your company
- General understanding of the tables in the EDM data model and their relationships to each other
- High-level understanding of EDM Library terminology
- EDM Library products and licensing

Related Topics

[Xpedition EDM Server and Utilities Guide \[Xpedition EDM Server and Utilities Guide\]](#)

[EDM Administrator's Guide \[Xpedition EDM Library Guide for Administrators\]](#)

[EDM Library Terminology \[Xpedition EDM Library Overview\]](#)

Command Line Syntax Conventions

Several shell commands, described throughout this manual, are available to perform administrative functions. The reference pages that describe shell commands use the same syntax conventions.

Table 1. Conventions for Command Line and Function Syntax

Convention	Example	Usage
Boldface	-configname <i>login_config</i>	A boldface font indicates a required argument.
[]	[-pack] [-blobdir <i>blob_directory</i>]	Square brackets enclose optional arguments. Do not enter the brackets.
<i>Italic</i>	-queryfile <i>query_file</i>	An italic font indicates a user-supplied argument.

Table 1. Conventions for Command Line and Function Syntax (continued)

Convention	Example	Usage
	asclId2dms -version -help -batch	The vertical bar indicates an either/or choice between items. Do not include the bar in the command.

Chapter 2

Software Installation and Deployment

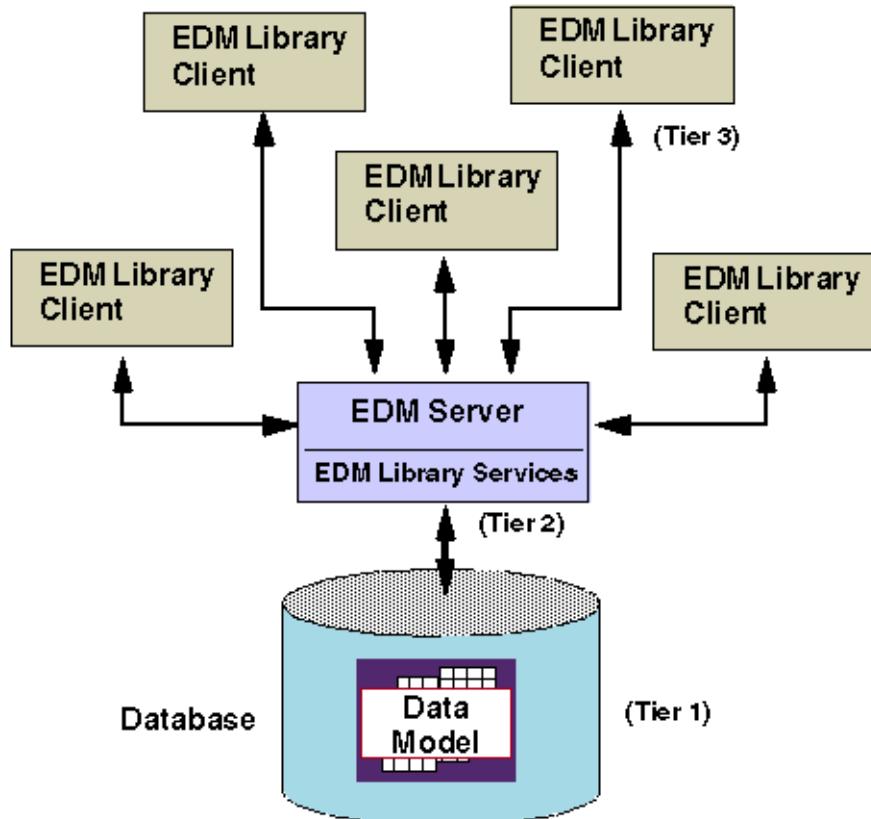
Before installing the EDM Library software, be sure you have an understanding of the client/server model and requirements. Then, proceed through installation, verification, configuration, and post-installation configuration tasks.

- [Tiers of the Client/Server Model](#)
- [Software and Version Compatibility](#)
- [Installing EDM Library Software](#)
- [Deploying EDM Library Services](#)
- [Post-Installation Administration](#)

Tiers of the Client/Server Model

Three conceptual tiers representing the database, server, and client make up a functioning EDM Library configuration.

Figure 1. EDM Database, EDM Server, and Multiple Clients



Following are the three tiers of the EDM Library client/server model:

- **Database (Tier 1)** — A relational database consisting of the EDM Library data model, configured on an Oracle or PostgreSQL server. The database receives and stores data from, and provides data to, many client applications and interfaces.



CAUTION:

The data model is proprietary. Unless specifically directed by a task or procedure, never use SQL commands to directly alter table content in the data model. Tables in the data model often reference data in one or more other tables and sometimes require a specific data format. Without a complete understanding of all the relationships between tables, rows, and columns, altering data using SQL commands could have unintended consequences, lead to unpredictable behavior, and render the data model unusable.

The database contains many data tables that make up the data model schema. The tables serve varied purposes:

- Some are always required for clients to operate correctly when connecting to the database.
 - Some hold configuration information to control the appearance of tabs, menus, characteristics, and object hierarchies in the EDM Library Cockpit application or in other client applications that access the database.
 - Others store the different types of data that a company manages. Data storage tables are grouped into modules, such as the Design module for DBOM management, and the Comp module for component management.
- **EDM Server (Tier 2)** — A server manages the communication between the database and client applications. To handle requests from EDM Library client applications, the server must be configured with EDM Library Services. EDM Library Services enable the server to support certain types of user roles and data management operations consistent with your purchased licenses.

One server could provide communication between all clients and the database for an entire company, or several servers could each provide database access for a division, building, floor, subnet, or other grouping of clients and users.

Read the *Xpedition EDM Server and Utilities Guide* for information about how to configure and deploy an EDM Server.

- **EDM Library Clients (Tier 3)** — An application, interface, or process capable of using EDM Library Services to access the database. The EDM Library Cockpit application is the general purpose tool that a design engineer, component engineer, and others use to access and modify data in the database. Examples of more targeted client applications include Parts Request Manager for originating library change requests and managing request queues, and EDM Librarian for library part development.

Through each server, a single database can receive data from and provide data to many clients at the same time. A single database might support connections to a single client, a hundred clients, or even a thousand clients, with each simultaneously accessing data. A client might run on the same physical system as the server, or on a system in a different building, in a different city, or even on a system in another country.

Software and Version Compatibility

EDM Library provides the ability to connect to many different Siemens EDA and third-party applications, depending on the product modules installed and the licenses purchased.

Installation Media

Starting with the X-ENTP VX.2.6 release, you can only install EDM client and server software from 64-bit release media.

PostgreSQL and Oracle Server Software

PostgreSQL server software is available as the default embedded database when you install and configure an EDM Server. If your company plans to use an Oracle server to host the database, you must set up the Oracle environment separately as described in “Oracle Database Setup” in the *Xpedition EDM Server and Utilities Guide* before deploying an EDM server with EDM Library services.

Siemens PCB Design Software Interfaces

PCB design software that designers use to communicate with a database must be from the latest compatible Siemens EDA software release. For more information about compatibility and using PCB applications with EDM Library software, refer to the following resources:

- *EDM Library and Design Compatibility Matrix* (available by searching the Support Center).
- *Xpedition EDM Library Guide for Designers*

.NET Framework

To create a new 3D model on any Windows 8 and later client system, the system must have the .NET framework from Microsoft. On a server system, a server administrator must have enabled the .NET framework as a Windows feature.

Related Topics

[Creating a New 3D Model Using a Template \[Xpedition EDM Library EDA Library Module Guide\]](#)

Installing EDM Library Software

EDM Library software is available for installation from the Xpedition Enterprise release media. Use the release media with the Siemens Install Program (SIP) to create one or more software trees containing the EDM Library client or server software.



Note:

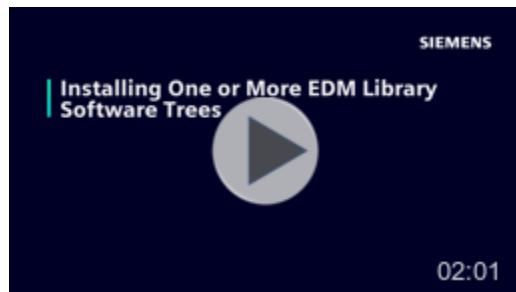
For information about using SIP for other Siemens EDA products, or to use SIP to install documentation, refer to the *Installation and Administration Manual for X-ENTP*.

Prerequisites

- You downloaded the Xpedition Enterprise 64-bit software release from Support Center or have a DVD containing the same software release. For more information, refer to “Installation” in the *Installation and Administration Manual for X-ENTP* manual.

Video

- Choose Xpedition EDM Server options.
- Choose Xpedition EDM Library and Xpedition Designer client software.



Procedure

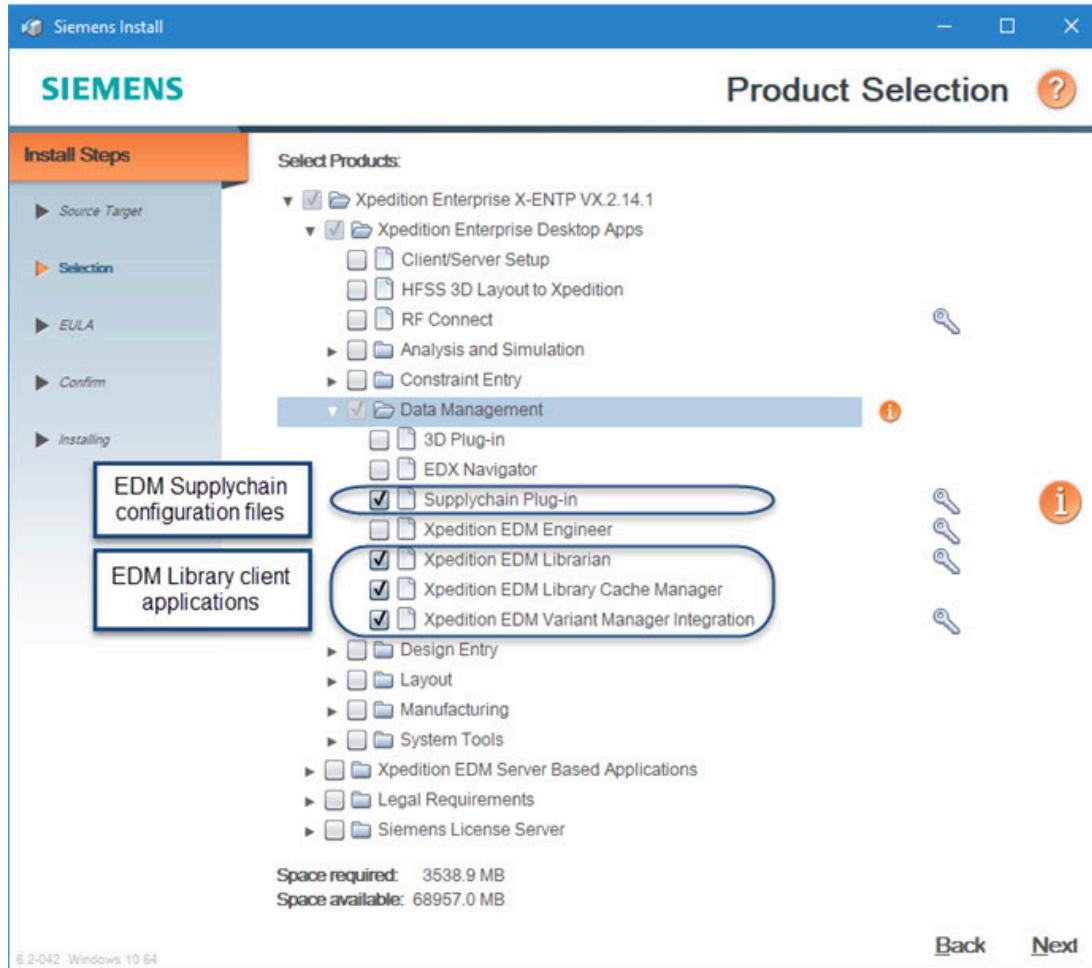
1. Launch SIP and begin the installation.
2. When you reach the Product Selection screen, expand the Xpedition Enterprise Desktop Apps > Data Management branch and select the appropriate check boxes for EDM Library client applications.

You can install the following EDM Library client applications (circled in [Figure 2](#)):

- **Supplychain Plug-in** — Installs the files that enable an administrator to configure the optional EDM Supplychain plug-in extension to EDM Library.
- **Xpedition EDM Librarian** — Installs the Xpedition EDM Library Cockpit application, EDM Librarian, the Compliance Manager application, and the interface software that lets Xpedition Designer and Databook applications work with EDM Library. Install EDM Librarian on each client system.
- **Xpedition EDM Library Cache Manager** — Installs the EDM Library Cache Manager application used to export EDA data from a database to a central library cache on the file system. Usually only librarians need access to this application.

- **Xpedition EDM Library Variant Manager Integration** — Installs the interface software so that the Xpedition Variant Manager application works with EDM Library.

Figure 2. EDM Library Client Application Product Selections



3. Expand the Xpedition EDM Server Based Applications branch and select the server software to install.

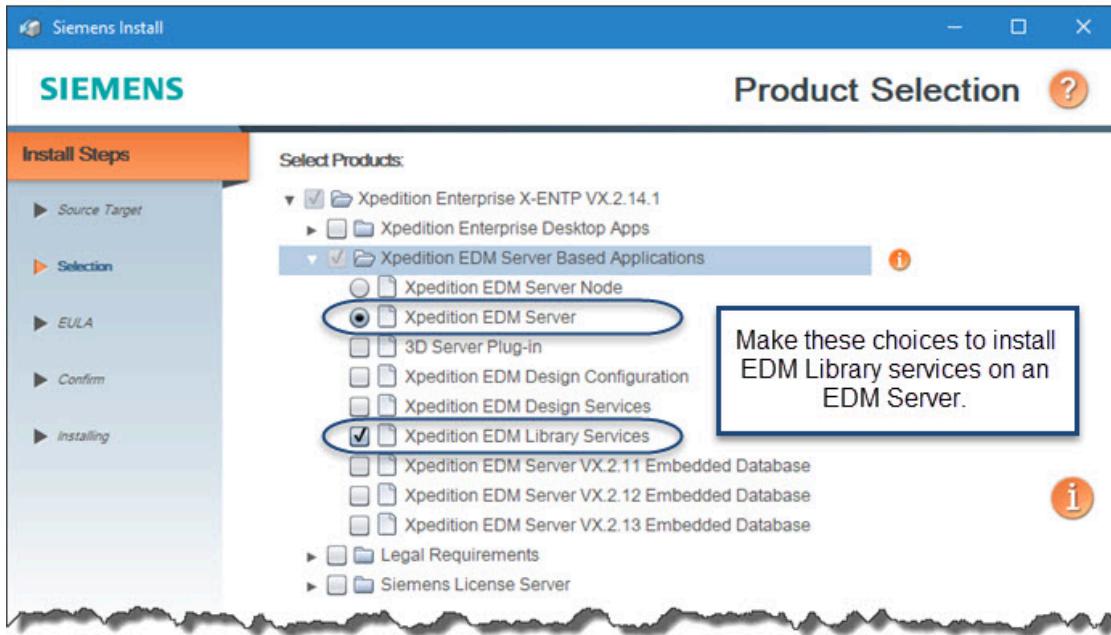


Note:

You must load the selections from the Xpedition EDM Server Based Applications branch into at least one software tree to act as a source for a server configuration and administration (a server software tree contains added programs for managing the server that are not available in a client software tree). You do not need to load the server applications into every software tree.

Select Xpedition EDM Library Services and Xpedition EDM Server to configure an EDM Server and EDM Library Services (Figure 3). After installation, you use the EDM Server Cockpit program to set the database configuration parameters before deploying the EDM Server. When distributing multiple servers across a network, also select EDM Server Node.

Figure 3. EDM Server Product Selections



4. Complete the installation.



Note:

For Linux platforms, do not exit the Install program or the Quick Install window until the post-installation processes are complete.

Results

One or more client software trees now contain the software to run EDM Library applications, and at least one server software tree contains the files to configure EDM Library Services on an EDM Server.

Related Topics

[Deploying EDM Library Services](#)

[EDM Supplychain Installation and Configuration \[Xpedition EDM Library Guide for Component Engineers\]](#)

Deploying EDM Library Services

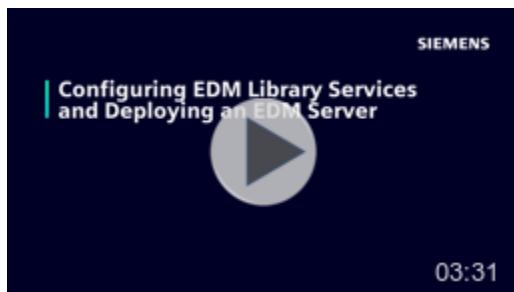
Use the EDM Server Cockpit web application to deploy EDM Library Services on an EDM Server.

Prerequisites

- You have installed EDM Server software as described in “[Installing EDM Library Software](#)” on page 15.
- You know if your company has licenses for add-on EDM Library Services data module extensions.
- You are logged into your system with a root or administrator account.

Video

- Invoke the EDM Server Cockpit web application.
- Display the EDM Library Services configuration tile.
- Set EDM Library Services configuration parameters.
- Run a pre-test.
- Deploy the EDM Server.



Procedure

1. Display the EDM Server Cockpit web page:
 - **Windows Start menu** — Choose **All Programs > Xpedition Enterprise X-ENTP VX.x > EDM Server Utilities > EDM Server Cockpit**.
 - **Windows Apps view** — Under Xpedition Enterprise VX.x, click **EDM Server Console**.
 - **Shell or Command window** — Navigate to the `<SiemensEDA-root>/SDD_HOME/EDM-Server/Utilities` directory and type **ServerCockpit**.
2. On the **EDM Server Cockpit** tab in your web browser, click **Config and Deploy**.
3. Click **EDM Library Services** to display the EDM Library Services configuration window.
Each field of the EDM Library Services configuration window provides tooltips to assist in making the correct choice. To display a tooltip, hover over the or icon.

4. Under General Settings, change any of the optional settings.
-



CAUTION:

Choose only the add-on data model extensions consistent with the functionality you plan to use. Once you deploy the server with a data model extension, you cannot easily remove the extension from the server. Changing the setting to **No** in a future data model update does not remove previously installed extensions. Therefore, you should consider data model extensions as a permanent addition to the database.

You can enable data model extensions for

- 3D models
- Process flow management
- Compliance management
- Capital library data storage

5. Optionally, use any of the following options:

- Ports to manually set an EDM Core port for EDM Library Services.
- Web Apps to add additional EDM Parts Request Manager or EDM Library Connector web application instances.
- Advanced Settings to add or change the default settings for performance tuning or error reporting.

6. Save your settings and close the EDM Library Services window.

7. Configure any other EDM Server settings, and then click the green **Review, Test, and Deploy** button.

8. In the Review Configuration popup window, make sure all settings are correct for your installation, then click **Run Pre-Tests**.

9. Make sure there are no test errors in the Pre-test Results window, then click **Deploy EDM Server**.

Related Topics

[EDM Server Configuration and Deployment \[Xpedition EDM Server and Utilities Guide\]](#)

[EDM Library Cockpit Invocation Details \[Xpedition EDM Library Overview\]](#)

Post-Installation Administration

After installing, configuring, and properly licensing EDM Library software, there are several additional administrative tasks to do before users can use the EDM Library applications to access data in the database.

Server Startup

You must deploy an EDM Server with EDM Library Services before an EDM Library client can connect to a database.

After starting the server, you must communicate the server connection naming string to users so that they can type it into the login dialog box when they invoke the EDM Library Cockpit application.

User Account Creation

User accounts provide a login name and password to connect to the EDM Server.

Users often belong to one or more user groups that establish that user's data access privileges when connected to the database. After deploying an EDM Server, the data model contains several default user accounts and user groups organized by user role, which can be used to gain familiarity with groups and user rights, or as templates to create other user accounts.

Data Model Customization

Mid-size and larger companies seldom use the default EDM Library data model in a production environment without making extensions that capture company-specific data and workflow requirements.

The following are some examples of reasons to extend the data model:

- To capture and store company-specific data not covered by the default data model object classes and characteristics, and to create a classification scheme for stored data that reflects your company's organizational scheme.
- To change labels displayed in EDM Library client applications to reflect company-specific terminology.
- To alter default toolbox definitions that define which characteristics to transfer to a design during instantiation or part replacement.
- To alter toolbox definitions that define the properties to extract from a design when generating bills of materials.

Many of these changes require using an account with administrator group or role privileges, but also selecting an Librarian or Developer license when connecting to an EDM Server.

Related Topics

[EDM Server Configuration and Deployment \[Xpedition EDM Server and Utilities Guide\]](#)

[User Administration](#)

[User Account, Group, and Role Management \[Xpedition EDM Administrator's Guide\]](#)

[Default Versus Customized Data Model](#)

Chapter 3

User Environment

EDM Library Services, when configured on an EDM Server, enable EDM Library client applications to communicate with a database on a network or corporate intranet. In addition to an EDM Server configuration, an administrator must ensure the environment on a user's system is configured appropriately.

[Software Directory Structure](#)

[User Administration](#)

[Local Xpedition EDM Library Cockpit Cache](#)

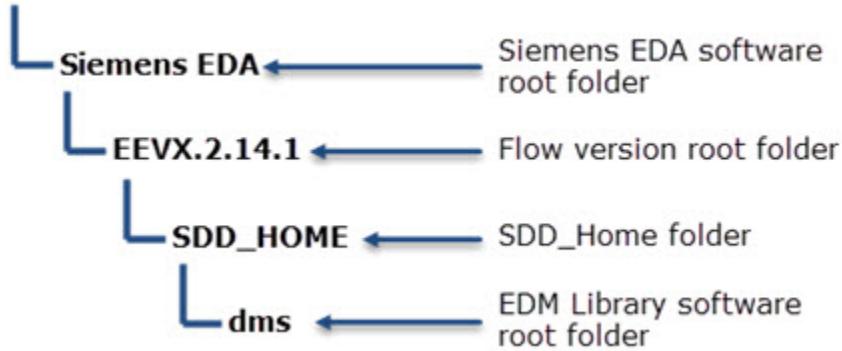
Software Directory Structure

As an administrator, you should know the general directory structure for all Siemens PCB flows, and the specific directory structure for an EDM Library software tree. That way, you are able to easily navigate the software tree to find things such as source files, configuration files, log files, programs, and scripts, and can troubleshoot problems.

General Siemens EDA Directory Structure

Figure 4 on page 23 shows the structure of a local Siemens EDA software tree. The figure shows only the branches related to EDM Library client software. Each directory might contain additional entries, depending on the installed software.

Figure 4. Siemens PCB Systems Tree Structure Example



Note:

The Install program enables you to set the name of the Siemens EDA software root directory. The names of all other directories below the Siemens EDA root are fixed by the installation process.

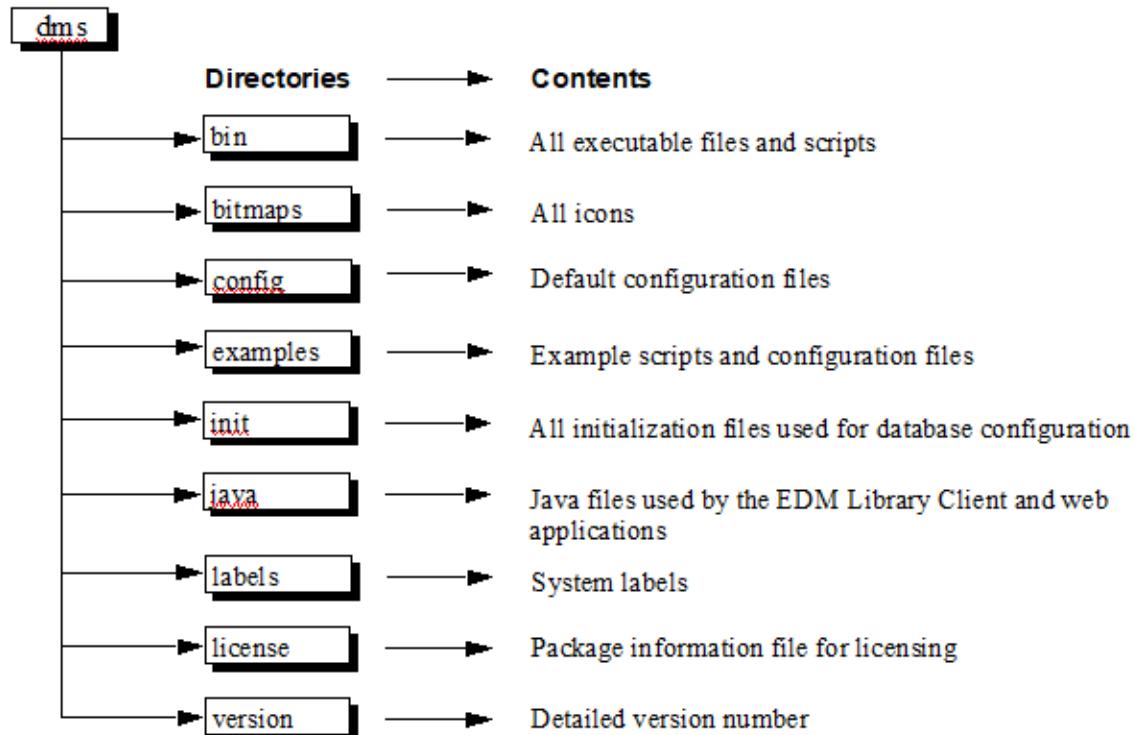
- **Siemens EDA Software Root Folder** — The root location for all Siemens EBS software residing on a system. The location is specified as the installation target when using the Install program.
- **Flow Version Root Folder** — The root folder for a particular Siemens EBS release. The flow version root folder uses a naming convention of <flow><version>. For example, you might have flow version root folders named EEVX.2.14.1, EEVX.2.14, and EEVX.2.13.
- **SDD_Home Root Folder** — The *SDD_HOME* folder holds the software trees for individual product groupings, including EDM Library client software. The names and amount of subdirectories in the *SDD_HOME* directory depend on the products from a given release that were installed.
- **EDM Library Software Root Folder** — The *dms* folder is the root location of the EDM Library software tree. To maintain a working EDM Library environment, an administrator must sometimes run applications or edit files that reside inside the EDM Library software root location.

EDM Library Directory Tree Structure

Installing EDM Library software creates the *dms* directory structure in [Figure 5](#). Do not change the names of directories or files, or move objects in the software tree, as this might cause operational problems. The *init* and *labels* subdirectories, which are used to configure the data model when deploying an EDM Server, are only present if you installed EDM Library services into the software tree.

The *dms* directory can also contain one or more platform-dependent system subdirectories with platform-dependent executables.

Figure 5. Directory/File Structure After EDM Library Software Installation



In many places throughout this manual, the location of an administrative tool or file you need to use resides within the *dms* software directory. When this is the case, a path begins with <SDD_HOME> (for example, <SDD_HOME>/dms/config) to assist with navigating to the correct location. When this convention occurs, substitute the path to the SDD_HOME directory in your Siemens EDA software tree for <SDD_HOME>. For example, <SDD_HOME>/dms/config might expand to *E:\SiemensEDA\EEVX.2.11\SDD_HOME\dms\config* on your local system.

User Administration

User administration refers to the creation and management of user accounts used to connect with the database.

When running EDM Library services on an EDM Server, an administrator uses the EDM web portal to manage user, group, and role information. For more information, refer to “User Account, Group, and Role Management” in the *Xpedition EDM Administrator’s Guide*.

[Default User Groups](#)

[Access Permission Levels](#)

[Showing User Group Permissions](#)

Default User Groups

The default user groups provide an easy way to grant class rights to a user account based on one or more general EDM Library user types. Each default group has rights that align with the general user types described in the *EDM Library Overview*.

Use the EDM web portal to view and edit default user groups.

Table 2. Default User Groups

User Group	Supported User Type
Library Administrators	Administrator
Component Engineers	Component Engineer
Designers	Design Engineer
Librarians	Librarian
Master Librarians	Librarian with Master Librarian
Manufacturers	Manufacturing Engineer
Procurements	Procurement Engineer (Purchasing)
Project Managers	Project Manager
Library Super Users	Administrator with superuser privileges
Library Users	Viewing privileges only
Library Cache Managers	Controls the right to run the EDM Library Cache Manager program

Related Topics

[Accessing the EDM Portal \[Xpedition EDM Administrator's Guide\]](#)

[User Account, Group, and Role Management \[Xpedition EDM Administrator's Guide\]](#)

[Access Permission Levels](#)

[Showing User Group Permissions](#)

Access Permission Levels

EDM Library enables an administrator to finely tune the access rights of a user or group by class, catalog, or characteristic to define what that user or group is able to do with the data stored in the database.

Each level of access represents a more specific degree of control. The user role assigned to a user account defines the rights that a user or group has to classes of objects in the database. User or group rights to catalog and Characteristic objects are specified on the catalog or Characteristic object definitions.

Level 1: Access Permissions for Classes

Class rights set the privileges pertaining to the viewing, editing, creation, deletion, copying, moving, releasing, and revisioning of objects within a specified class. An administrator uses **Account Management > Roles** in the EDM web portal to set the class rights for a specific user role.

A user can view a summary of their own class rights or the rights of the groups to which they belong by using the **Tools > Administration > User Rights View** menu item in the Xpedition EDM Library Cockpit application. The User Rights View is also useful to an administrator for viewing the class rights of all users at the same time.

Level 2: Access Permissions for Catalog Groups

To grant view or edit rights to a specific user for a specific catalog group, an administrator can use the **Rights** tab in the Catalog Group information window to override the object class access permissions displayed in Account Management > Roles in the EDM web portal.

For example, assume an electrical engineer with a username of bobbyc is a member of the group named Designers. The Designers group has view access rights that enable electrical engineers who are members of the group to use the EDM Library Cockpit for part research and part instantiation. Class rights do not permit members of the Designers group to change component information stored in the database. However, bobbyc is a leading industry expert on liquid crystal display technology. The company has decided that bobbyc is the best person to qualify LCD component data and enter it in the database. Specifying edit rights for bobbyc on the **Rights** tab of the LCD Catalog Group information window enables bobbyc to enter LCD component data into the database, while retaining only view rights for other members of the Designers group.

To return the catalog group rights for a particular user or group, use a search query.

Level 3: Access Permissions for Characteristics

Beside the access permissions for object classes and catalog groups, you can also allocate view or edit permission to a particular user for a specific characteristic. Like a Catalog Group information window, the Characteristic information window also contains a **Rights** tab with which to specify the view or edit permission to that characteristic for a particular user or group.



Note:

To grant a specific user edit rights through the **Rights** tab, the Edit Allowed status bit on the **Status Tab** must be unchecked. If checked, you cannot grant individual edit rights on the **Rights** tab.

To return the characteristic rights for a particular user or group, use a search query.

Related Topics

[Viewing User Account Rights \[Xpedition EDM Library Overview\]](#)

[Showing User Group Permissions](#)

[Catalog Group Object - Rights Tab](#)

[Characteristic Object - Rights Tab](#)

Showing User Group Permissions

To determine all of the rights to objects within a database, you must use both the EDM web portal and the Xpedition EDM Library Cockpit application to determine the full class, catalog group, and characteristic rights for a user or user group.

A user is assigned to a user group, and user group is assigned one or more user roles (which define the class rights). Some default user groups also have a predefined set of catalog and characteristic rights. For example, the Librarians group is assigned to the Librarian user role, but the Librarians group also has specific catalog and characteristic rights defined in the database. When using the EDM web portal to create a new user group that is the equivalent of an existing group, it is important to not only make the appropriate role assignment, but also to search the database for any catalog group and characteristic access permissions so that you can create those same catalog and characteristic permissions for the new group.

Prerequisites

- An administrative account that permits logging into the EDM web portal to view user, group, and user role information.
- An account that lets you log into EDM Library Cockpit with administrative privileges.

Procedure

1. Invoke a web browser and enter the URL for the EDM web portal.

The default addresses for http and https use the following formats:

`http://<server_name>:31000/xdm`

`https://<server_name>:31443/xdm`

2. When prompted, enter the administrator login name and password. After you are logged in and connected to the EDM Server, perform the following steps to view the Class rights for a particular user group:

- a. Click the **Admin** icon at the top of the EDM web page.
 - b. In the navigation pane to the left, click **Account Management**, and then click **Groups**.
The right pane specifies the roles assigned to that group.
 - c. After you know the role assignments, click **Roles** and then click the specific role name to see the class rights for that role.
3. To see if a user group has any specific catalog group or characteristic view or edit rights, perform the following steps:
-



Tip

For general information about searching in EDM Library Cockpit and how to formulate a search query, refer to “Searching and Replacing” in the *EDM Library Overview*

- a. Invoke EDM Library Cockpit and connect to the database using an account with administrator privileges.
- b. Choose **Admin > Catalog Groups** in the object classification pane or specify the path in the location bar of EDM Library Cockpit.
- c. Make sure that the search criteria pane is in advanced search mode.
- d. Click the **Rights** tab and type the name of a group in the User field. Leave the Rights field empty.
- e. Make sure the User and Rights boxes are both checked to display in the search results and click **Search**. A list of catalog groups with rights defined for the user group (if any) show in the search results.
- f. Repeat Steps b through e for the **Admin > Characteristics** object class.

Related Topics

[Accessing the EDM Portal \[Xpedition EDM Administrator's Guide\]](#)

[User Account, Group, and Role Management \[Xpedition EDM Administrator's Guide\]](#)

[Invoking EDM Library Cockpit \[Xpedition EDM Library Overview\]](#)

[Creating a Search Query and Returning a Result List \[Xpedition EDM Library Overview\]](#)

[Default User Groups](#)

Local Xpedition EDM Library Cockpit Cache

When invoking the Xpedition EDM Library Cockpit application, the server creates, and then subsequently manages, a user-specific local cache directory on the client system. Automatically caching certain information, such as *.jar* files and labels, makes invocation of the Xpedition EDM Library Cockpit application quicker because the server does not have to send the same information with each subsequent invocation.

The local cache usually requires about 30 MB or less of disk space for an initial cache, and about 15 MB of additional disk space per each additional different database to which a user connects. If a user invokes a newer version of EDM Library Cockpit, or if there have been changes to the EDM data model, the server compares the timestamps in the local cache with the timestamps on the objects being delivered, detects the timestamp in the local cache is older, and updates the local cache. Any error with the timestamps, or detection of an empty timestamp, causes the server to reload the local cache with the latest information.

[Cache Contents](#)

[Cache Location](#)

[Reinitializing the Cache](#)

[Preferences Storage](#)

Cache Contents

The server only writes certain types of data to the local cache directory, which includes the EDM data model, graphical user interface data, and labels.

Because they do not require much space (and are therefore quick to transfer), the server does not write the following types of data to the cache:

- Search presets
- Input patterns
- Units

In addition, the server also writes various log files into the local cache directory.

Cache Location

By default, a server creates the cache in the HOME directory of the user who invokes the Xpedition EDM Library Cockpit application and names the root cache directory *.DMSBrowser<version>*, where *<version>* is the Siemens EDA software version.

For example, user lindseyt has a HOME directory location of C:\Users\lindseyt. When lindseyt invokes the 10.2.13 (EEVX.2.13) version of EDM Library Cockpit for the first time, the server creates a default cache directory at C:\Users\lindseyt\.*DMSBrowser10.2.13*.



Note:

Because the *config.ini* file is located within the Siemens EDA software tree, a subsequent software installation can overwrite any changes made to the *config.ini* file.

It is usually not necessary to change the default cache location. But, if the default location is not suitable, edit the following lines in the *config.ini* file located in the <SDD_HOME>\dms\java\DMSCBrowser\configuration\ directory (the example is for the 10.2.10 version of the *config.ini* file):

```
#dms.home.dir is used by log4j config file (logs are stored in
 ${user.home}/${dms.home.dir} directory)
dms.home.dir=.DMSCBrowser10.2.13
osgi.configuration.area=@user.home/.DMSCBrowser10.2.13

#Directory for dynamic plugins. Here all bundles from database will be
 downloaded in db specific folder
mentor.dynamic.area=@user.home/.DMSCBrowser10.2.13/dynamicplg
```

Be sure to change all lines where the path to the local cache occurs.

Reinitializing the Cache

The Xpedition EDM Library Cockpit application provides items in the **File** and **Tools** menus that force the server to reinitialize the cache. You must be logged into a server with an administrative account to execute either menu item.

Menu Item	Purpose
File > Refresh Data Model	Refreshes only the local cache to see the changes in the current EDM Library Cockpit session. When making simple changes to the EDM data model (such as adding or removing a characteristic from an object class), Refresh Data Model is usually adequate to see the change.
Tools > Administration > Reload Data Model	Exits the EDM Library Cockpit application, restarts all superservices, and then restarts the EDM Library Cockpit. Reload Data Model forces a cache update of all local clients so that all users connecting to the database can see the change. Reload Data Model is necessary if Refresh Data Model does not fully update the cache to reflect the data model changes. Reload Data Model is always required to see changes when modifying class objects such as creating a new class or when changing a class status from "Under Construction" to "Approved."

Preferences Storage

Although the Xpedition EDM Library Cockpit application stores preferences locally, the storage location is not the *.DMSCBrowser<version>* cache directory. Rather, preferences reside as registry settings (in

User Environment
Preferences Storage

the Windows registry or in `$HOME/.java` on Linux), thereby enabling preferences to be persistent from session to session even with a forced update of the local cache.

Chapter 4

Core Object Class Administration

An administrator is responsible for managing objects stored in the core object classes found in the Admin module in the Xpedition EDM Library Cockpit classification pane.

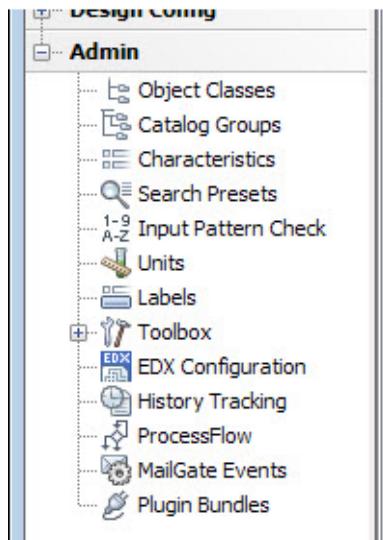
[Admin System Classes](#)
[Default Versus Customized Data Model](#)
[Object Class Administration](#)
[Catalog Group Administration](#)
[Characteristic Administration](#)
[Search Preset Administration](#)
[Input Pattern Checking](#)
[Unit Administration](#)
[Label Customization](#)
[The Toolbox](#)
[History Tracking](#)
[MailGate Events](#)
[Plugin Bundles](#)

Admin System Classes

The Admin module in the classification pane of the Xpedition EDM Library Cockpit application provides access to a set of core system classes in the database. Objects in these system classes enable EDM Library software to operate properly, and control the appearance or functionality of application clients. The Admin module is visible when selecting the Admin or Show All perspectives and when logging into an EDM Server using an account with administrator group privileges.

[Figure 6](#) shows the Admin object classes. The ProcessFlow class is only present if a database is configured with the optional Process Flow data model extension.

Figure 6. Admin System Classes



CAUTION:

Do not modify any of the system class definitions or delete default objects in these classes unless directed by a specific task topic. Misuse can result in the deletion of important data or, worse, destroy the proper operation of EDM Library software.

In a typical EDM Library configuration, most users have only view rights to the Admin classes. Except for the Search Presets class, only users with administration rights can add, delete, or edit Admin class objects.

Objects in the Admin classes serve the following purposes:

- **Object Classes** — Defines information about all object classes in the database including the class name, object ID, the table in the database that holds objects of a class, and the location of the class in the classification tree.

To add a new object class, you must be using an account with administrative privileges, and also have acquired a Developer license.

["Object Class Administration" on page 40](#) contains information about the Object Classes class.

- **Catalog Groups** — Defines the catalogs that organize objects within object classes (for example, the Components object class has catalog groups that classify and structure Component objects). The Catalog Groups class includes information about how to modify and display classification data, as well as user-dependent edit permissions for individual catalog groups.

["Catalog Group Administration" on page 78](#) contains information about how to manage objects in the Catalog Groups class.

- **Characteristics** — Contains information about characteristics that display in the search and information windows within the Xpedition EDM Library Cockpit application. The Characteristic class contains information such as

- Names and position of characteristics in a search window and in an information window
- The value type such as char, integer, double, date, and so on
- The characteristic type such as input characteristic, an action button, a pin property, a graphic property, and so on
- Table columns within the class table that store the characteristic values
- List, sort order, and characteristic status definitions
- User-dependent edit permissions for characteristics

[“Characteristic Administration”](#) on page 103 provides information about how to manage Characteristic objects.

- **Search Presets** — Stores saved search criteria for later recall. Of all the **Admin** classes, Search Preset objects are the only items a design engineer or a librarian can manage instead of an administrator.

For more information about search presets, refer to “Saving Search Criteria” in the *EDM Library Overview* and [“Search Preset Administration”](#) on page 245.

- **Input Pattern Check** — Named objects that define common input patterns.

[“Input Pattern Checking”](#) on page 246 contains more information about the Input Pattern Check class.

- **Units** —Permits the storage of an operator, range limit, and range name using a key value representing the unit (for example, “A” for “ampere”).

[“Unit Administration”](#) on page 252 contains more information about the Unit class.

- **Labels** — Stores the names assigned to labeled objects such as buttons and messages.

[“Label Customization”](#) on page 256 contains more information about the Labels class.

- **Toolbox** — Information that enables EDM Library applications and interfaces to send and receive data to and from an external EDA application or another client through an interprocess communication channel (IPC).

[“The Toolbox”](#) on page 260 contains more information about the Toolbox class.

- **EDX Configuration** — Defines the rules when exporting information to, or importing information from, a .edx package file.

“*EDX Configuration Files*” and “*Creating and Importing a Configuration File*” in the *Xpedition EDM Library Guide for Component Engineers* describes how to create configuration files and then import them into the database as EDX Configuration objects.

- **History Tracking** — Tracks changes on objects such as who modified the object, when they modified the object, the old object value(s), and the new value.

[“History Tracking”](#) on page 323 contains information about the History Tracking class.

- **MailGate Events** — Defines system events that can create an automatic notification.
“[MailGate Events](#)” on page 332 contains information about the MailGate Events class and how to configure automatic notification using the EDM Library MailGate service.
- **Bundles** — Provides a way to integrate custom plug-in applications into the **Tools** menu.
“[Plugin Bundles](#)” on page 346 contains more information about the bundles object class.

Default Versus Customized Data Model

As an administrator, you may be asked to customize the data model with classes, catalogs, and characteristics that better reflect the terminology, data organization, and data capture needs of your company. The default data model provides a starting point for such customizations.



CAUTION:

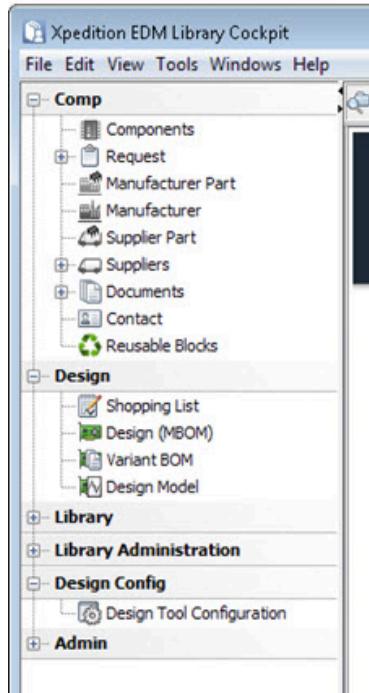
When making customizations, do not delete characteristics with a System status value or any object classes from the default data model. Doing so can break EDM Library functionality.

Example Custom Classification Hierarchy

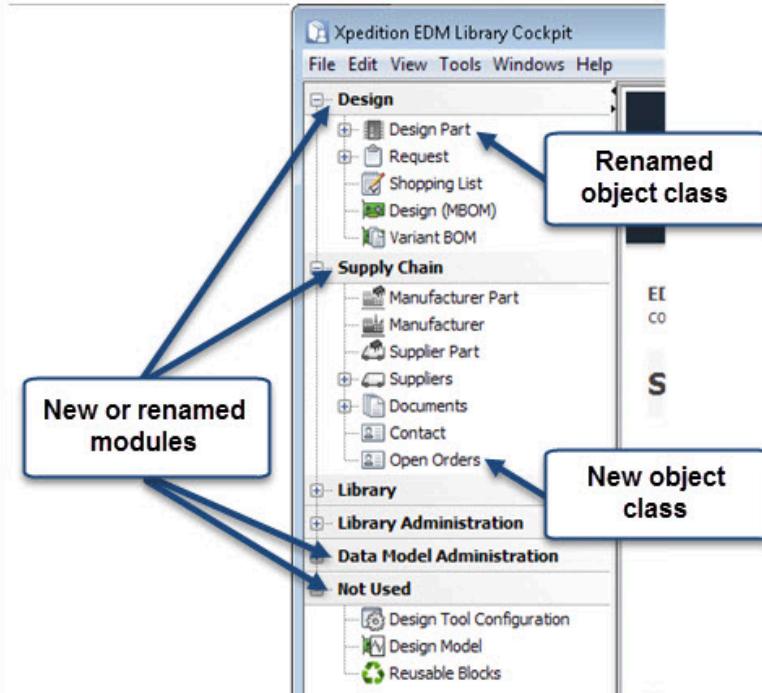
[Figure 7](#) shows the classification hierarchy in the default data model, and an example of a custom classification hierarchy obtained by editing or creating objects in the Object Classes class.

Figure 7. Customized Classification Hierarchy Example

Default Data Model



Customized Data Model



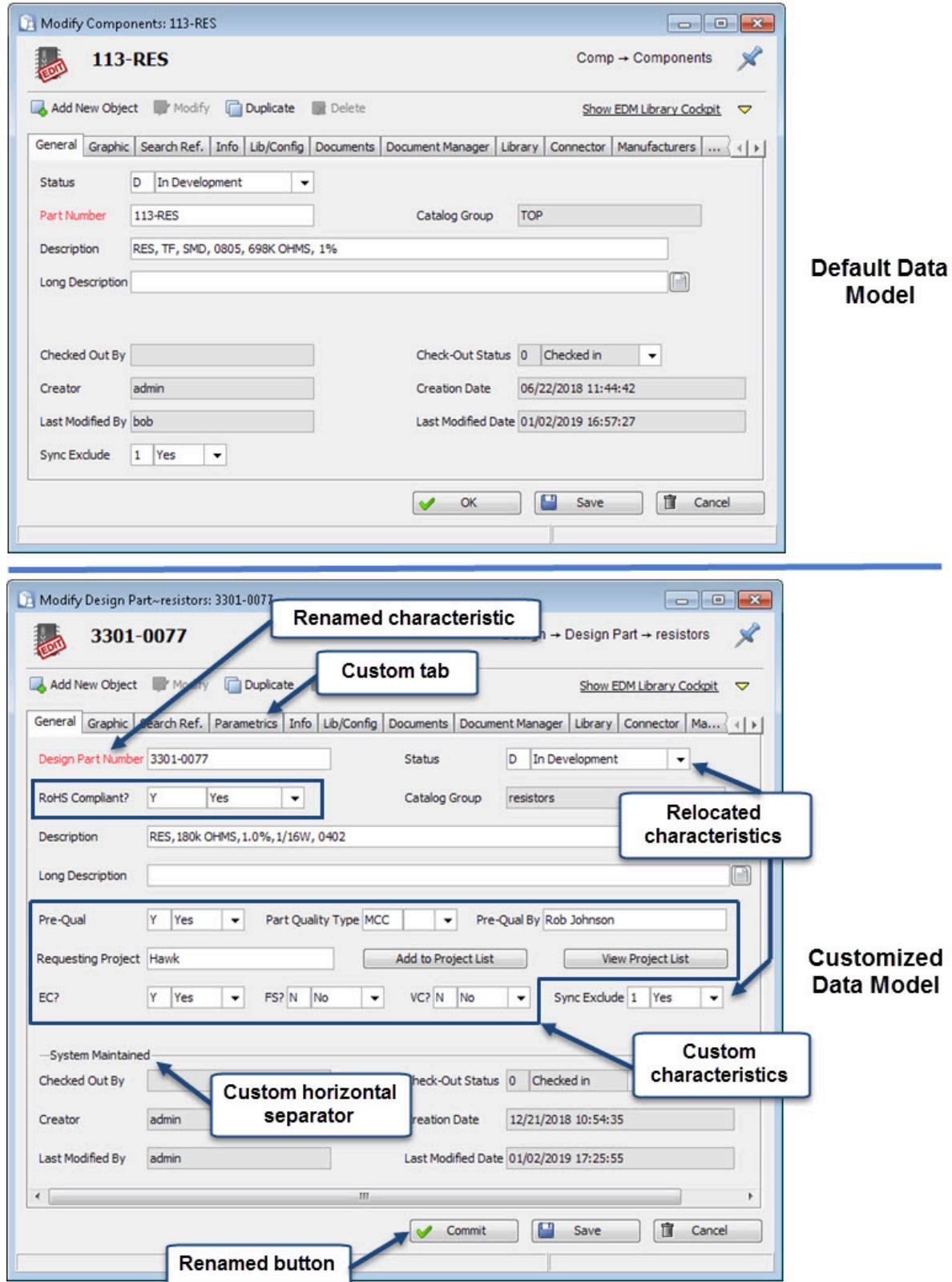
The custom classification hierarchy in [Figure 7](#) has the following modifications:

- Existing object classes from the default Comp, Design, and Design Config modules are reorganized into Design, Supply Chain, and Not Used modules.
- The Components object class is renamed Design Part.
- The list of object classes in the Library, Library Administration, and Admin modules are unchanged, but the Admin module is renamed Data Model Administration (to differentiate it from the Library Administration module).
- There is a new object class named Open Orders.

Example Custom Information Window

[Figure 8](#) compares the **General** tab of a Components object, now renamed to Design Part, in a customized data model. Changes result from editing or creating objects in the Object Classes, Characteristics, and Labels classes.

Figure 8. Customized Information Window Example



The information window in [Figure 8](#) has the following modifications:

- The Part Number characteristic is renamed to Design Part Number to align with the class name change from Components to Design Part.
- The information window contains a new **Parametrics** tab for company-specific parametric data.
- The Status and Sync Exclude characteristics are relocated from their original positions on the default **General** tab.
- The tab contains several custom characteristics that capture company-specific process data.
- The tab contains new **Add to Project List** and **View Project List** buttons.
- A horizontal separator bar now separates system-maintained information from editable information (with the exception of the Catalog Group characteristic, which is in its original location).
- The **OK** button label renamed to **Commit**. This is a global change that affects all information windows.

Related Topics

- [Object Classes Class](#)
- [Characteristic Administration](#)
- [Label Customization](#)

Object Class Administration

In EDM Library, an object class is a high-level category of objects with the same type of data. For example, Components is the name of an object class within the default EDM Library data model. Examples of object classes for other types of objects include Suppliers, Symbol, Mapping, Package, Catalog Groups, Characteristics, and Units. Object classes often, but not always, correspond to a branch below a module name in the classification pane of the Xpedition EDM Library Cockpit application. Each object class has one or more corresponding tables in the data model that store objects in that class.

EDM Library stores information about object classes in the Object Classes class, which a user with administrator or superuser authorization rights can access by opening the Admin module in the classification hierarchy and selecting Object Classes. To be able to create a new object class, you must use an account with administrator or superuser privileges and acquire a Developer license when connecting to the EDM Server.

Conceptually, EDM Library divides object classes into the following categories:

- *Regular object classes* that contain user data (such as component or EDA information) not required for the correct operation of EDM Library applications or interfaces. Users access regular object classes through categories in the classification pane other than Admin. Examples include Components, Manufacturer, Mapping, Interface, and Package. Custom classes a developer might create are also regular object classes.
- *System object classes* manage objects required for proper operation of an EDM Library application or interface (for example, the Object Classes or Characteristics object classes). Most system classes display as items in the Admin module of the classification hierarchy. Some of the system classes internally required by the database remain hidden (for example, pin classes to manage the pins for graphical objects such as interfaces or cells).



CAUTION:

Do not modify any of the system class definitions or delete default objects in these classes unless directed by a specific task topic. Misuse can result in the deletion of important data or, worse, destroy the proper operation of EDM Library software.

[Database Concepts](#)
[Object Classes Class](#)
[Object Class Creation](#)

Database Concepts

To use a Developer license to create new object classes, you need to have a basic understanding of object-oriented and relational databases.

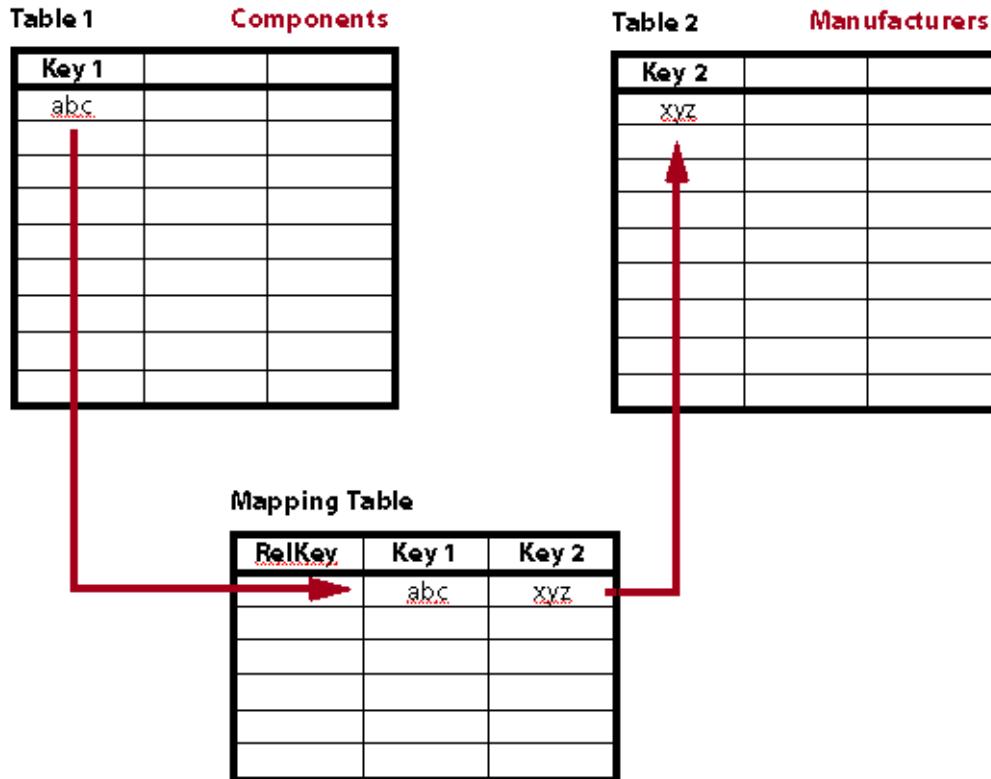
- [Relational Databases](#)
- [Object-Oriented Databases](#)
- [Data Modeling Basics](#)
- [EDM Library Data Model Naming Conventions](#)

Relational Databases

A relational database stores data in several different table, with each table having one or more columns. Rows are added as the table is populated with data. A mapping table often establishes a relationship between two tables by specifying how the values in one table relate to the values in another table.

Figure 9 shows an example of a relationship between items in the Components table and rows in the Manufacturers table. Every row in each table contains an identifier, or key, that uniquely identifies that row. For example, in the Components table, the key value uniquely identifies the component, while in the Manufacturers table, the key value uniquely identifies a manufacturer.

Figure 9. Relational Database Structure



Every component has a manufacturer and several components might come from the same manufacturer. Storing all data in one table would mean that all manufacturer information (for example, Name,

Street, ZIP, Phone, and so on) would need to be repeated in each row for each component from that manufacturer.

Storing the manufacturer information in a different table means that each dataset of each manufacturer is only stored once. Each component then contains a reference to the key value denoting the appropriate manufacturer.

An administrator is only responsible for creating the tables and reference tables if the database does not already offer functions for that purpose.

Object-Oriented Databases

Object-oriented databases have a Database Management System (DBMS) that manages all the data that can be spread across several tables. With such a management system, the user only sees an object and does not know where the data is stored (that is, in which table and tablespace).

The EDM Library software on top of Oracle or PostgreSQL is a mixture of both a relational database and object-oriented management. Oracle and PostgreSQL are relational databases, EDM Library is the management system. Once an administrator has properly set up the management system, the user does not see where the information is actually stored in the data model.

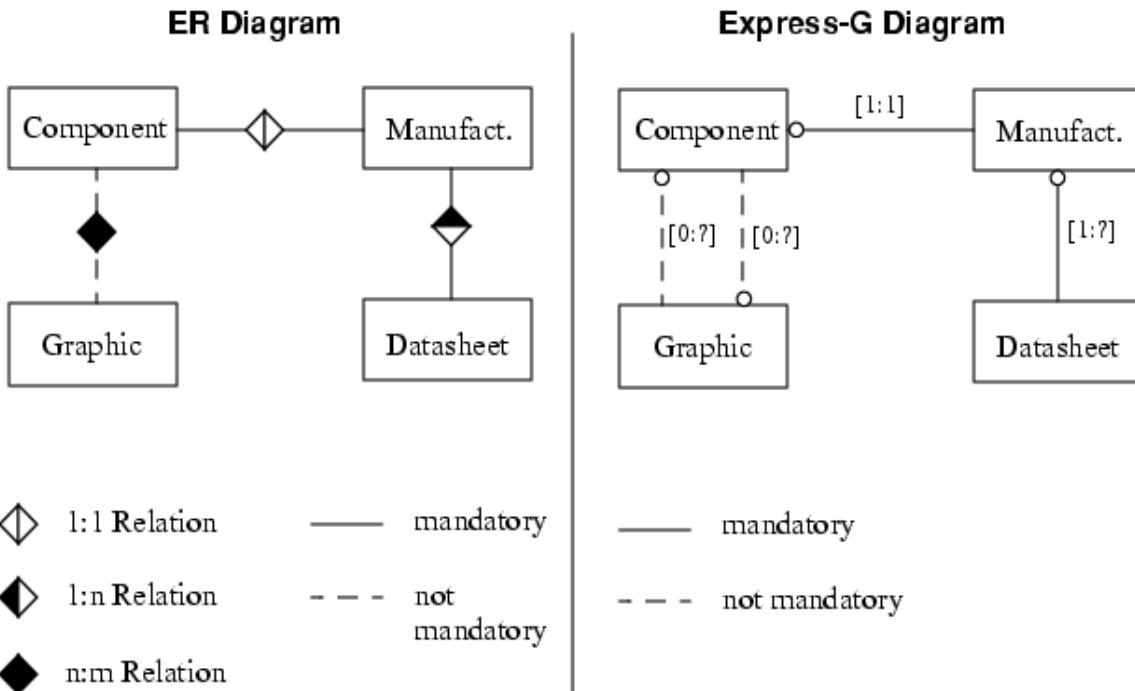
Data Modeling Basics

The EDM Library data model consists of all the tables in the database in which information associated with EDM Library resides. You can represent any data model graphically through an entity relationship diagram (ERD) or similar method. A modeling language such as UML is another way to represent a data model.

An ERD shows the object classes (tables) within the data model and the relationship between classes. In addition, a detailed ERD sometimes shows the key values and other columns within the table.

[Figure 10](#) shows two common ways to graphically represent a data model.

Figure 10. Data Models

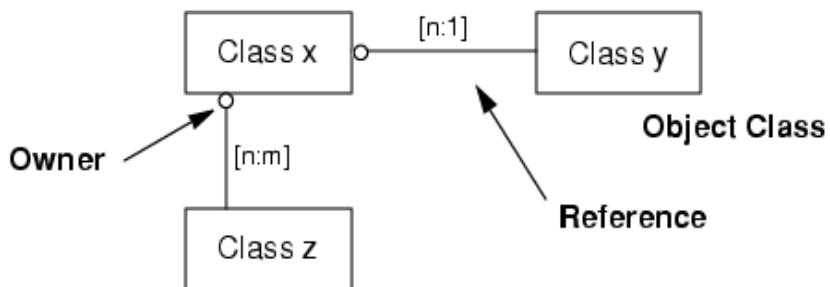


Each box represents a table within the database. Additional tables, sometimes called mapping tables, define the relationships between tables.

EDM Library Data Model

In the EDM Library data model, each box corresponds to an object class. Lists establish 1:n or n:m relations, while a single relation characteristic establishes a 1:1 relation.

Figure 11. Example Data Model Rendered in an Express-G Diagram



In the EDM Library data model, there are object classes, references, and owners. Data modeling languages offer additional extended features and syntax elements (for example, to represent inheritance and aggregation).

Object Classes

Each object class has a class table, labels, and attributes that identify the class. A main key characteristic must exist for every object class. To create a new object class, you start by creating a new object in the

Object Classes class, and then define a main key for the class. Object classes where users can create and edit objects require a set of several management characteristics.

References

A reference describes the relationship between object classes. References in the EDM Library database are set up by using reference characteristics. A single characteristic can set up a [1:1] and [n:1] reference; [n:m] relations require a reference table.

Before creating a reference characteristic, the object class and key characteristic must exist.

Owners

Each reference has an owner, marked by a circle in the data model graphic to indicate where the reference data is entered. For example, in [Figure 11](#) class x is the owner of the reference between class x and class y. Class x contains the information that associates objects of class y to each object of class x.

Related Topics

[Management Characteristics](#)

[Reference Characteristics](#)

[References \[Xpedition EDM Library Overview\]](#)

[Creating an Object Reference Characteristic](#)

EDM Library Data Model Naming Conventions

When extending the EDM Library data model with new object classes, you cannot use an SQL keyword when declaring a name for a class table. For example, you cannot name a table ADD, SET, CHECK, COUNT, or any other keyword that is reserved in SQL. Naming a table using an SQL keyword can lead to data corruption. For a list of reserved words in SQL, refer to your database documentation.

Although not required, the default data model uses a convention where most table names begin with te, tl, tr, or tw, as follows:

- te_<name> is an entity table. Tables that begin with “te” are the main class tables used to hold most objects.
- tl_<name> is a table that holds list characteristics.
- tr_<name> is a reference table (that is, a “lookup” table) that defines references to other objects.
- tw_<name> is a graphic table holding graphic primitives (lines, arcs, and so on).

Classes use the following numbering conventions:

- EDM Library reserves class numbers 0-899 for factory classes shipped by Siemens Digital Industries Software. User-defined applications can use class numbers 905-999.
- Column names within a table cannot have leading numbers, but can have trailing numbers.

A table usually represents one object class and columns within a table represent characteristics within that object class. To find which characteristics and which objects reside in which tables in the data model,

use an **Admin > Object Classes** advanced search to return the table associated with a class, and then use an **Admin > Characteristics** advanced search to return characteristics that store values in that table.

Object Classes Class

The Object Classes class stores information about object classes that are defined in the data model.

- [Opening an Object Class Information Window](#)
- [Object Classes Object - Top Tab](#)
- [Object Classes Object - Status Tab](#)
- [Object Classes Object - Dialog Tab](#)
- [Object Classes Object - Java Macros Tab](#)
- [Object Classes Object - Macros Tab](#)
- [Object Classes Object - History Tab](#)

Opening an Object Class Information Window

The Object Classes class stores information about object classes defined within the data model. Create a search query to return a list of Object Classes objects to the search result area, and then select an Object Class object and open an information window to view the contents of that object.

Prerequisites

- You invoked the Xpedition EDM Library Cockpit application and connected to an EDM Server.
- Your user account has administrator privileges.
- You acquired an Administrator license (to modify existing Object Classes objects) or a Developer license (to create new Object Classes objects).

Procedure

1. In the classification pane, open the **Admin** module and then choose **Object Classes** ([Figure 12](#)). Or, type Admin/Object Classes in the location bar.

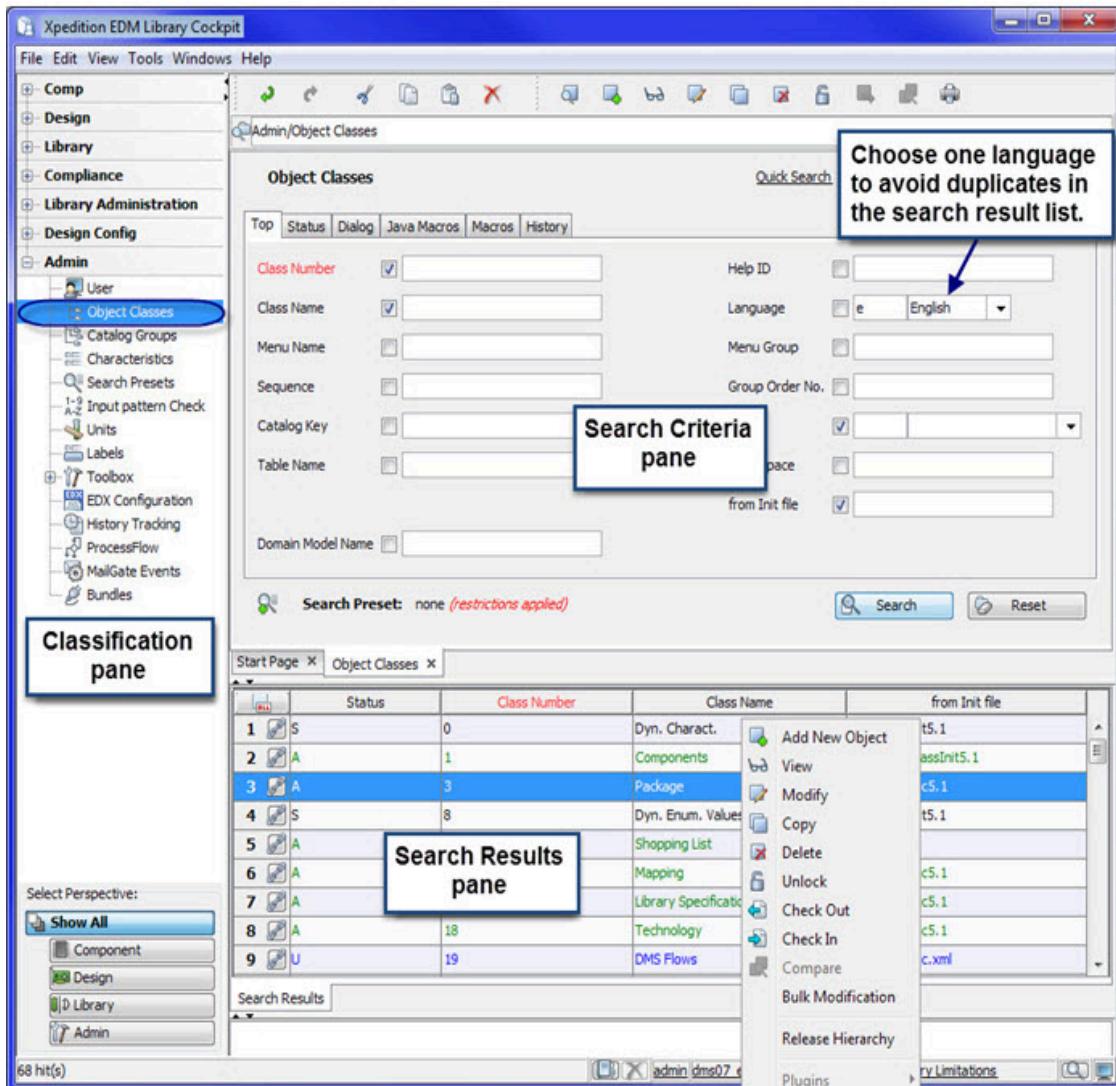
The Object Classes search criteria pane is in either quick search mode or advanced search mode, depending on your preference settings. A link at the top of the search criteria pane lets you switch between modes.



Tip

For general information about searching in EDM Library Cockpit and how to formulate a search query, refer to “Searching and Replacing” in the *EDM Library Overview*.

Figure 12. Object Class Search Window



2. Type a search query into the search criteria pane. Leave all search criteria fields empty to return a list of all class objects.

Advanced search mode enables you to enter search criteria into one or more input fields on multiple tabs and place a check mark next to the characteristics that you want as columns in the search results list.



Tip

Object classes objects with labels in more than one language can show up more than once in the search results list. To avoid showing the same class object in the search result more than once, use the advanced search pane to choose only one language to display.

3. Click the **Search** button.

4. Select a row from the search results list. Then, with the cursor still in the search results area, right-click to choose an object editing function such as **Add New Object**, **View**, or **Modify** to open the information window for the selected object class.

Alternately, click the reference button to the left of a row to open an information window in View mode without first having to select the row or use the popup menu.

An information window can either be floating or docked. Your EDM Library Cockpit preferences determine the default.

5. Use the information window to view or modify characteristic values associated with the Object Classes object.

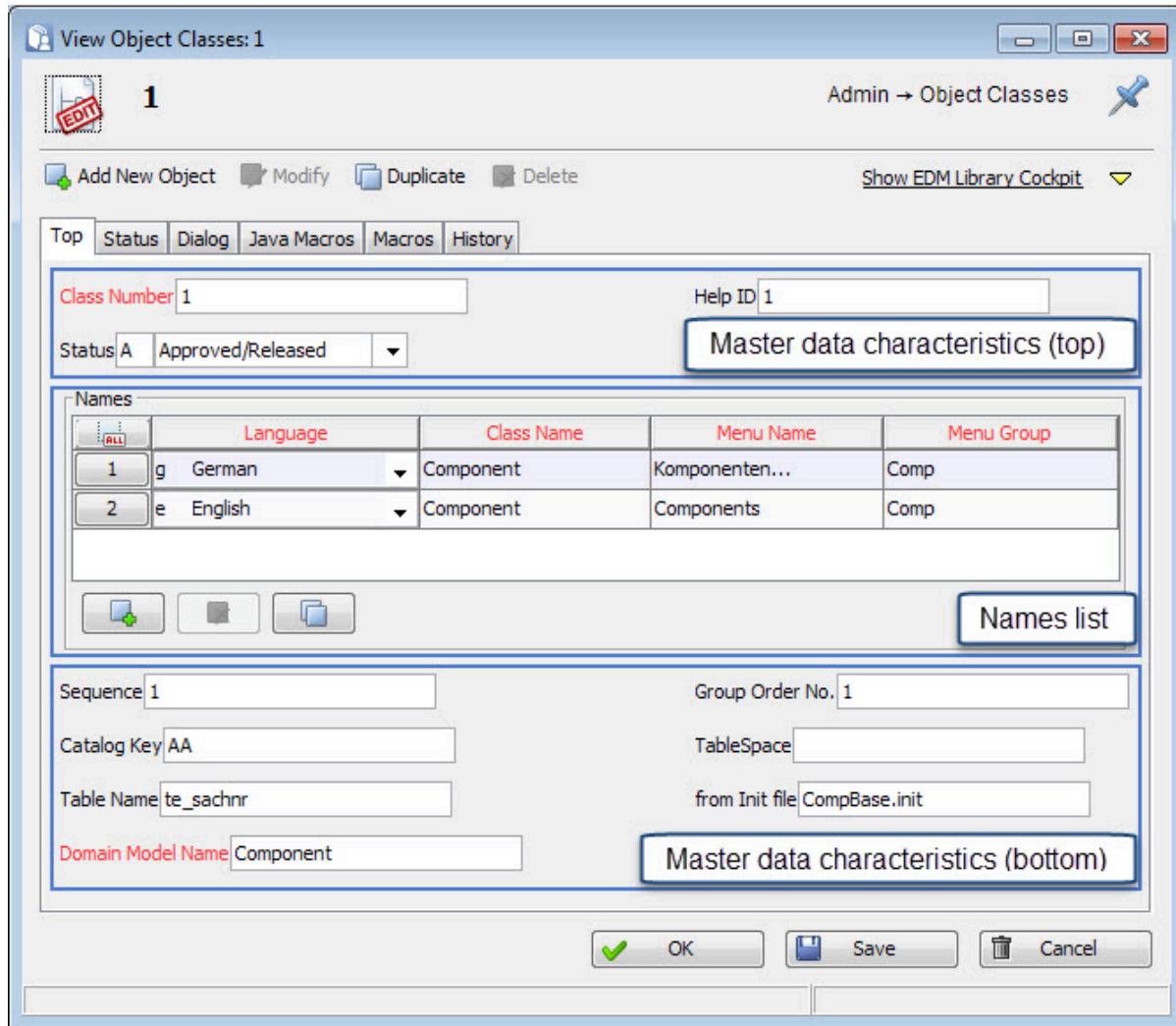
Object Classes Object - Top Tab

To access: Refer to “[Opening an Object Class Information Window](#)” on page 46.

The **Top** tab contains the master data characteristics and a names list.

Description

Figure 13. Top Tab of an Object Classes Object



Characteristics

Characteristics on the **Top** tab can be divided into:

- Master data characteristics (top)
- Names list
- Master data characteristics (Bottom)

Master Data Characteristics (Top)

- **Class Number** — A value that is unique for the class and is different from all other classes.
- **Help ID** — No longer used. The software ignores any value in this field.
- **Status** — Can be U (Under Construction), A (Approved), or S (System). You can modify classes with status U and A, but should avoid modifying system object classes (Status S).

System classes provide information critical to the functionality of the software. Classes with a status of U do not show in the classification hierarchy and typically denote an obsolete object class or an object class not yet ready for use.

Names List



Note:

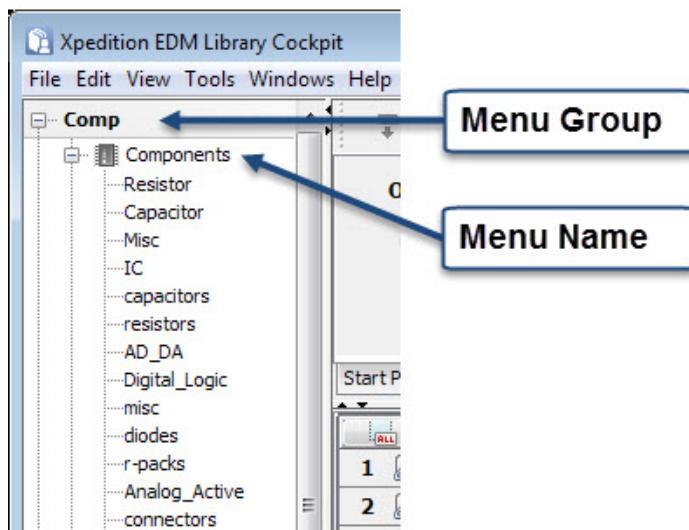
VX.2 and later releases only support English language labels and ignore other rows in the **Names** list box.

The Names list box contains columns for the class name, the pulldown menu entry, and the menu group allocated to the class for all languages defined in the database.

- **Language** — Language for displayed text
- **Class Name** — Name of the object class for the corresponding language
- **Menu Name** — Location in the classification hierarchy where a user can access to the object class inside the allocated menu group (for example, Components)
- **Menu Group** — Category in the classification hierarchy displaying the menu name (for example Comp)

[Figure 14](#) shows an example of how the Menu Name and Menu Group values display in the user interface.

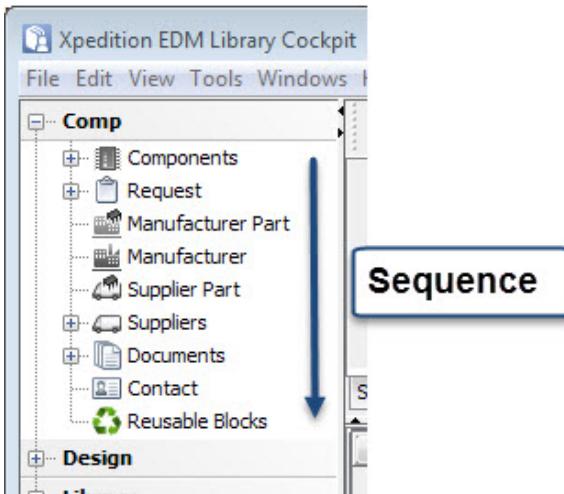
Figure 14. Menu Name and Menu Group



Master Characteristics (Bottom)

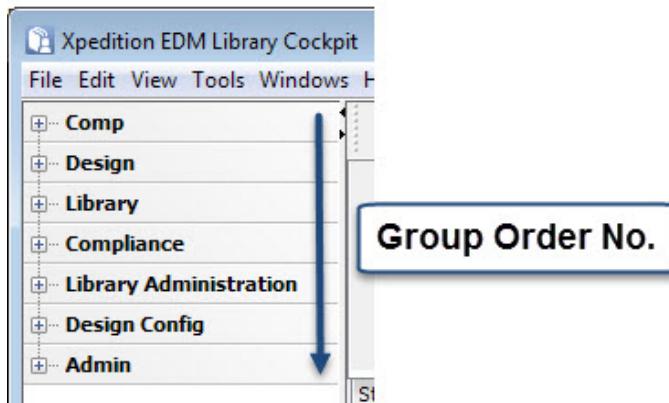
- **Sequence** — An integer that defines the position of the Menu Name within the Menu Group (Figure 15).

Figure 15. Sequence of an Object Class



- **Group Order No.** — An integer defining the order of the categories (Figure 16).

Figure 16. Group Order No. of an Object Class



- **Catalog Key** — EDM Library uses the Catalog Key value to build the catalog tree structure of an object class. The Catalog Key must contain a base key for each object class, which the software assigns to the TOP catalog of the respective object class. Two letters are usually used as a value (for example, AA, ..., zz). Use catalog keys that begin with X or x (for example Xa or xe) for custom object classes to avoid conflict with factory object classes.



CAUTION:

Although you can use the name of a tree that exists as a Catalog Key value to automatically take over that tree structure from another class, be careful! Modifications of the tree also affect the other object class that have the same tree and vice versa.

- **TableSpace**— The Oracle tablespace in which to store the class table specified by Table Name. Leaving the field empty uses the default Oracle tablespace.
- **Table Name**— The name of the object class table inside the Oracle or PostgreSQL database.
- **from Init file** — A value instructs the **batchadmin** tool to automatically write the object class definition to the initialization file named in the field. When initializing a database, the from Init file field shows which initialization file created the object class in the data model.
- **Domain Model Name** — A required value, and API program uses this name to identify the class when determining a path through the data model to a specific characteristic. Domain Model Name must contain a unique value that is not duplicated in other class objects.

Use the **domain_model_checker** command, located in the <SDD_HOME>/dms/bin directory (where <SDD_HOME> is the path to the SDD_HOME directory in your Siemens EDA software tree), to check if a Domain Model Name value is unique. The **domain_model_checker** command checks for class objects without a domain name and class objects with the same domain name, and returns a message for either condition.



Note:

Do not change the value of Domain Model Name in default object classes, as this can break functionality in client API programs offered as Siemens EDA products.

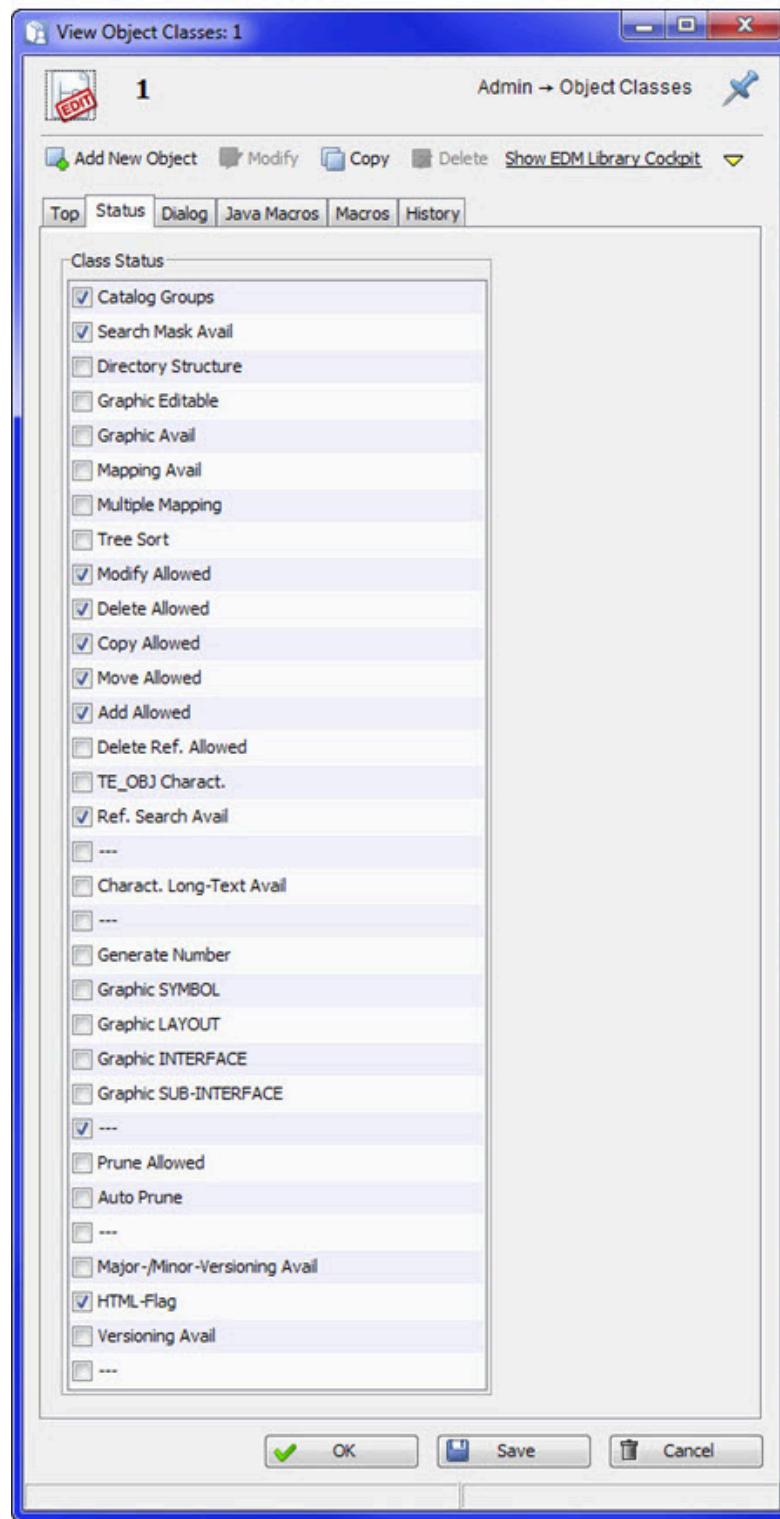
Object Classes Object - Status Tab

To access: Refer to “[Opening an Object Class Information Window](#)” on page 46.

The **Status** tab contains individual status bit characteristics that set the properties and define the behavior of the class.

Description

Figure 17. Status Tab of an Object Classes Object



Characteristics



Note:

The status bits with dashes are reserved for future use and must not be active.

- **Catalog Groups** — Permits the object class to have a classification tree with catalog groups.
 - **Search Mask Avail** — Makes a search window available for the object class. The default is unchecked, so that the object class has no search window and does not display in any classification hierarchy branch.
 - **Directory Structure** — Controls unique tree branch names. Check to indicate that a tree branch name must be unique within one hierarchy level of the tree, such as in a directory/file structure. Unchecked (the default), lets the same branch name show twice on a tree hierarchy level.
 - **Graphic Editable** — Enables editing of the graphical object in EDM Library Cockpit.
-



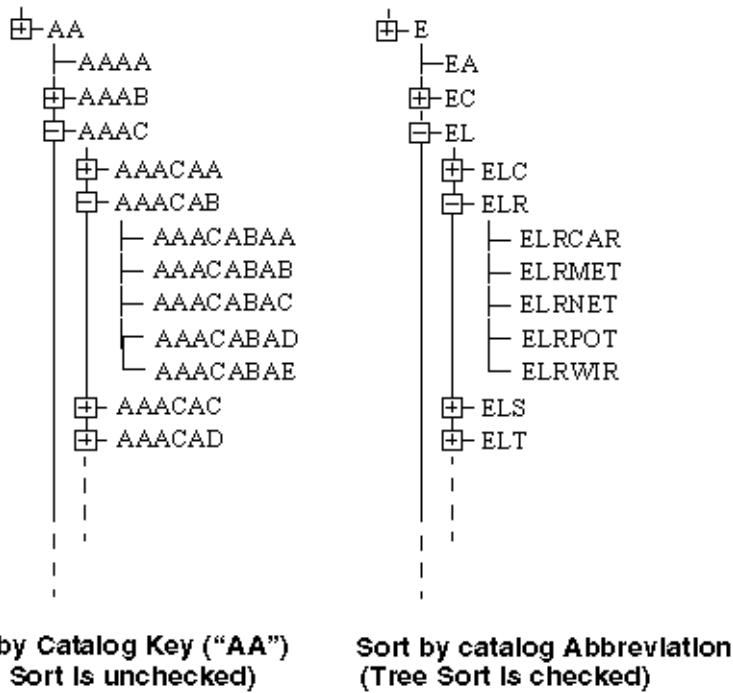
CAUTION:

Always uncheck the Graphic Editable status bit. Editing graphics inside EDM Library Cockpit can result in damaged graphics when exporting the object to the file system.

- **Graphic Avail** — Indicates that objects in this class have graphical data (for example, a symbol or cell), which EDM Library Cockpit displays in a **Graphic** tab in the object information window.
- **Mapping Avail** — Objects in this class have mapping data (for example, an assignment between pins such as physical and logical pins or top interfaces and sub interfaces). The EDM Library Cockpit displays mapping data on a **Mapping** tab in the object information window. The Mapping Avail status bit is only applicable to the Mapping object class (class number 10).
- **Multiple Mapping** — A Mapping object can contain multiple sub-interfaces on the right side of the Mapping table. Unchecked (the default) means that only one sub interface can be assigned.
- **Tree Sort** — Sorts the classification tree of this object class alphabetically according to the catalog group identifier. Unchecked (the default) sorts the tree by the catalog key (that is, according to the creation date and time of the tree branches).

[Figure 18](#) shows an example of possible catalog abbreviations to achieve an alphabetical tree sort.

Figure 18. Possible Tree Sorts



- **Modify Allowed, Delete Allowed, Copy Allowed, Move Allowed, and Add Allowed** — Controls the object editing functions in the search result popup menu of the class. The settings override all settings specified for individual users or user groups:
 - **Modify Allowed** — Activates the **Modify** popup menu item.
 - **Delete Allowed** — Activates the **Delete** popup menu item.
 - **Copy Allowed** — Activates the **Duplicate** popup menu item.
 - **Move Allowed** — Activates the **Move** popup menu item.
 - **Add Allowed** — Activates the **Add New Object** popup menu item.
- **Delete Ref. Allowed** — Enables a user to delete objects in this class even when they are referenced by other database objects. Unchecked (the default), displays an error message when trying to delete an object referenced by another object.
- **TE_OBJ Charact** — Automatically creates object characteristics for this class in the te_obj table, which are required by some functionality. Examples include the <class>obj_lock and <class>obj_user characteristics for object locking that exist in the user_name and lck_sts table columns of the class table.

The software creates the object characteristics at run time or when using **batchadmin** program to initialize a database with class information from an initialization file. Object characteristics created at runtime exist in memory to provide certain functionality and might not always be visible in EDM Library Cockpit as Characteristic objects associated with a particular class.

Not checking TE_OBJ means that an administrator or developer must define object characteristics manually.



Note:

When checking TE_OBJ Charact, you must also set the Class_No. flag inside the **Status** tab of the corresponding <class>obj_id characteristic for this object class.

- **Ref. Search Avail** — Creates a **Search Ref** tab on the information window of objects in this class.
 - **Charact. Long Text Avail** — Creates a **Text Information** tab on the information window of objects in this class. The **Text Information** tab has a text box that can hold up to 32,000 lines of text.
 - **Generate Number** — Automatically generates a unique identification number for a any new object created in the class. Checking requires also checking the Catalog Groups status bit. Unchecked (the default), requires a user to enter an identification number manually when creating a new object.
For information about the number generator, refer to “[Catalog Group Object - Generator Tab](#)” on page 90.
 - **Graphic SYMBOL, Graphic LAYOUT, Graphic INTERFACE, and Graphic SUB-INTERFACE** — No longer used and have no effect in the EDM Library Cockpit application.
 - **Prune Allowed** — Enables the Prune functionality used with major/minor versioning. Prune deletes specific minor versions of an object as specified by the user in the Prune window.
 - **Auto Prune** — Enables the auto prune functionality to automatically prune all objects with minor versions when creating a new major version of the object or revisioning the last minor version to a new major version. You can only use Auto Prune with the major/minor versioning functionality.
 - **Major-/Minor-Versioning Avail** — Enables major/minor versioning for objects in the class. When checked, saving the object class creates the required versioning characteristics in [Table 3](#) and migrates any existing data.
-



Note:

Do not check Major-/Minor Versioning Avail if you have checked Versioning Avail.

Table 3. Characteristics Required for Major-/Minor- Versioning

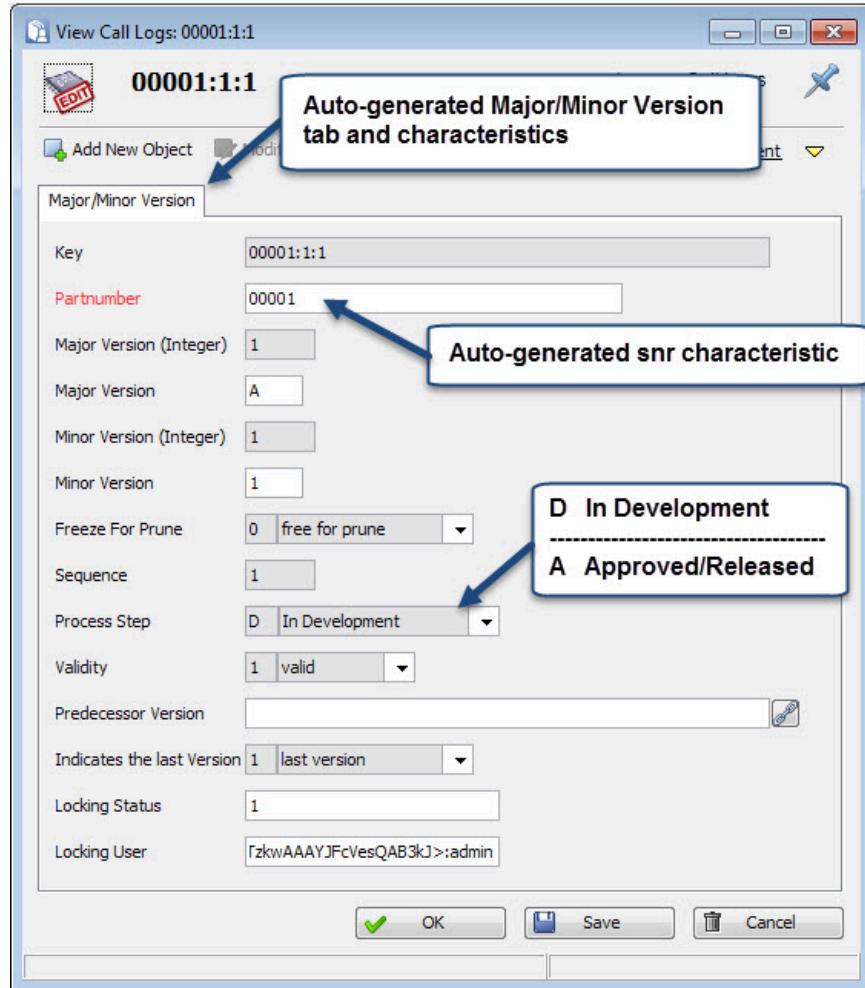
Characteristic Name	Definition
<Class-No.>obj_id	An object identifier that consists of the object's major version (intvers), minor version (iminvers) and the object name (snr).
<Class-No.>snr	The name of the object, labeled Part Number by default.
<Class-No.>vers	Major Version.

Characteristic Name	Definition
<Class-No.>intvers	Major Version Integer (for system-internal use).
<Class-No.>minvers	Minor Version.
<Class-No.>iminvers	Minor Version Integer (for system-internal use).
<Class-No.>freeze	Characteristic used to determine whether or not an object is frozen for prune (0=free for prune, 1=frozen for prune). An object which is frozen for prune will not be deleted when the user performs a prune action.
<Class-No.>seq	A sequence number (that is, a counter for object modifications).
<Class-No.>proc	Status of an object (process step).
<Class-No.>aktobj	Characteristic used to identify valid and invalid versions.
<Class-No.>pre_vers	Characteristic used to determine the previous version of an object.
<Class-No.>lastvers	Characteristic to identify the latest version of an object that has been created.

The tab, label(s), and placement of Major-/Minor- versioning characteristics depend on one of three use cases:

- **Creating a completely new object class with Major/Minor Versioning** — Places characteristics on a **Major/Minor Version** tab, with characteristics having default labels and placement coordinates ([Figure 19](#)). In addition, the software creates the D (In Development) and A (Approved/Released) default process steps.

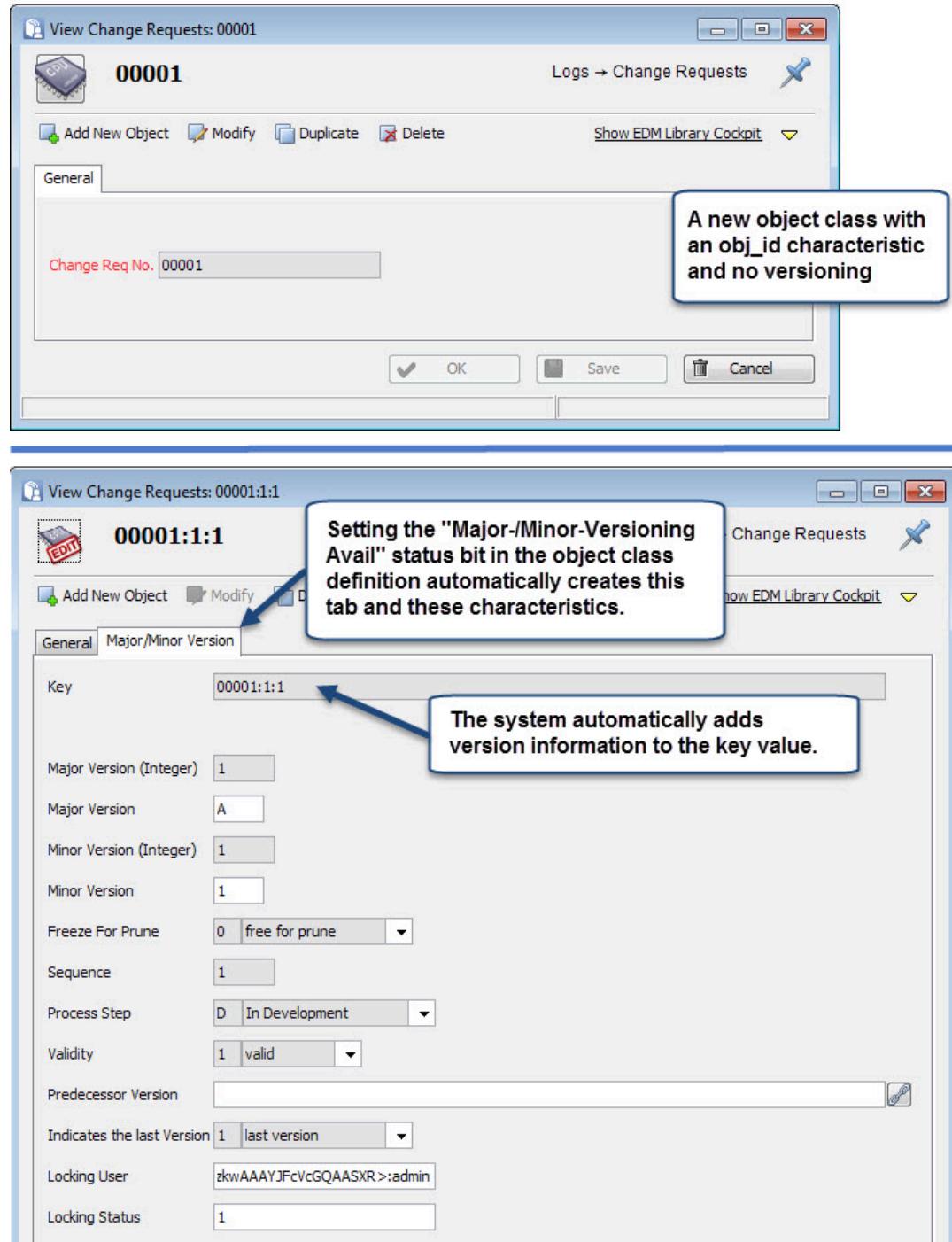
Figure 19. Major/Minor Versioning for a New Object Class



- Initializing Major/Minor versioning on an existing object class without versioning, but that has data — Also creates a **Major/Minor Version** tab with default characteristics and the D (In Development) and A (Approved/Released) default process steps.

In addition, the software moves data from the obj_id main key characteristic of the class to the snr characteristic and generates a new main key characteristic by concatenating the contents of the ^snr^:^intvers^:^iminvers^ fields (Figure 19).

Figure 20. Major-/Minor- Versioning on Existing Object Class (No Versioning)



- Initializing Major/Minor versioning on an existing object class with versioning — The following occurs

- Existing characteristics stay on the original tab. The software places characteristics that did not exist on a **Major/Minor Version** tab.
 - Existing characteristics keep the existing label. New characteristics have default labels.
 - The software does not modify the placement, names, and characteristic labels of characteristics that already exist.
 - Existing process steps stay the same.
 - The software changes data in the obj_id main key characteristic to reflect the Major-/Minor versioning syntax (^snr:^intvers:^iminvers^). For example, EDM Library software changes an obj_id that had a value of "Project_Cell_Phone:0001" to "Project_Cell_Phone:1:1".
-



Note:

Initializing Major-/Minor versioning for an existing class that already has versioning initialized can result in gaps between characteristics on different tabs. To make the characteristic placement more visually appealing, use compose mode.

- The software migrates existing data to the new Major-/Minor data model.

The updated information window is now similar to [Figure 20](#).



Note:

Deactivating the Major/Minor Versioning flag on a class with Major/Minor versioning already enabled results in an error when saving the object class.

- **HTML-Flag** — No longer used and has no effect in the EDM Library Cockpit application.
 - **Versioning Avail** — Enables standard versioning for objects in the class.
-



Note:

Do not check Versioning Avail if you have checked Major/Minor Versioning Avail.

In addition to checking the Versioning Avail status bit, a specific set of characteristics must also be available to activate the versioning functionality ([Table 4](#)).

Table 4. Characteristics Required for Release and Revision Control

Characteristic Name	Definition
<Class-No.>obj_id	Object identifier that consists of the object name (snr) followed by the object version (vers)
<Class-No.>vers	Version of the object
<Class-No.>snr	Name of the object

Table 4. Characteristics Required for Release and Revision Control (continued)

Characteristic Name	Definition
<Class-No.>proc	Process step, first level (process state)
<Class-No.>st	Process step, second level (process status)
<Class-No.>seq	A sequence number (that is, a counter for object modifications)
<Class-No.>adat	Start date to indicate the beginning of the life cycle of the object
<Class-No.>edat	End date for the life cycle of the object. If the edat characteristic contains a value, the version is invalid.
<Class-No.>aktobj	Help characteristic used to identify valid and invalid versions
<Class-No.>ctext	Text characteristic that enables the user to describe a reason for the object modification
<Class-No.>pre_vers	Optional help characteristic used to determine the previous version of an object.

Related Topics

[Release and Revision Control](#)

[Pruning an Object \(Major/Minor Versioning\) \[Xpedition EDM Library Overview\]](#)

[Releasing and Revisioning Objects \[Xpedition EDM Library Overview\]](#)

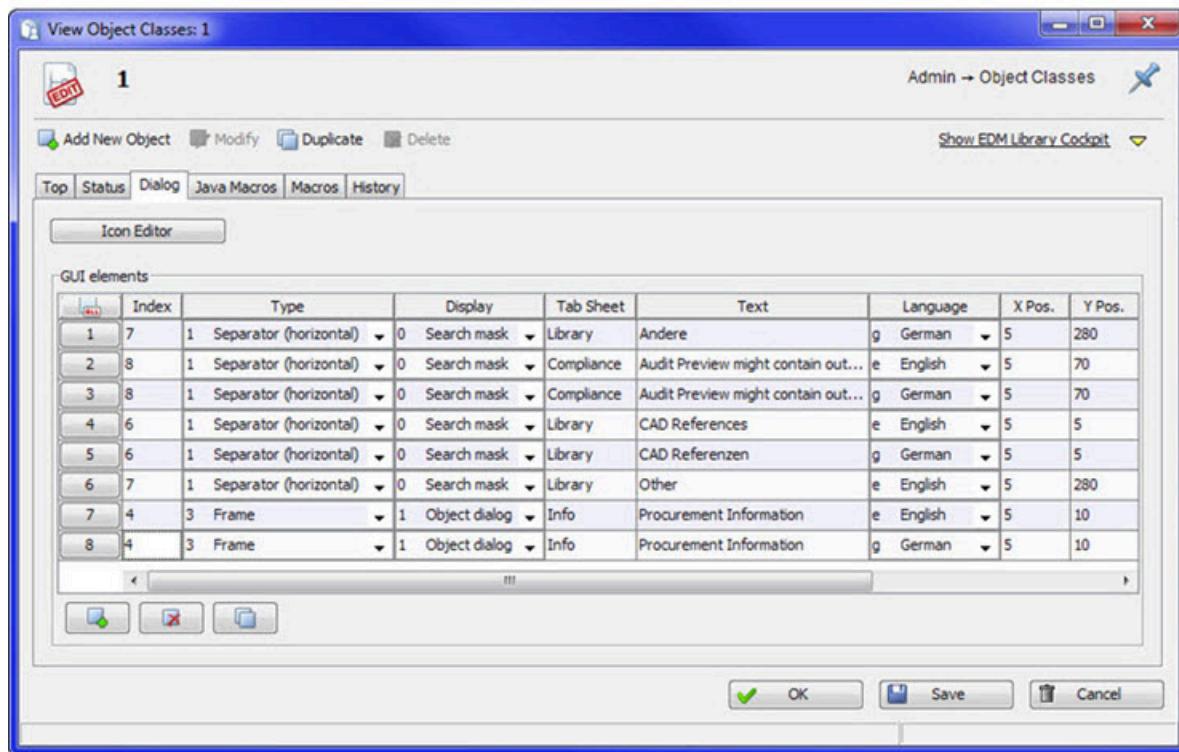
Object Classes Object - Dialog Tab

To access: Refer to “[Opening an Object Class Information Window](#)” on page 46.

The **Dialog** tab defines icons associated with the object class and contains the definition and position settings for any graphical elements (such as text separators or boxes around characteristics) in the search and information windows.

Description

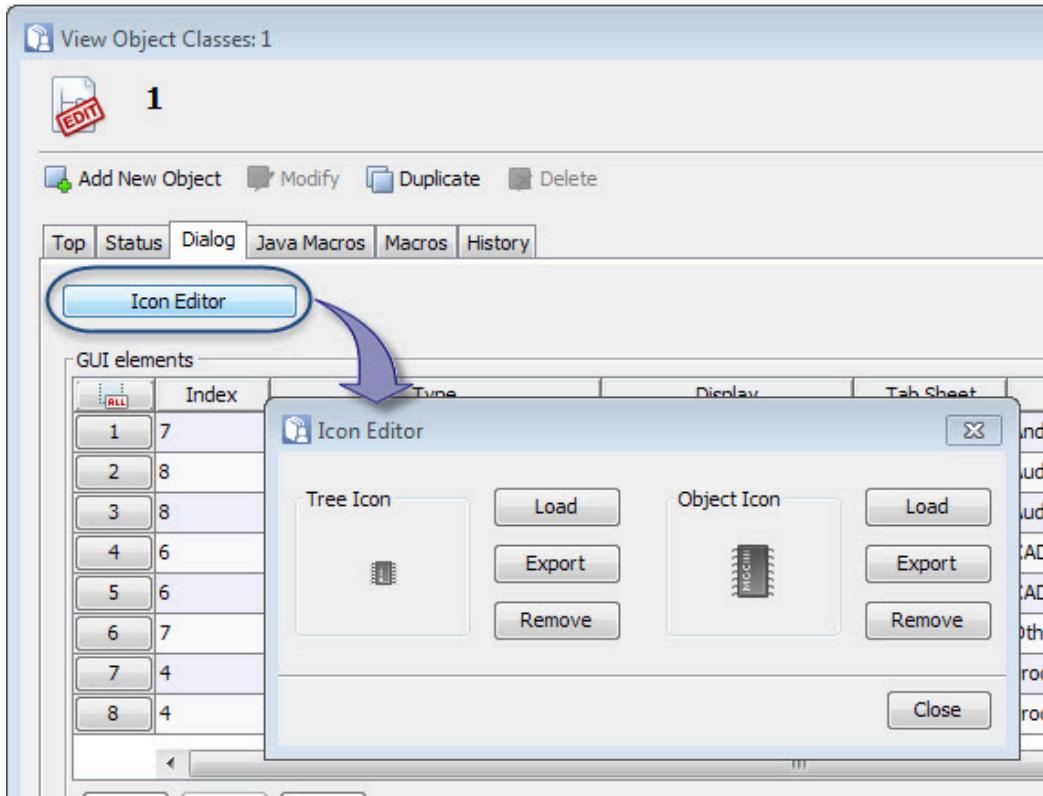
Figure 21. Dialog Tab of an Object Classes Object



Icon Editor

Icons that display in the classification tree and on object information windows reside inside the data model. To associate an icon with the top level of the tree, use the **Icon Editor** button (Figure 22).

Figure 22. Defining the Icons for an Object Class



Clicking the **Icon Editor** button produces a dialog box with the following three buttons to set the tree icon or object information window icon:

- **Load** — Import a bit map image from the file system into the data model and associate it with the object class at the top level of the classification tree.
- **Export** — Export the icon to the file system as a bitmap image for editing.
- **Remove** — Remove the icon from the data model so that the object class no longer has an associated icon in the classification tree.

GUI Elements

The GUI elements list box at the bottom of the **Dialog** tab lets you position text on the search window or information window using graphical elements such as horizontal separators, vertical separators, labels, or frames. These graphical elements help to group or label characteristics in the GUI.

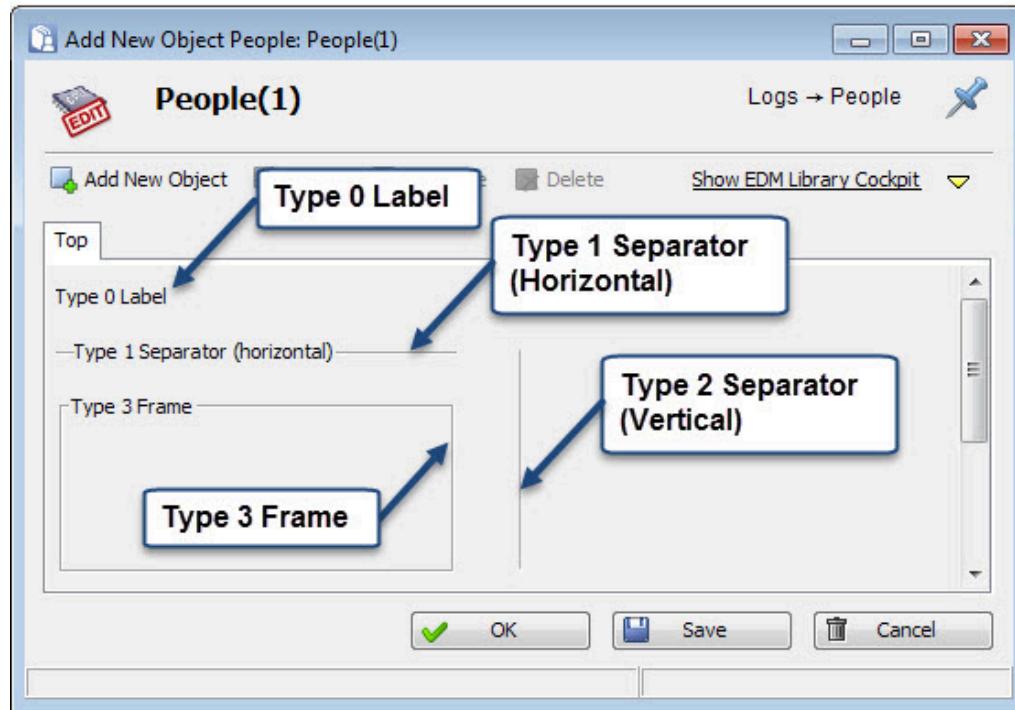
To define graphical elements, specify values in the following columns:

- **Type** — The type of graphical element:
 - 0 (label)** — Text positioned on the search window or information window
 - 1 (horizontal separator)** — Text positioned on the search window or information window with a horizontal bar beneath

2 (vertical separator) — Vertical bar positioned on the search window or information window

3 (frame) — Text positioned on the search window or information window with a box of a specified size

Figure 23. Label, Separator and Frame Examples



- **Display** — Whether to show the GUI element in the search window, in the information window, or both.
 - **Tab Sheet** — The name of the tab of the search window or information window on which to display the GUI element.
 - **Text** — The text to show in the GUI element.
-



Note:

VX.2 and later releases only support English language labels and ignore non-English rows.

- **Language** — The language defined in your data model. The default choices are “e” for English and “g” for German.
- **X Pos and Y Pos** — The X and Y coordinates of the top left corner of the GUI element.
- **Width and Height** — The width of the graphical element (for horizontal separators and frame GUI element types) or the height of the graphical element (for vertical separators and frame GUI types). Because a label is a text string with no other lines associated with it, use “0” as a value for width and height.

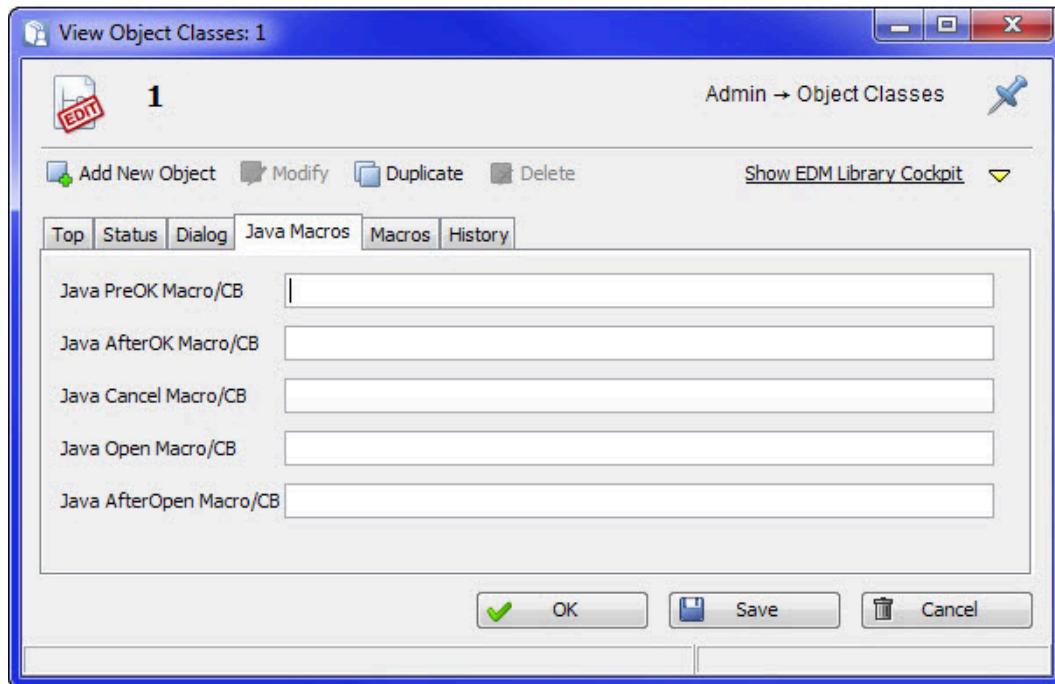
Object Classes Object - Java Macros Tab

To access: Refer to “[Opening an Object Class Information Window](#)” on page 46.

The **Java Macros** tab contains the names of custom Java methods.

Description

Figure 24. Java Macros Tab of an Object Class



Calls listed in the **Java Macros** tab always run synchronously to the Xpedition EDM Library Cockpit application, meaning that the software blocks the user interface until the client method has finished. You can control this behavior through the Java client program so that the Java method immediately returns to EDM Library Cockpit and executes its operation in its own thread, thereby enabling EDM Library Cockpit to run in parallel to the Java method.

The Java macros defined in this tab override the corresponding macros defined in the **Macros** tab. For example, assume the following:

- The **Java Macros** tab defines a Java PreOK Macro/CB.
- The **Macros** tab defines a PreOK Macro/CB and an AfterOK Macro/CB.

Based on these assumptions, the system executes the PreOK Macro/CB from the **Java Macros** tab and the AfterOK Macro/CB from the **Macros** tab.

Characteristics

The characteristics on the **Java Macros** tab enable a developer to implement check and transfer functions associated with a particular user action:

- **Java PreOK Macro/CB** — When a user clicks the **OK** button, the system executes the specified Java method, and then saves (commits) information to the database.
- **Java AfterOK Macro/CB** — When a user clicks the **OK** button, the system saves (commits) information to the database, and then executes the specified Java method.
- **Java Cancel Macro/CB** — The system executes the Java method in this field when a user clicks the **Cancel** button.
- **Java Open Macro/CB** — When a user opens an object for viewing or editing, the system executes the specified Java method before displaying the object information window.
- **Java AfterOpen Macro/CB** — When a user opens an object for viewing or editing, the system opens the object information window, and then executes the specified Java method.

Object Classes Object - Macros Tab

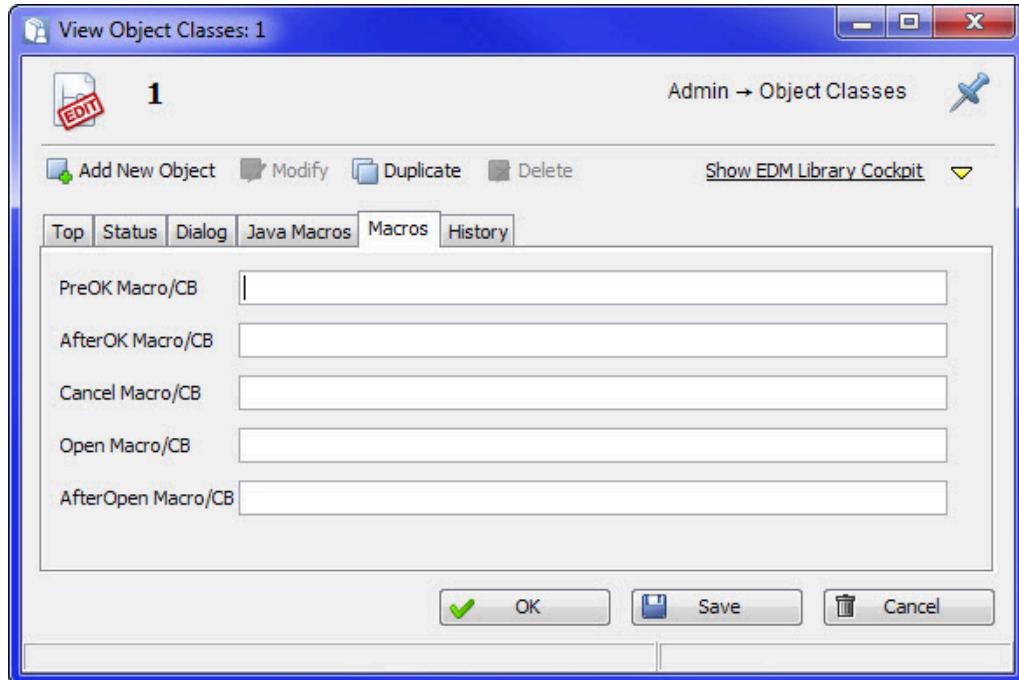
To access: Refer to “[Opening an Object Class Information Window](#)” on page 46.

The **Macros** tab contains the names of user-defined macros or callbacks.

Description

Java methods on the **Java Macros** tab override macros in the **Macros** tab.

Figure 25. Macros Tab of an Object Class



Characteristics

The characteristics on the **Macros** tab enable implementation of custom client functions using C++, TCL/TK, or SQL shell scripts. The value for each of these characteristics is a pathname that includes the macro or callback name as a leaf.

Commands can run synchronously (the EDM Library application is blocked until the command call has finished) or asynchronously by specifying an ampersand sign (&) at the end of the command line. An ampersand causes operation to continue in parallel to the command call.



Note:

EDM Library software automatically executes any PreOK and AfterOK macros when using the EDM Library Cockpit application to graphically “drag and drop” objects from one catalog group into another.

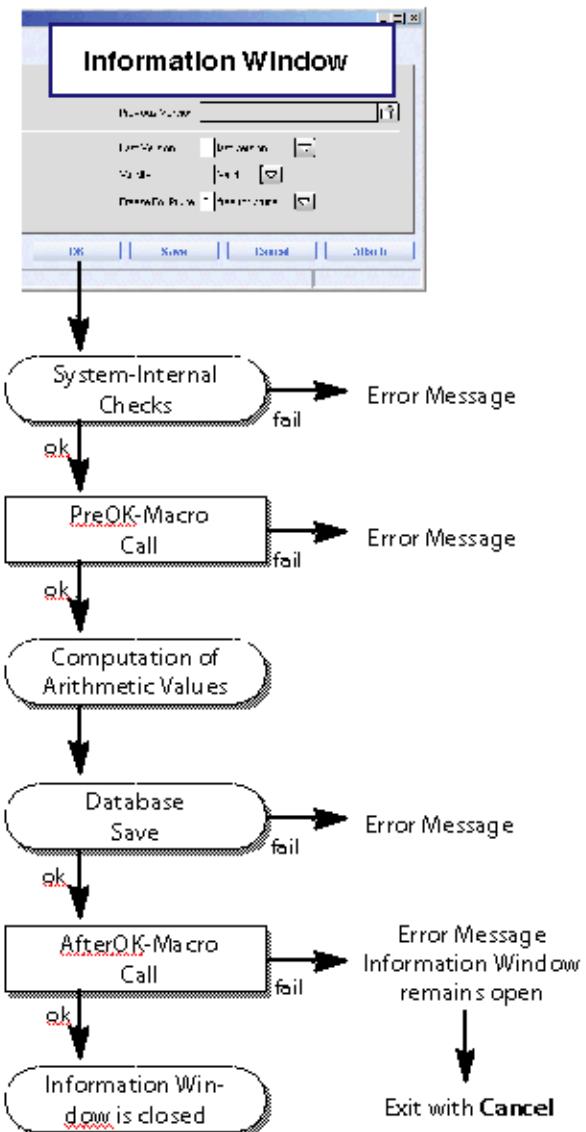
- **PreOK Macro/CB** — Executes a PreOK macro or callback after a user clicks the **OK** button but before actually committing the information to the database.
- **AfterOK Macro/CB** — Executes an AfterOK macro or callback after a user clicks the **OK** button and after committing the information to the database.
- **Cancel Macro/CB** — Executes a Cancel macro or callback after the user clicks the **Cancel** button.
- **Open Macro/CB** — Executes an Open macro or callback when a user opens an object for viewing or editing, but before displaying the object.
- **AfterOpen Macro/CB** — Executes an AfterOpen macro or callback immediately after opening an object for viewing or editing.

Using the characteristics on the **Macro** tab enable a developer to implement check and transfer functions associated with a particular user action. Further, EDM Library can pass the contents of all characteristics inside the class as well as the database name and the mode (Add, View, Modify, Copy, Delete, and Print) to a macro by specifying the name in triangular quotes (for example, ^mode^, ^obj_id^, or ^database^).

Example

This example describes the function of the macros when used in a process ([Figure 26](#)).

Figure 26. Macros - Process



Process:

After opening the information window of an object class with an OKMacro and AfterOK defined and clicking **OK** to save the object, EDM Library starts internal checks (for example, a mandatory field check when Status is set to A).

If the internal checks are not successful, an error message appears.

If the internal checks are successful, EDM Library calls the **OK** macro. The macro returns either a 1 (fail) or 0 (OK).

If the **OK** macro succeeds, EDM Library computes the arithmetic values for the fields with a formula.

Commits the object information to the database. If the commit fails, an error message appears.

If the save is successful, EDM Library calls the AfterOK macro.

If the AfterOK macro succeeds, EDM Library closes the object's information window. If an error occurs, a message appears and the information window remains open. To close the information window, click the **Cancel** button.

The following shows the OK macro content:

```

#!/bin/sh
# NAME      : example_macro.sh
# VERSION   : 1.1
# CREATED ON : August 18, 2009
# MODIFIED  : August 25, 2009
# MODIFIED  :
# COMMENT   :

***** DEBUGGING *****
  
```

```

# set -x
# echo "$@"

***** ARGUMENTS *****
# The sequence of the arguments can be the same as from the call in EDM
# Library and the macro may not be executed as a background process!
#
# For example: example_macro.sh ^mode^ ^obj_id^ ^database^

# ARGUMENT_COUNT = <number of arguments>

ARGUMENT_COUNT=3

MODE=$1
OBJ_ID=$2
DBNAME=$3

#... and more...

***** VARIABLES *****
# DBNAME is the name and the password of the database
# DBNAME=<db_user>/<db_passwd>@<servicename>

# Definition of the different modi (is fix!)

ADD=1
VIEW=2      # VIEW mode has no OK button
MODIFY=3
COPY=4
DELETE=5
MOVE=6
PRINT=7     # PRINT mode has no OK button

***** FUNCTIONS *****
----- USER-Message -----
message()
{
    xmESSAGE -center -timeout 10 $NEWS
    echo "\n***** E R R O R *****\n"
    echo $NEWS
}
----- DUMMY -----
dummy_function()
{
echo "$@"
}

***** MAIN ***** MAIN ***** MAIN ***** MAIN *****

```

```
# Change the following lines for your own application.

#----- Check the argument count -----

if [ $# -ne $ARGUMENT_COUNT ];then
NEWS="wrong argument count!"
message
exit 1
fi

#----- Check for required arguments -----

if [ -z "$1" -oz "$2" ];then
NEWS="missing obligatory arguments!"
message
exit 1
fi

#----- Check for correct MODE -----

if [ $MODE -ne $ADD -a $MODE -ne $MODIFY ];then
exit 0
fi

#----- Your own macro code -----
# valid return values:  0 = OK      and      >0 = FAIL
dummy_function $*
exit 0
```

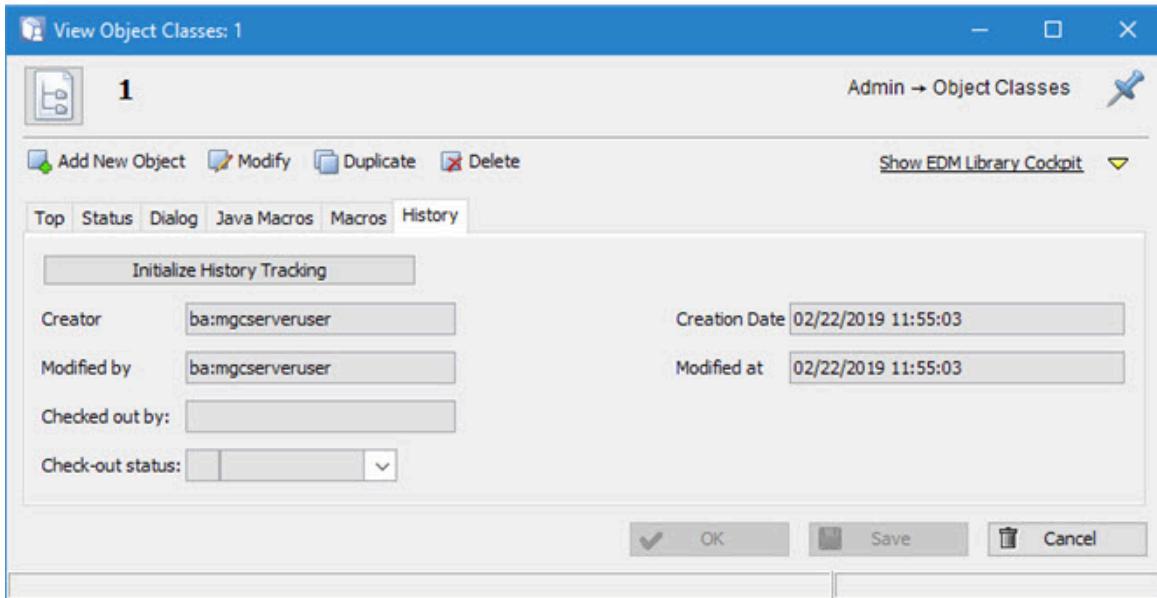
Object Classes Object - History Tab

To access: Refer to “[Opening an Object Class Information Window](#)” on page 46.

The **History** tab contains system-maintained information about who modified the object, the date of modification, and an action button to initialize history tracking.

Description

Figure 27. History Tab of an Object Class



Characteristics

The **History** tab contains the following system-maintained characteristics:

- The **Initialize History Tracking** button initializes history tracking for all characteristics that have the History Tracking bit checked and that are associated with the class.
- **Creator** — The login name of the user who created the object.
- **Modified by** — The login name of the user who last modified the object. The system automatically updates this field when a user edits an object.
- **Creation Date** — The date of creation.
- **Modified at** — The last date that the object was modified.
- **Checked out by** — The username of the person that has the object checked out. Checking out an object prevents others from editing that object.
- **Check-out status** — When set to "1" indicates the object is checked out and cannot be edited by a user other than the one identified by the "Checked out by" characteristic value. When an object is checked in and has a value of "0", other users can edit that object.

Object Class Creation

Creating a new object class enables users to store objects that hold a specific category of data, usually unique to your company processes, not covered by the default object classes. Before creating a new object class, you must find an open object class number and catalog key to use for the new class.

[Finding an Open Object Class Number and Catalog Key](#)

[Creating an Object Class](#)

Finding an Open Object Class Number and Catalog Key

Before creating an object class, you must determine a unique identification number and catalog key to use for the new object class that is not already used by default or custom object classes in the data model.

Prerequisites

- You have invoked Xpedition EDM Library Cockpit and connected to an EDM Server.
- Your user account has administrator privileges.

Procedure

1. Choose **Admin > Object Classes** in the classification area of EDM Library Cockpit.

A search criteria pane for the object classes class displays. The search criteria pane can be in either quick search mode or advanced search mode, depending on your preference settings. A link at the top of the search criteria pane lets you switch between modes.

2. Display the advanced search criteria pane and check the following minimum set of characteristics to return as column heads in the search results list:

- Class Number
- Class Name
- Catalog Key
- Table Name

3. Click the **Search** button to return a list of all Object Classes objects in the database. ([Figure 28](#)).

You may see duplicate entries in the search results list if your database is configured for multi-language support.

Figure 28. List of Object Classes in the Search Result Pane

ALL	Status	Class Number	Table Name	Catalog Key	Class Name
1	S	0	te_dyn		Dyn. Charact.
2	A	1	te_sachnr	AA	Components
3	A	3	te_pack	CC	Package
4	S	8	te_glb_rastw		Dyn. Enum. Values
5	A	9	te_shop	HH	Shopping List
6	A	10	te_baust		Mapping
7	A	15	te_lib		Library Specification
8	A	18	te_geomtech		Technology
9	U	19			Flows
10	A	20			Editor BOM
11	A	21	te_stckliste		Variant BOM
12	S	22	te_skn		Catalog Groups
13	A	33	te_grafikdef		Editor Prefs.
14	S	35	te_mgevents		MailGate Events

4. If not already in numerical order, click the Class Number column heading to sort the object class list by class number.
5. Find the largest object class number and choose a larger number for the new object class.



Note:

Numbers less than 900 are for system and EDM Library application classes. 901 through 904 are also reserved. Developers should choose a number between 905 to 999 for any new custom classes.

6. Sort the displayed classes again by Catalog Key to show the catalog keys used by other object classes. Choose a key for the new class that is not already used (for example, ct, WW, or XA).
7. Sort the displayed classes again by Table Name to show the names of tables in the database where object classes store characteristic data. Make sure to choose a table name for the new class that is not already used.

Results

You now know what object classes already exist, their class numbers, their catalog keys, and their tables within the data model. When performing the [Creating an Object Class](#) task, make sure to choose class numbers, catalog keys, and table names that do not conflict with another object class.

Creating an Object Class

After you have a unique identification number and class key, you can create a new object class.

Prerequisites

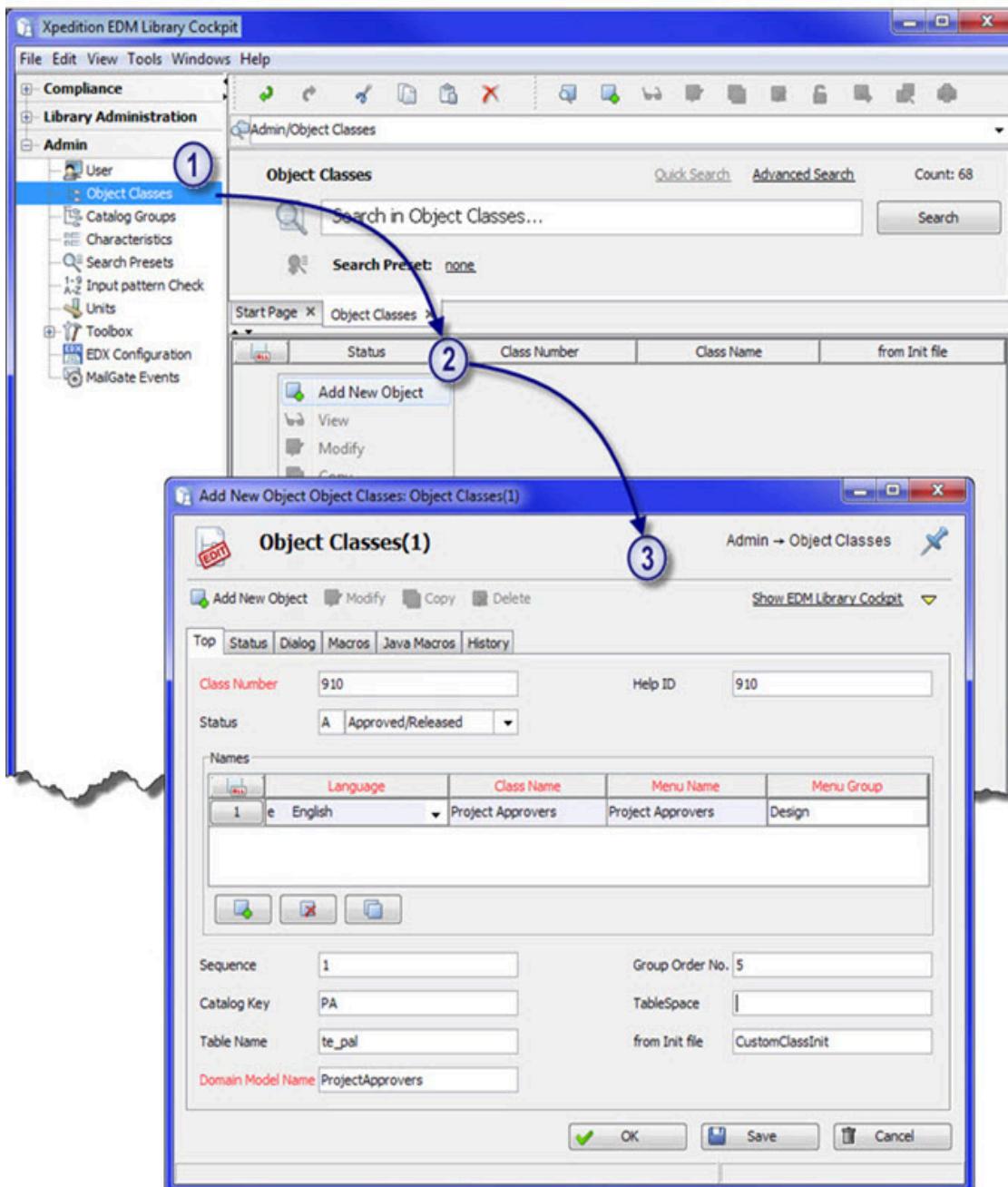
- You completed the “[Finding an Open Object Class Number and Catalog Key](#)” task.
- Your user account has administrator privileges.
- You acquired a Developer license that permits creating new object classes in a database.

Procedure

1. Choose **Admin > Object Classes** in the classification area of the Xpedition EDM Library Cockpit application ([Figure 29](#)).
2. Add a new object classes object or copy an existing object classes object:
 - To display an empty information window, right-click in the search results area and then choose **Add New Object**.
 - To copy and adapt the properties of an existing object class, perform a search and select a row in the search results list. Then, right-click and choose **Duplicate**.

It is usually easier to copy an object classes object using **Duplicate** because you then only need to change those entries in the information window that require new values.

Figure 29. Creating a New Object Class



3. Type or verify data on the **Top** tab ([Table 5](#)).

Table 5. Top Tab Characteristic Data Values

Characteristic	Data Value
Class Number	Type a unique three digit integer between 905 and 999 (for example, 910). The class number must not be a duplicate of any existing class number and must not be in the range 0-904.

Characteristic	Data Value
Help ID	Leave this field empty.
Names list	<p>Create a new row and then choose a Language value of “English”, and type values for Class Name, Menu Name, and Menu Group. Optionally, you might want to have custom classes reside in a menu group different from any of the factory default menu groups. For example, you create a new class you name “Rad Tracking Reports” in a new menu group you name “Custom”.</p> <p> Note: In VX.2 and later, EDM Library only supports English (e). Refer to Figure 14 on page 50 for an example of how the Class Name and Menu Group values display in the EDM Library Cockpit classification pane.</p>
Sequence	Type an integer number to define the vertical position of the Menu Name within the Menu Group (refer to Figure 15 on page 51).
Group Order Number	Type an integer number to define the vertical position of the Menu Group category in relation to other Menu Group categories (refer to Figure 16 on page 51). If placing your new object class into an existing Menu Group, use the same integer value as used other object classes in that Menu Group.
Catalog Key	Type a unique value that is not used as a catalog base key by other object classes. For custom object classes, start the catalog key with “X” or “x” followed by any other letter (for example, the value “XA”).
TableSpace	Leave this field empty unless you want to manage this class in a particular Oracle tablespace.
Table Name	<p>Type a unique name for the class table.</p> <p> CAUTION: Make sure that the table name does not already exist and that you do not use a reserved word in SQL (refer to your database documentation).</p>
from Init File	<p>As a developer of a custom object class, always enter a value for from Init file to enable the batchadmin tool to automatically write the object class definition to an initialization file.</p> <p>When initializing a database by loading the contents of several initialization files, an administrator can then subsequently determine which initialization file created the object class in the data model.</p>
Domain Model Name	Type a name that uniquely identifies the class to the software.

4. Check the following bits on the **Status** tab as required by the new class:

- Catalog Groups to implement a tree structure
- Search Mask Avail to enable the display of the search window

- Modify, Delete, Copy, Move and Add Allowed to enable the use of all object editing functions.
- TE_OBJ Charact. to automatically create object characteristics for this class in the te_obj table
- Generate Number to automatically generate the Object IDs for newly created objects.

5. Click **OK** to save the new object class.



Note:

Users cannot yet add objects to the object class.

6. Choose **File > Refresh Data Model**.

Results

The new object class is now in the classification hierarchy.

Before a user can create new objects in the object class, you must create a characteristic named obj_id inside the table of the object class (refer to “[Management Characteristics](#)” on page 105).

Related Topics

[Finding an Open Object Class Number and Catalog Key](#)

[Object Classes Object - Top Tab](#)

[Object Classes Object - Status Tab](#)

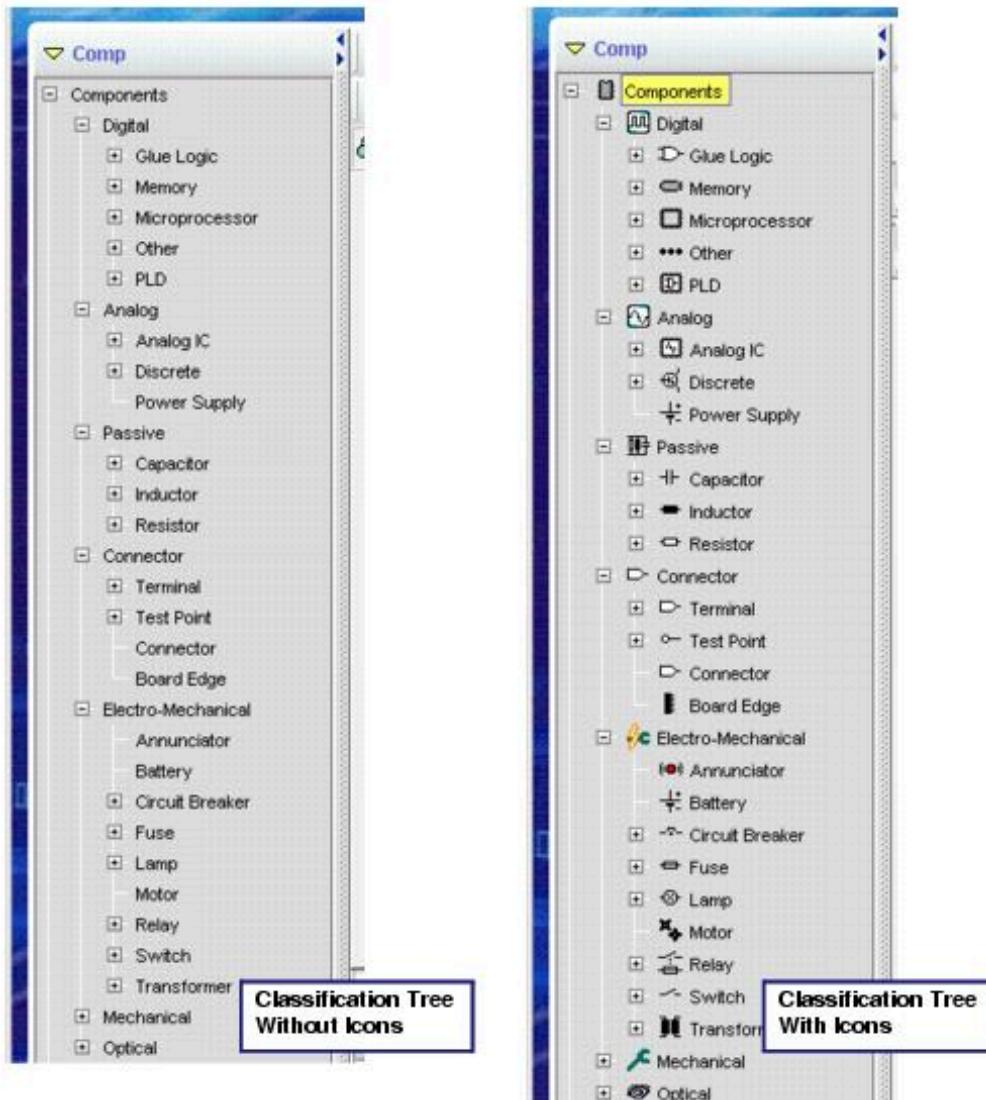
[EDM Library Data Model Naming Conventions](#)

Catalog Group Administration

Catalog groups create a hierarchy that classifies objects within an object class. The configuration of catalog groups within an object class is called a classification tree. You can configure a classification tree to use identifying icons, or not use identifying icons.

Figure 30 shows an example classification scheme (not all branches of the classification tree are shown) for Component objects both with and without icons.

Figure 30. Example Components Object Class Tree Structure



Each line item in the hierarchy shown in Figure 30 represents a Catalog Group object that resides in the catalogs group object class (for example, Digital, Power Supply, Passive, Resistor, and so on are all Catalog Group objects). Users that are not administrators can sometimes add new subcatalog groups to certain class hierarchies (for example, component engineers who need to create categories of components or librarians who need to create catalog groups for Request objects). Some programs also create catalog objects in the database automatically to store imported data. For example, when importing

central library data into a database for the first time, the loader program creates a catalog hierarchy that mirrors the partition structure of the library on the file system.

[Opening a Catalog Group Information Window](#)

[Catalog Group Key](#)

[Catalog Group Object - Top Tab](#)

[Catalog Group Object - Generator Tab](#)

[Catalog Group Object - Admin Tab](#)

[Catalog Group Object - GUI Tab](#)

[Catalog Group Object - Rights Tab](#)

[Catalog Group Object - History Tab](#)

Opening a Catalog Group Information Window

You can open an information window for a Catalog Group object by using the popup menus in the classification tree pane or the popup menus in the search results list.

[Opening a Catalog Group Information Window From the Classification Pane](#)

[Opening a Catalog Group Information Window From the Search Results List](#)

Opening a Catalog Group Information Window From the Classification Pane

You can use the classification tree popup menu items to open a Catalog Group information window for the purpose of creating a new catalog group, or to view or edit the definition of an existing catalog group.

Prerequisites

- You invoked Xpedition EDM Library Cockpit and connected to an EDM Server.
- To add new catalog groups, you must use an account with EDIT permission for the parent catalog group and the Top catalog of the catalog groups object class (refer to [“Access Permission Levels”](#) on page 27).

Procedure

1. Select a catalog in the classification hierarchy pane.
2. With the cursor still in the classification hierarchy pane, right-click and then choose **Add Catalog**, **View Catalog**, or **Modify Catalog**.

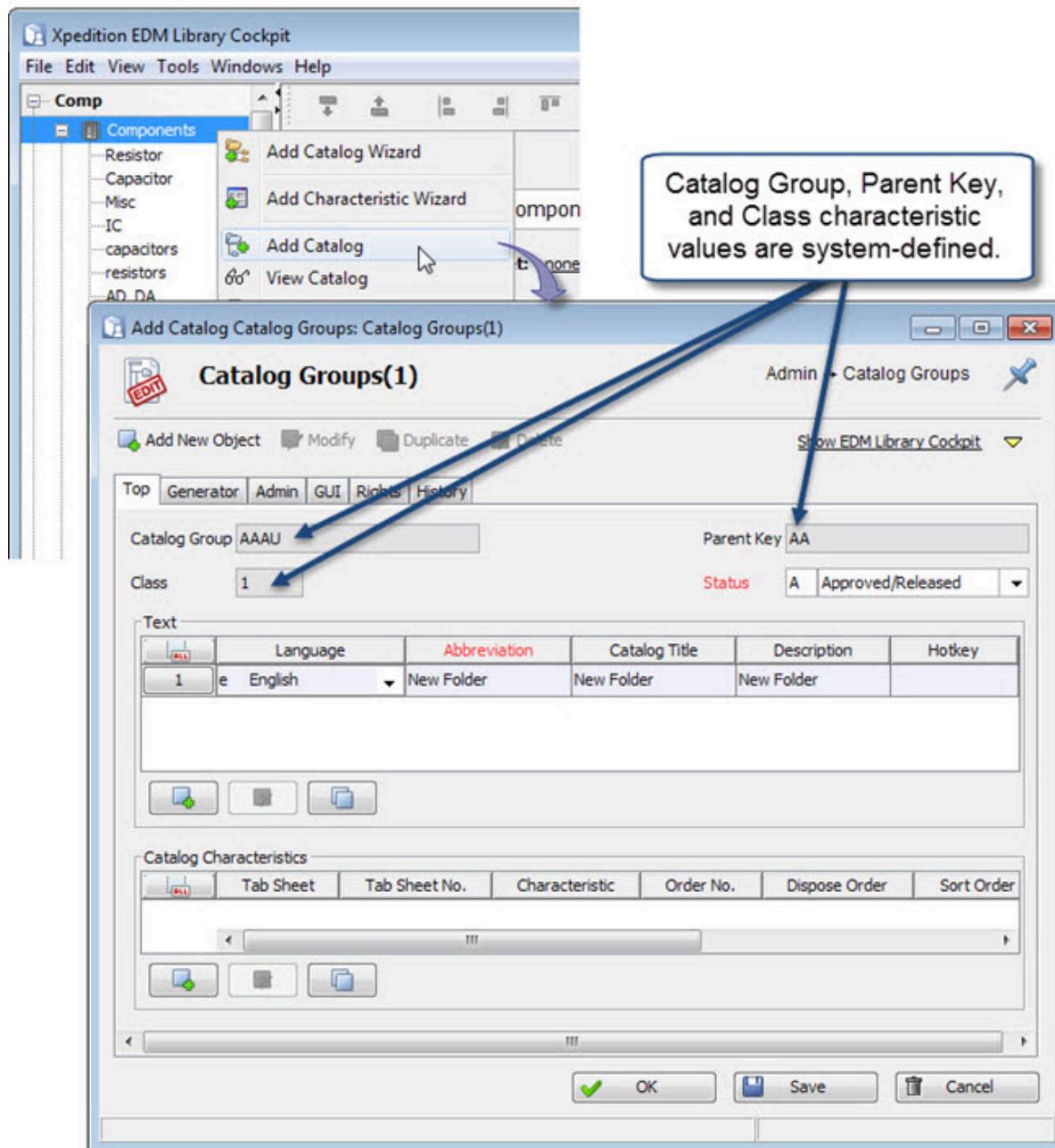
Each menu item displays a Catalog Group information window. “Classification Tasks” in the *EDM Library Overview* contains more information about tasks you can perform from the classification hierarchy area.

In most Comp module object classes, you can also use the Create/Modify Catalog wizard described in “Adding or Modifying a Catalog” in the *Xpedition EDM Library Guide for Component Engineers*.

Results

The Catalog Groups information window displays (Figure 31).

Figure 31. Adding a New Catalog Group



Related Topics

[Opening a Catalog Group Information Window From the Search Results List](#)

[Catalog Group Key](#)

[Catalog Group Object - Top Tab](#)

[Catalog Group Object - Generator Tab](#)

[Catalog Group Object - Admin Tab](#)

[Catalog Group Object - GUI Tab](#)

[Catalog Group Object - Rights Tab](#)

[Catalog Group Object - History Tab](#)

Opening a Catalog Group Information Window From the Search Results List

You can use the search results list popup menu items to open a Catalog Group information window for the purpose of creating a new catalog group, or to view or edit the definition of an existing catalog group.

Restrictions and Limitations

- Because there is no reference point with which to formulate an automatic catalog group key, you cannot add a Catalog Group object using the catalog group search window. You must use the classification tree popup menu to add a new catalog group.

Prerequisites

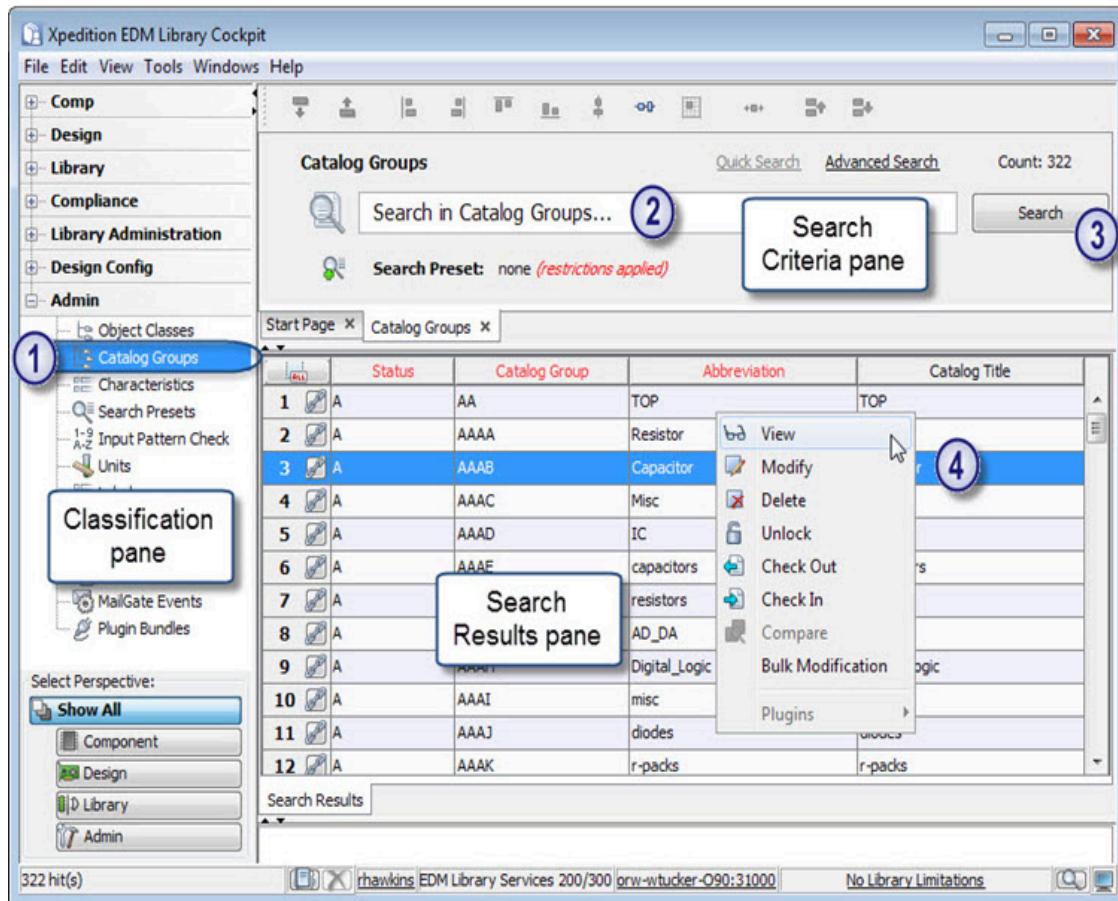
- You invoked Xpedition EDM Library Cockpit and connected to an EDM Server.

Procedure

- In the object classification pane, open the Admin module and choose **Catalog Groups** ([Figure 32](#)). Alternately, type Admin/Catalog Groups in the location bar.

The search criteria pane displays in either quick search mode or advanced search mode, depending on your preference settings. A link at the top of the search criteria pane lets you switch between modes.

Figure 32. Catalog Group Search Window



Tip

For general information about searching in EDM Library Cockpit and how to formulate a search query, refer to “Searching and Replacing” in the *EDM Library Overview*

2. Type a search query into the search criteria pane. Leave all search criteria fields empty to return a list of all Catalog Group objects.

Advanced search mode enables you to enter search criteria into one or more input fields on multiple tabs and place a check mark next to the characteristics that you want as columns in the search results list.

3. Click **Search**.

4. Select a row from the search results list. Then, with the cursor still in the search results area, right-click and choose **View** or **Modify**.

Alternately, click the reference button to the left of a row to open an information window in view mode without having to first select the row or use the popup menu.

An information window can either be floating or docked. Your EDM Library Cockpit preferences determine the default.

Related Topics

[Opening a Catalog Group Information Window From the Classification Pane](#)

[Catalog Group Key](#)

[Catalog Group Object - Top Tab](#)

[Catalog Group Object - Generator Tab](#)

[Catalog Group Object - Admin Tab](#)

[Catalog Group Object - GUI Tab](#)

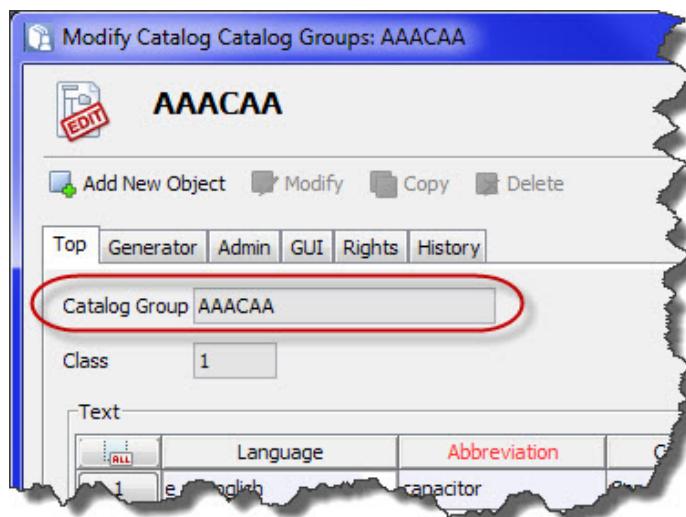
[Catalog Group Object - Rights Tab](#)

[Catalog Group Object - History Tab](#)

Catalog Group Key

The catalog group key is a letter code in the Catalog Group information window that uniquely defines all catalog groups stored in the database.

Figure 33. Catalog Group Key



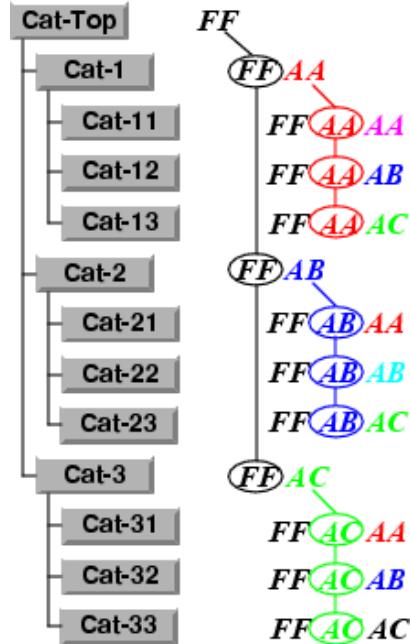
The key consists of a user-defined base key and a hierarchy key that the system automatically generates, as follows:

- **Base Key** — The base key matches the catalog group key of the object class or Top catalog group. The base key has two digits and is defined in the Object Classes object class. Any value chosen for the base key must be unique. The convention is to use two letters (for example, AA, bb, ... ZZ). You cannot change the base key in the Catalog Groups object class.
- **Hierarchy Key** — When you create a catalog group, the system automatically appends a hierarchy key to the base key to uniquely define the catalog group, as shown in [Figure 34](#). For each further level in the hierarchy, the system continues to automatically append two letters. The

hierarchy key of the first group in a hierarchy level begins with AA, the second with AB, and so on.

The catalog group key of the top level consists only of the base key (two letters; FF in the example in [Figure 34](#)). The system appends two letters of the hierarchy key to this at the second level of the hierarchy for a total of four letters. A total of six letters occur at the third level of the hierarchy. The figure below illustrates the key codes:

Figure 34. Example Allocation of Catalog Group Keys



When adding a new catalog group, the software automatically fills in the next available unique key value, thereby denoting the location of the catalog in the classification hierarchy. For example, in [Figure 31](#) on page 81 the key of the selected parent catalog group (named Components) is "AA". The Components catalog group already has eight subcatalog groups whose unique keys are AAAA through AAAH. The software automatically assigns the next available unique key value of AAAI to the new catalog group to note that it is the ninth subcatalog group below the parent.

Related Topics

[Opening a Catalog Group Information Window](#)

[Catalog Group Object - Top Tab](#)

[Catalog Group Object - Generator Tab](#)

[Catalog Group Object - Admin Tab](#)

[Catalog Group Object - GUI Tab](#)

[Catalog Group Object - Rights Tab](#)

[Catalog Group Object - History Tab](#)

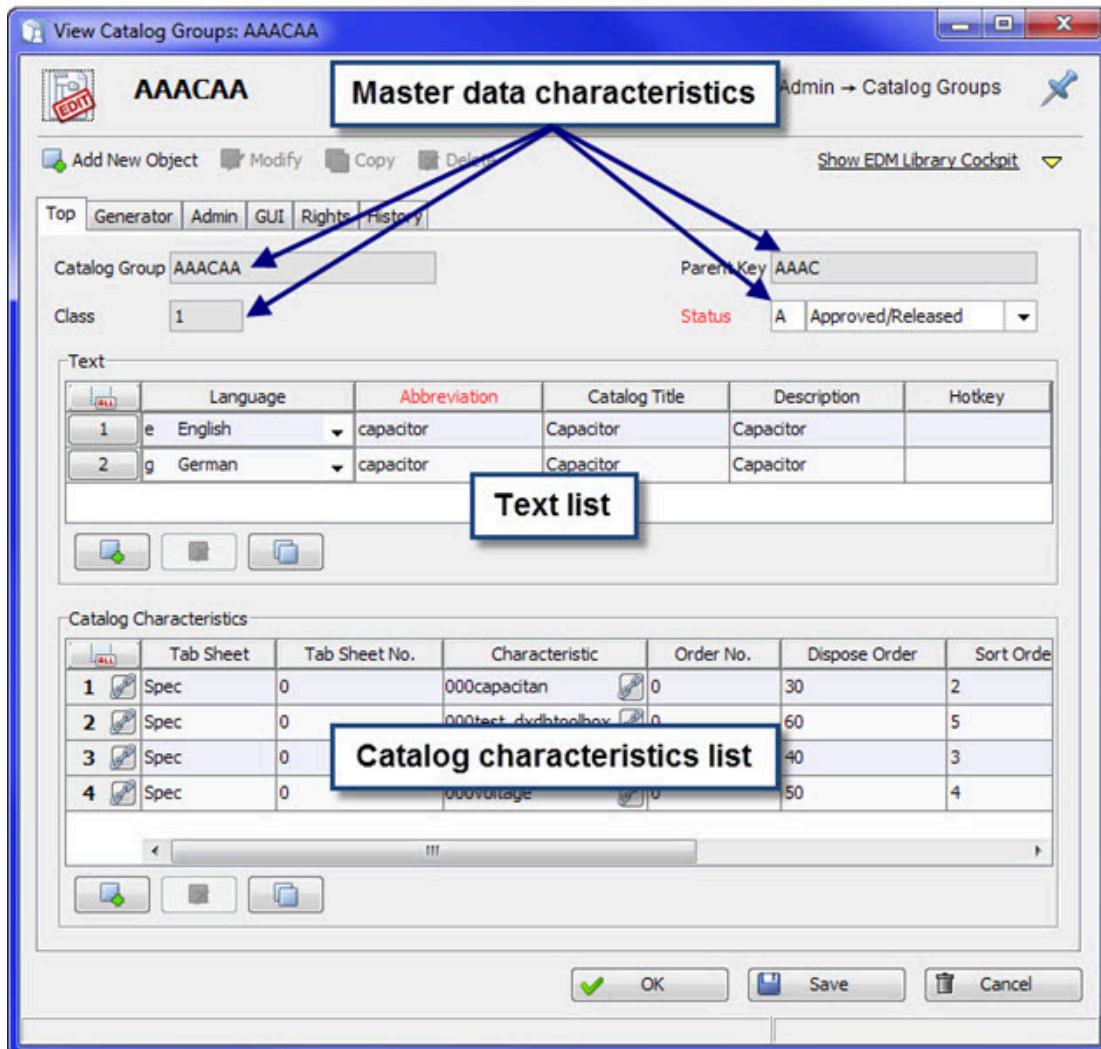
Catalog Group Object - Top Tab

To access: Refer to “[Opening a Catalog Group Information Window](#)” on page 80.

The **Top** tab of a Catalog Group object contains the master data characteristics, a text list, and a catalog characteristics list.

Description

Figure 35. Top Tab of a Catalog Group



Characteristics

Characteristics on the **Top** tab can be divided into:

- Master data characteristics
- Text list
- Catalog characteristics list

Master Data Characteristics

The system generates the values for the Catalog Group, Parent Key, and Class master data characteristics when you add a new catalog group to a tree hierarchy.

- **Catalog Group** — A letter code that represents the unique key of each group.
- **Parent Key** — The letter key of the parent catalog group.
- **Class** — The integer number of the object class containing the catalog group.
- **Status** — The status of the Catalog Group object. The only valid status for a Catalog Group object is A (approved/released).

Text List



Note:

VX.2 and later releases only support English language labels and ignore non-English rows.

The Text list has following characteristic columns:

- **Language** — The language in which the names in the other columns apply. When creating a new catalog group, the default setting (and only recognized value) is "e" for English.
- **Abbreviation** — The abbreviation or number of the catalog group (for example, "CO" as an abbreviation for "Components"). EDM Library Cockpit sorts the catalog groups by Abbreviation when the Tree Sort status bit is set on the **Status** tab of the Object Class object.
- **Catalog Title** — The name of the catalog group as it displays in the classification tree.
- **Description** — Text describing the catalog group.
- **Hotkey** — The software no longer uses values in this column.

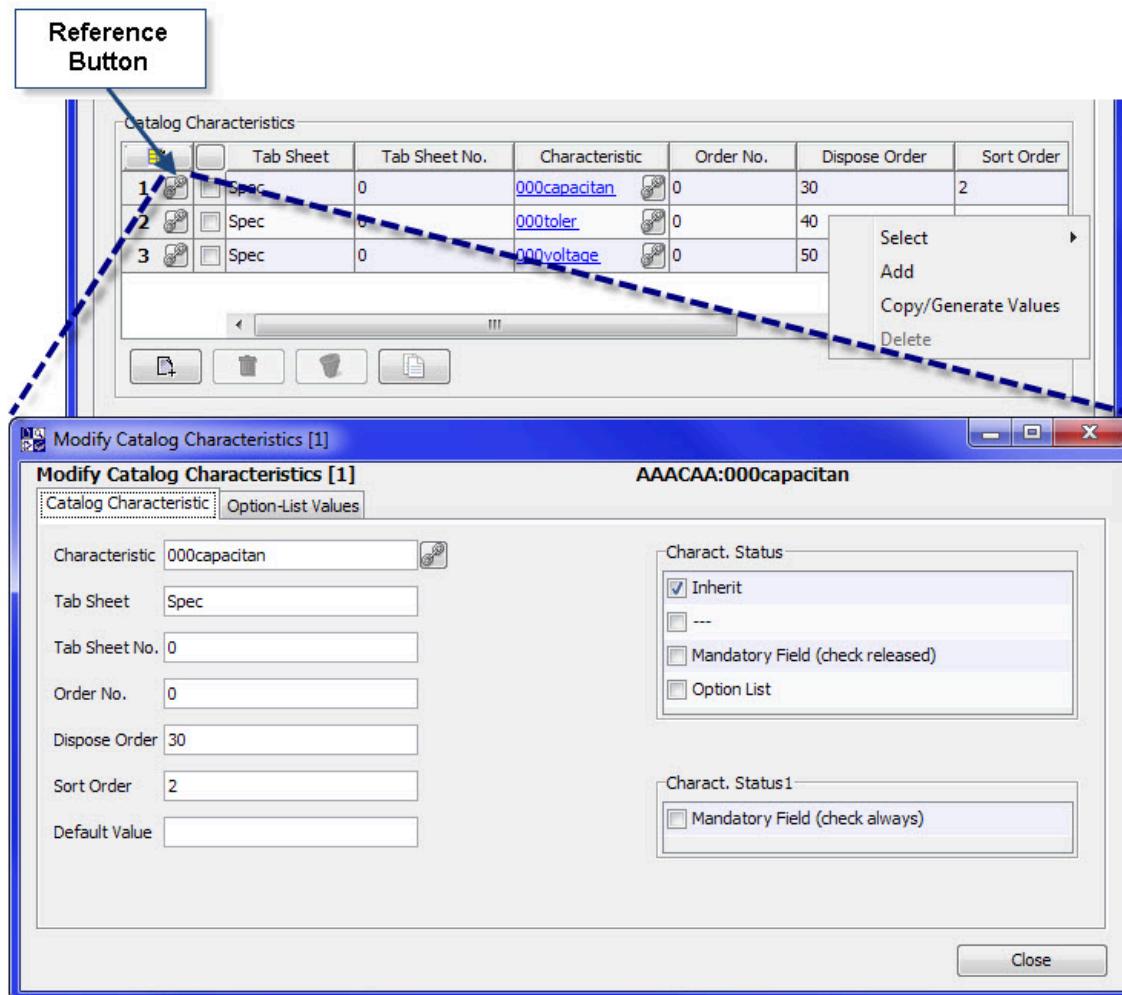
Catalog Characteristics List

The Catalog Characteristics list contains a listing of the dynamic characteristics specific to the catalog group.

Use these buttons to edit the list:

- **Reference button** () - Display the Modify Catalog Characteristics window (Figure 36). Use the window to edit column values in an existing row.
- **Add (popup menu item or button)** — Displays the same window as the reference button, but without values. Complete the form to a new characteristic to the list.
- **Delete (popup menu item or button)** — Delete a row.
- **Copy/Generate Values (popup menu item or button)** — Display a form you can use to automatically append prefixes or suffixes to values within a column, or automatically generate a series of column values with a specified increment. At least one row must first exist in the Catalog Characteristics list.

Figure 36. Catalog Characteristics List



The Catalog Characteristics list contains the following columns:

- **Characteristic** — Contains the alphanumeric ID value of the characteristic to include on the tab. The value of Characteristic must match the value of an existing characteristic in the Characteristics object class.

Use the reference button () next to the Characteristics field to display a search window for the characteristics object class you can use to search for a characteristic ID value. Upon locating the characteristic ID in the search window, double-click to include it in the Catalog Characteristic list. When viewing the definition of a catalog group, the reference button lets you easily see more information about the characteristic whose ID value is already present.

- **Tab Sheet** — Specifies the name of the tab within the catalog group in which the characteristic displays. Once the catalog group has been defined, the name of the tab also displays in the search and information window. The value is case-sensitive.
- **Tab Sheet No.** — An integer that identifies the characteristic group and determines the arrangement of the tabs in the search window and the information window. The larger the number, the farther to the right the tab is in the search window and the information window.
- **Order No.** — Specifies the order in which the characteristics display in the information window. If the value of Order No. is 0, the characteristics display on the tab in the same order as in the Catalog Characteristics list box.
- **Dispose Order** — Specifies the position of individual characteristics within the search results list. Characteristics are arranged in ascending order according to their number.
- **Sort Order** — Defines the order in which the EDM Library software sorts the characteristics within the search results list. The characteristic with the lowest number display first when sorting. A positive number indicates an ascending sort (the default). A negative number indicates a descending sort.
- **Default Value** — Specifies the default value of the characteristic that the EDM Library software automatically assigns when creating a new object. A default value specified in the catalog characteristics list overwrites any Default Value in the **Other** tab of the allocated characteristic.
- **Charact. Status** — Represents the status of the characteristic with reference to the Inherit, Mandatory Field, and Option List status bits. When adding a new row in the catalog characteristic list or modifying an existing row, an EDM administrator can either activate or deactivate these properties in the Catalog Characteristics window.

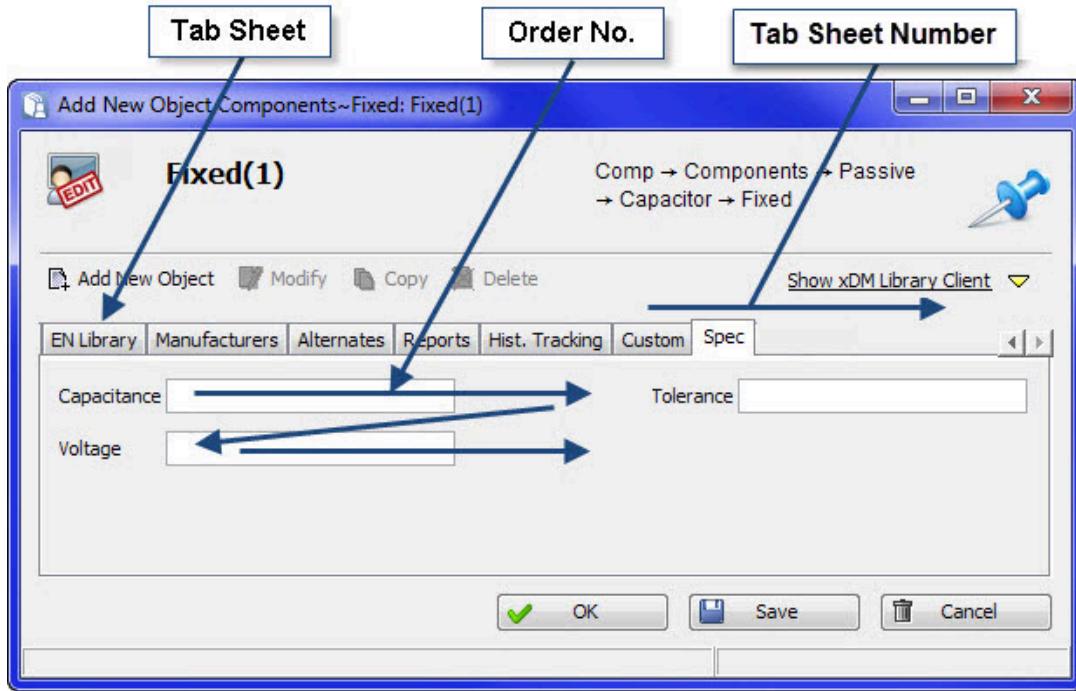
The **Status** tab of each characteristic also contains values for these status bits (refer to “Characteristic Administration” on page 103). If activated in the Characteristic definition, those status values have a higher priority than properties set individually in the **Top** tab of a catalog group. For example:

- If the respective status bit (for example, Inherit) is not set in the **Status** tab of the characteristic, it may be set individually for the characteristic.
- Once the status bit of the characteristic is set, it cannot be deactivated by typing 0 in the Characteristic Status field inside the **Top** tab of the catalog group.

To assign different properties in different catalog groups, be sure to deactivate these initially in the **Status** tab of each characteristic.

The Components information window in [Figure 37](#) shows the effect of the Tab Sheet and Tab Sheet No. values.

Figure 37. Components Information Window With Dynamic Characteristics



Related Topics

[Opening a Catalog Group Information Window](#)

[Catalog Group Key](#)

[Catalog Group Object - Generator Tab](#)

[Catalog Group Object - Admin Tab](#)

[Catalog Group Object - GUI Tab](#)

[Catalog Group Object - Rights Tab](#)

[Catalog Group Object - History Tab](#)

[Object Classes Object - Status Tab](#)

Catalog Group Object - Generator Tab

To access: Refer to "[Opening a Catalog Group Information Window](#)" on page 80.

The **Generator** tab for a Catalog Group object manages the automatic generation of object IDs when creating new objects.

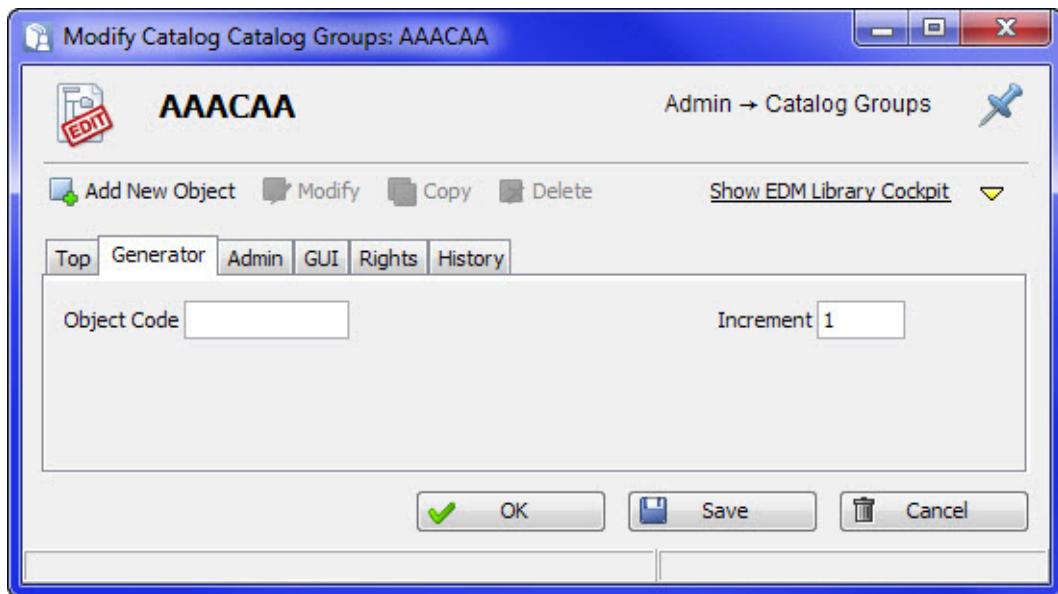
Description



Note:

The automatic generation of object IDs is only active if the Generate Number status flag on the **Status** tab of the corresponding object class is active.

Figure 38. Generator Tab of a Catalog Group



Characteristics

- **Object Code** — Identifies the catalog group within the database. If objects are created below this catalog group, the object code automatically prefixes the object name.

For example, object code E could be used to identify objects inside an Electronic Devices catalog group. A new object in the Electronic Devices catalog group would be assigned the automatically generated key E00001.

The Generate Number status flag of an object class must be set to use the Object Code feature.

- **Increment** — Defines the step by which the number generator increments the object ID when creating a new object. The default value is 1.

Related Topics

[Opening a Catalog Group Information Window](#)

[Catalog Group Key](#)

[Catalog Group Object - Top Tab](#)

[Catalog Group Object - Admin Tab](#)

[Catalog Group Object - GUI Tab](#)

[Catalog Group Object - Rights Tab](#)

[Catalog Group Object - History Tab](#)

[Object Classes Object - Status Tab](#)

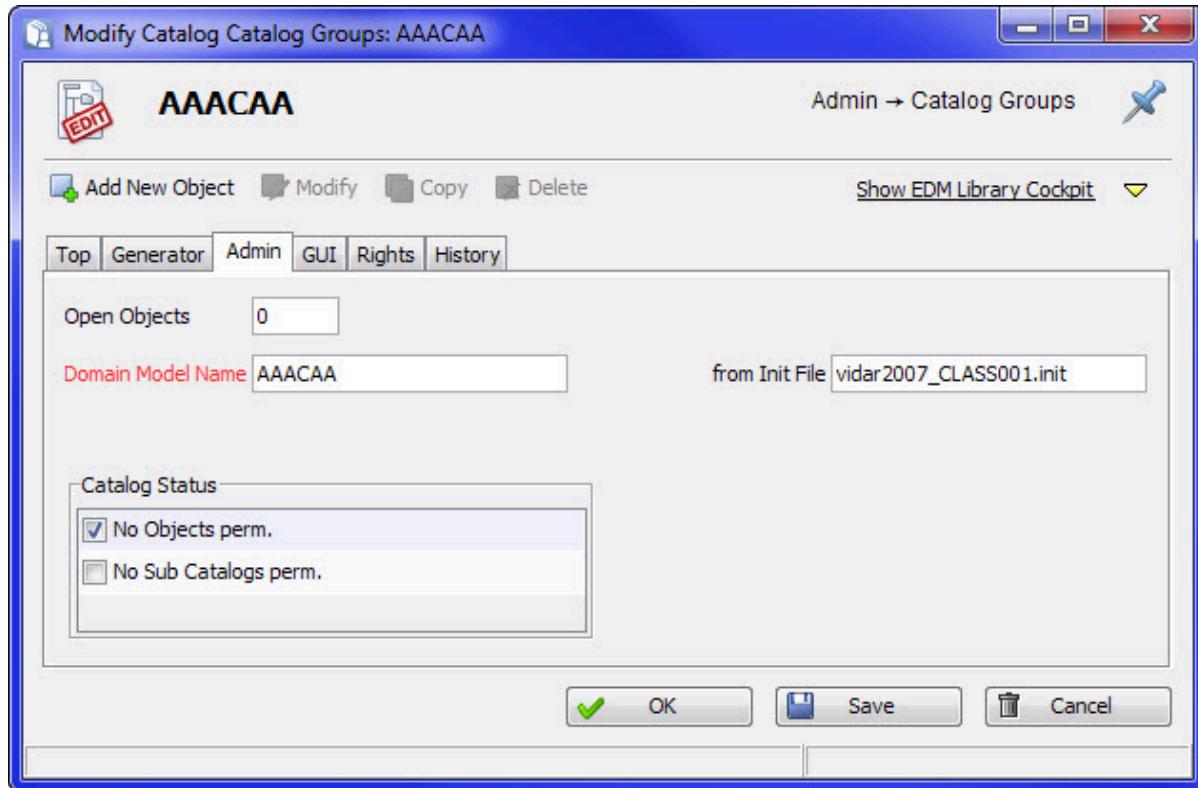
Catalog Group Object - Admin Tab

To access: Refer to “[Opening a Catalog Group Information Window](#)” on page 80.

The **Admin** tab for a Catalog Group object contains the characteristics and flags relating to catalog group administration.

Description

Figure 39. Admin Tab of a Catalog Group



Characteristics

- **Open Objects** — Displays how many objects are opened (locked) within this catalog group. An administrator will typically check this field to make sure that no objects are open before making modifications to this Catalog Group object.
- **Domain Model Name** — API client programs use this field to identify a catalog group and determine a path through the EDM data model to a given characteristic (refer to “[PathQuery Dialog Box - Path Builder Tab](#)” on page 359). The Domain Model Name value must be unique and not duplicated in other Catalog Group objects. Although not required, the convention is to

set the Domain Model Name to the value of the catalog title or catalog group code (for example, AAACAA). To check if a Domain Model Name is unique, use the **domain_model_checker** command located in the <SDD_HOME>/dms/bin directory. The command checks for catalog groups without a domain name and catalog groups with the same domain name, and returns a message if either condition occurs.

New catalog groups must have a Domain Model Name value.



CAUTION:

Do not change the value of Domain Model Name in default catalog groups that are part of the EDM data model (such as the top level catalog group for an object class), as this can break functionality in API client programs shipped with the software.

Similarly, changing the Domain Model Name of any existing custom catalog group runs the risk of breaking any custom API program that accesses data using a path containing the Domain Model Name.

- **from Init File** — If the process of loading an initialization file created the catalog group, the EDM Library software writes the name of the initialization file here. When creating a catalog group, typing a value in the **from Init file** field enables the **batchadmin** tool to automatically write the catalog group definition to that initialization file.
- **Catalog Status** — The following administration flags either permit or prohibit a catalog from containing either objects or other subcatalogs:
 - **No Objects perm** — No checkmark specifies that objects can be created within this catalog group. A checkmark specifies it is not possible to create objects within this catalog group.
 - **No Sub Catalogs perm** — No checkmark specifies that subgroups (lower-level catalog groups) can be created within this catalog group. A checkmark specifies it is not possible to create subgroups.

Related Topics

[Opening a Catalog Group Information Window](#)

[Catalog Group Key](#)

[Catalog Group Object - Top Tab](#)

[Catalog Group Object - Generator Tab](#)

[Catalog Group Object - GUI Tab](#)

[Catalog Group Object - Rights Tab](#)

[Catalog Group Object - History Tab](#)

Catalog Group Object - GUI Tab

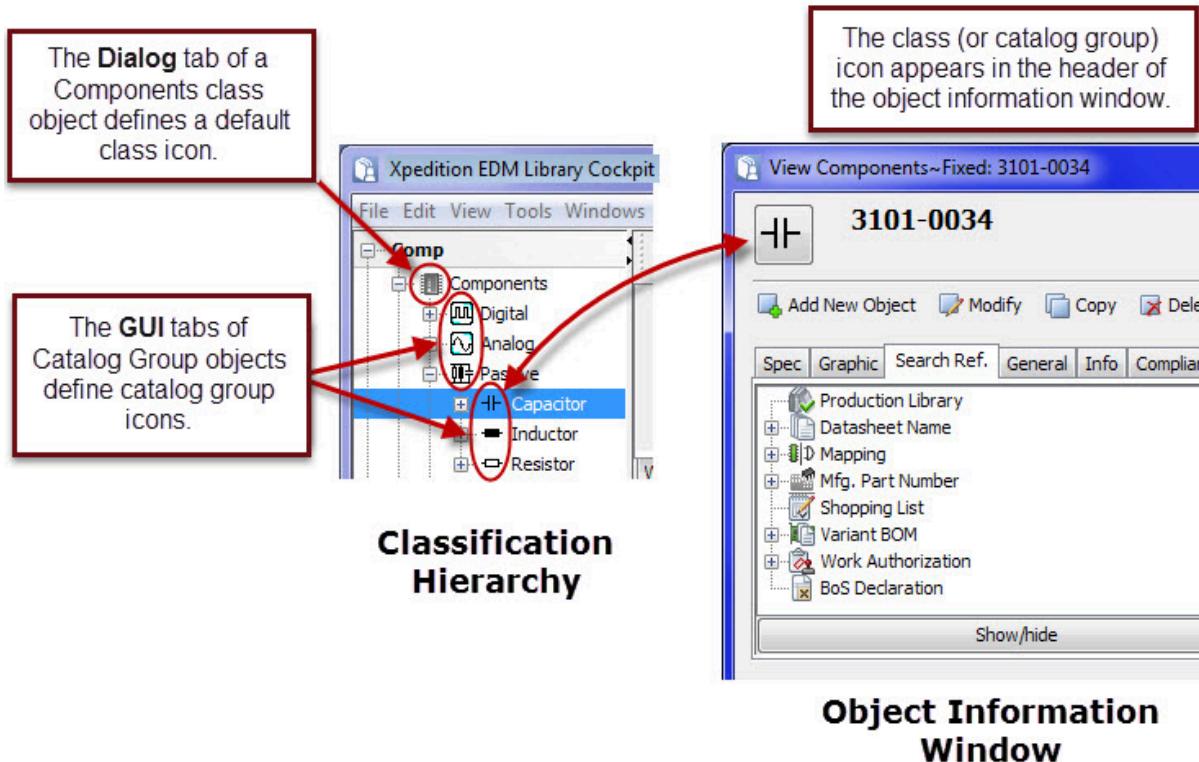
To access: Refer to “[Opening a Catalog Group Information Window](#)” on page 80.

The **GUI** tab of a Catalog Group object defines the icon that displays in Xpedition EDM Library Cockpit and other client applications next to the catalog group in the classification hierarchy and at the top of an information window of an object that is a member of the catalog group.

Description

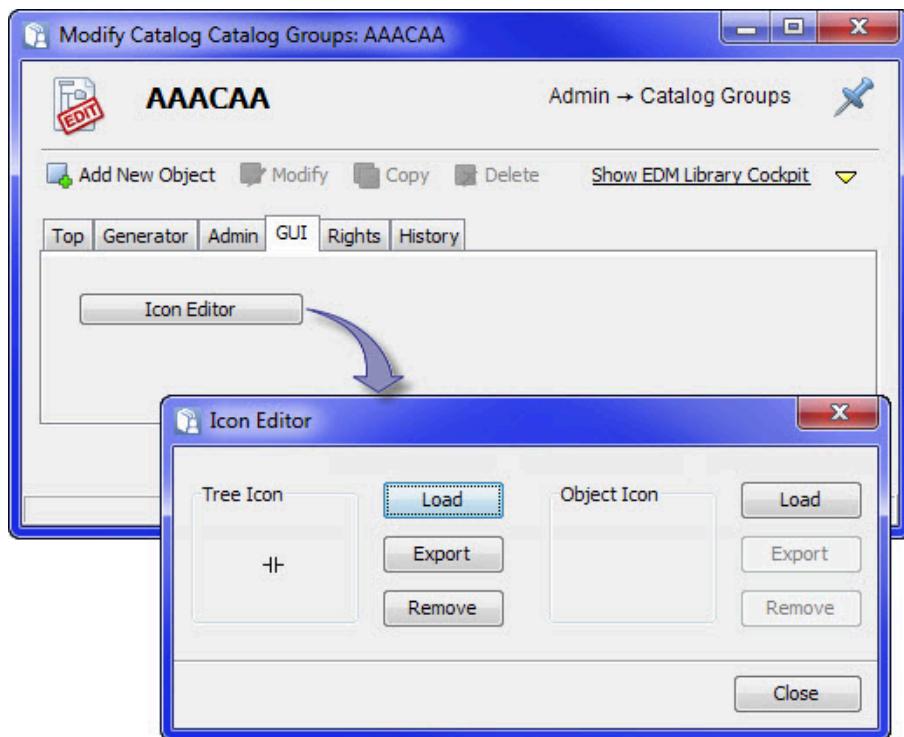
The EDM data model stores the icons that show in classification trees and in the title bar of an information window. Such icons, which an administrator can import into or export from a database, provide quick visual identification of an object type within EDM Library Cockpit and other applications. The default data model defines icons for all standard object classes (such as components, suppliers, and mappings), but does not have default icon definitions for any standard or custom catalog groups associated with the object class. [Figure 40](#) shows an example of icons defined in class and Catalog Group objects and how they display in EDM Library Cockpit.

Figure 40. Icons Defined by Class and Catalog Group Objects



Use the **GUI** tab of a Catalog Group object to associate an icon with a particular catalog group ([Figure 41](#)). A catalog group does not have to have an icon. When a catalog group does not have an icon definition, then no tree icon displays in the classification hierarchy for that catalog group and an information window uses the parent catalog group or class icon in the header.

Figure 41. GUI Tab of a Catalog Group



Characteristics

- **Icon Editor** — Clicking the button produces a dialog box with three buttons:
 - **Load** — Displays a navigation dialog box with which to import a bitmap image from the file system into the data model and associate it with the catalog group or an object from that catalog group. Images can be in .png, .jpg, or .gif format.
For compatibility with existing icons, a tree icon image should be 16 x 16 pixels, while an object icon should be 32 x 32 pixels.
 - **Export** — Displays a navigation dialog box with which to export the icon to the file system as a bit map image, where it can then be edited.
 - **Remove** — Removes the icon from the data model so that the catalog group no longer has an associated icon.

Usage Notes

Notice that [Figure 41](#) only defines a tree icon for the capacitor catalog group (AAACAA), but does not define an object icon. When an object icon is not specified for the catalog group, then the object information window header uses the tree icon. Adding an object icon to the **GUI** tab of the catalog group overrides the display of the tree icon in the object window header with the specified object icon. Tree icons and object icons are often similar images (with object icons usually having better resolution than the tree icon), but they do not need to be similar.

Related Topics

[Opening a Catalog Group Information Window](#)

[Catalog Group Key](#)

[Catalog Group Object - Top Tab](#)

[Catalog Group Object - Generator Tab](#)

[Catalog Group Object - Admin Tab](#)

[Catalog Group Object - Rights Tab](#)

[Catalog Group Object - History Tab](#)

Catalog Group Object - Rights Tab

To access: Refer to “[Opening a Catalog Group Information Window](#)” on page 80.

An administrator uses the **Rights** tab to tune user rights from the broad class rights inherent in a user role, to more specific user-based rights applicable only to objects residing in the catalog group.

Description

Rows in the User Rights list on the **Rights** tab define the view or edit rights of user or user groups to objects in the catalog group. If the User Rights list is empty (the default for new catalog groups), the catalog group inherits the class rights of the user role assigned to the user, or rights defined in the parent catalog group. You cannot grant greater rights to a catalog group than permitted at the class level by the user role. For example, if you add lindseyt to the User Rights list with edit rights, but the user role attached to lindseyt's user account only has view class rights, then lindseyt still only has view rights, and cannot edit objects in the catalog group.



Note:

As soon as one entry exists for one user in the User Rights list (whether or not that user has edit or view access permission), all other users lose their access rights to the catalog group unless you explicitly add their usernames. If another user attempts to access the catalog group and is not in the User Rights list, an access denied error message displays.

Figure 42. Rights Tab of a Catalog Group

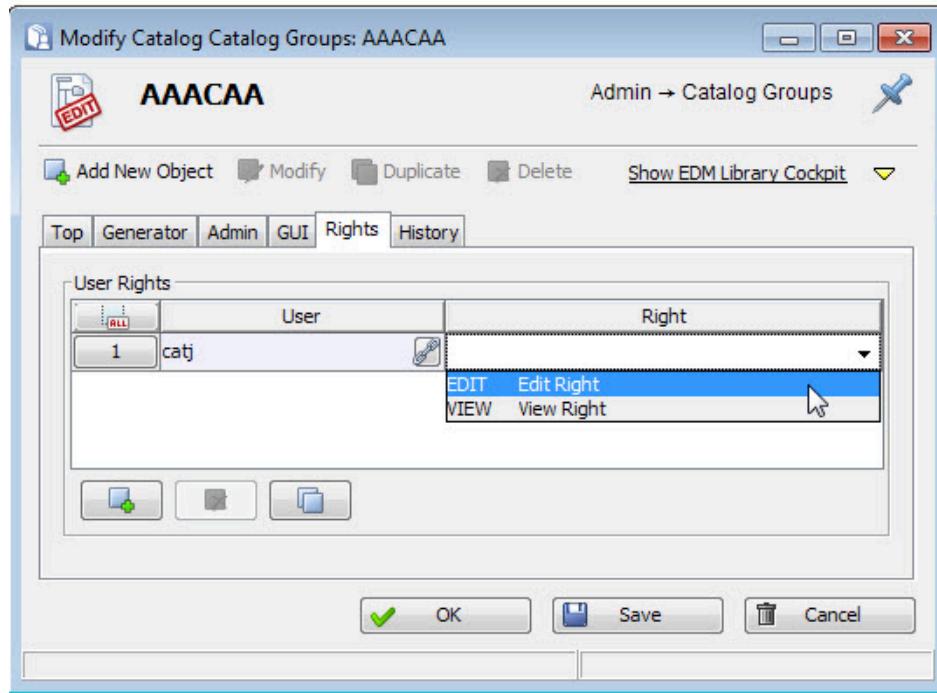


Table 6. Catalog Group Rights

Class Rights Setting (User Role)	Top Catalog Rights Setting	Sub Catalog Rights Setting	Resulting Object Right in Sub Catalog
View	View	View	View
View	View	Edit	View
View	Edit	View	View
View	Edit	Edit	View
Edit all	View	View	View
Edit all	View	Edit	Edit
Edit all	Edit	View	Edit
Edit all	Edit	Edit	Edit

Characteristics

The User Rights list contains the following two columns:

- **User** — A username or name of a user group.

Clicking the reference button in the User column displays a **User** tab in the search window with which to search for user names or user groups, and then return a value to the column.

- **Right** — The access permission to the catalog group for the named user or user group.

- **View** — The user or group can open objects in this catalog group for viewing only.

- **Edit** — The user or group has full edit permission.

If a user has edit permission to the Top catalog group, and creates a new subgroup, the user automatically inherits edit permission to the subgroup. Therefore, it is only possible to add new catalog groups to the tree if the user has the edit access permission for the parent catalog group (either explicitly or through inherited class rights).

- **No Entry** — If a user is specified in the User Rights list, but has no entry in the Right column (as shown in [Figure 42](#)), then the user has no catalog group access. That is, the catalog group is visible in the classification tree, but they have no permission to edit or view objects in the catalog group.

If the User Rights list has a single row, and the specified user has no rights entry, then only a superuser has permission to the catalog group. Catalog rights must be explicitly defined for all other users, or the users do not have catalog group access.

The User and Right characteristics also display in the **Rights** tab of the Catalog Group search window enabling both of these characteristics to be used to search for catalog groups to which only specific users have a specific right (EDIT or VIEW).

Usage Notes

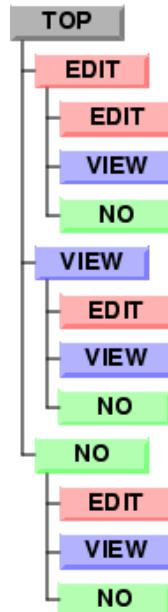
Child catalog groups can inherit the permissions from their parent catalog group. The following rules are valid for inheritance:

- EDIT rights have higher priority than VIEW rights. VIEW rights have higher priority than no rights (NO).
- Higher-priority rights of a catalog group take precedence over lower-priority rights of a superordinate catalog group.

The following example illustrates the access permissions hierarchy:

1. An administrator creates a catalog tree with the following rights:
 - The access permissions of the Top catalog group are not yet specified.
 - The administrator assigns EDIT, VIEW and NO rights to the subgroups.

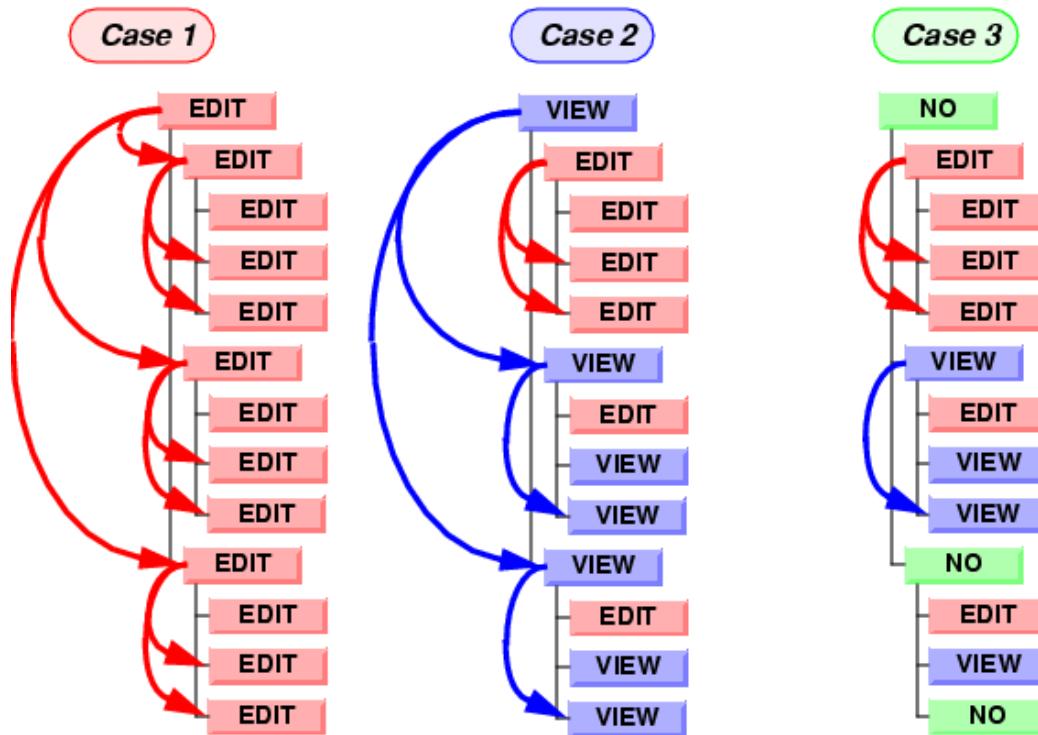
Figure 43. Example Allocation of Catalog Group Permissions (Top Catalog Rights Not Specified)



2. The administrator changes the rights in the Top catalog group. There are three possible effects that changing the rights to the Top catalog group can have on the subgroups, as shown in [Figure 44](#):
 - **Case 1** — The Top Catalog Group is Allocated EDIT Permission. All subgroups inherit the edit permission from the Top catalog group, regardless of their previous privileges.
 - **Case 2** — The Top Catalog Group is Allocated VIEW Permission. All subgroups with view rights or no rights inherit the view permission from the Top catalog group. Catalog groups with edit permission retain and pass on their privileges to lower-level groups.

- **Case 3** — The Top Catalog Group is Allocated NO Permission. All subgroups retain and pass on their privileges to lower-level catalog groups.

Figure 44. Example Allocation of Catalog Group Permissions With Changes to Top Catalog Rights Specified



Related Topics

[Opening a Catalog Group Information Window](#)
[Catalog Group Key](#)
[Catalog Group Object - Top Tab](#)
[Catalog Group Object - Generator Tab](#)
[Catalog Group Object - Admin Tab](#)
[Catalog Group Object - GUI Tab](#)
[Catalog Group Object - History Tab](#)

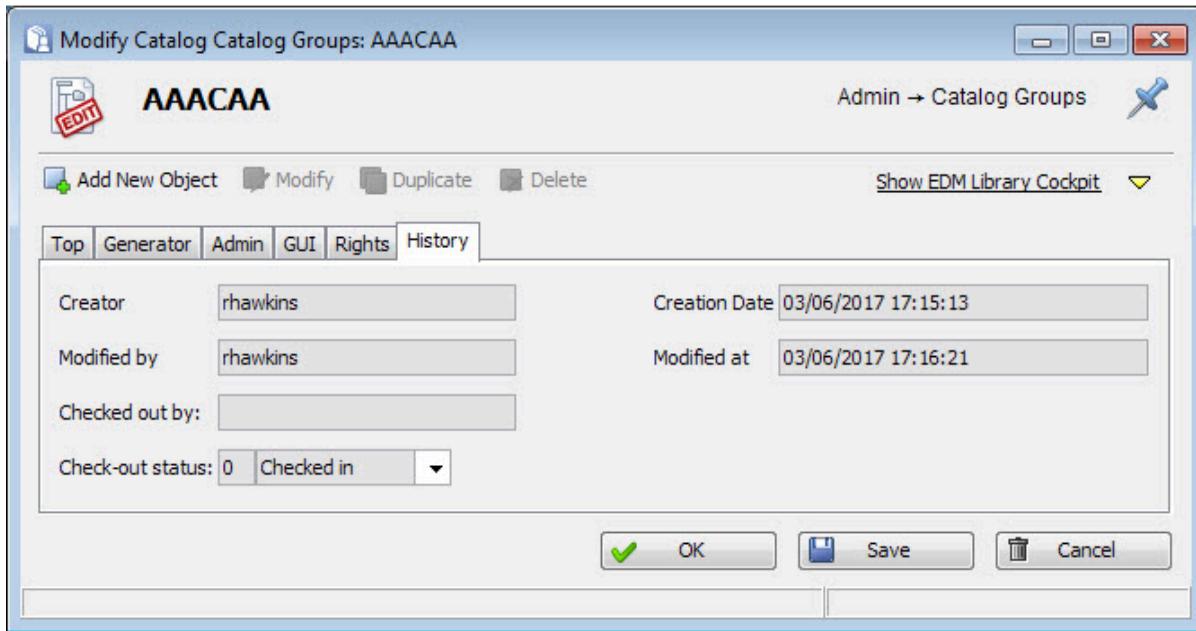
Catalog Group Object - History Tab

To access: Refer to [“Opening a Catalog Group Information Window”](#) on page 80.

The **History** tab of Catalog Group information window contains system-maintained information about the history of the Catalog Group object.

Description

Figure 45. History Tab of a Catalog Group



Characteristics

The **History** tab contains the following characteristics with values maintained by the system:

- **Creator** and **Creation Date** — The user who created the catalog group and the timestamp of when they created the object.
- **Modified by** and **Modified at** — The user who last modified the catalog group and the timestamp of when they made the modification. The system automatically updates these fields when a user edits an object.
- **Checked out by** — The user that has the object checked out. Checking out an object prevents others from editing that object.
- **Check-out status** — A flag that, when set to “1” indicates the object is checked out and cannot be edited by a user other than the one identified by the Checked out by characteristic.

Related Topics

[Enabling History Tracking](#)

[Opening a Catalog Group Information Window](#)

[Catalog Group Key](#)

[Catalog Group Object - Top Tab](#)

[Catalog Group Object - Generator Tab](#)

[Catalog Group Object - Admin Tab](#)

[Catalog Group Object - GUI Tab](#)

[Catalog Group Object - Rights Tab](#)

Characteristic Administration

The Characteristics class contains the definitions for the characteristics that display in the search and information windows of a class.

Every existing class or new custom class requires a minimum set of [Management Characteristics](#) that are necessary for built-in functionality to work on that class. An administrator (or other authorized user) can create any number of additional [Class Characteristics](#) and [Dynamic Characteristics](#) to store and display information, establish relationships, or perform actions as appropriate for the purpose and data stored within the objects of the class.

This section contains several topics with more information about defining and managing characteristics for an object class.

Table 7. Characteristic Administration Topics

<p>Conceptual information about Characteristic objects:</p> <ul style="list-style-type: none">• Classes, Class Number, and Class Tables• Management Characteristics• Class Characteristics and Dynamic Characteristics• Maximum Number of Characteristics in a Table• Characteristic Types• Status Administration• Release and Revision Control• Reference Characteristics• Using Compose Mode to Edit Characteristic Definitions
<p>Reference information about the Characteristic information window:</p> <ul style="list-style-type: none">• Opening a Characteristic Information Window• Characteristic Object - Top Tab• Characteristic Object - Search Ref Tab• Characteristic Object - Table Tab• Characteristic Object - Placement Tab• Characteristic Object - Status Tab• Characteristic Object - Rights Tab• Characteristic Object - Other Tab• Characteristic Object - History Tab
<p>Characteristic creation tasks:</p> <ul style="list-style-type: none">• Creating a Standard Characteristic• Creating a Characteristic to View Data From a Foreign Class• Creating a Characteristic With a Value That is an Arithmetic Function

Table 7. Characteristic Administration Topics (continued)

- Creating an Object Reference Characteristic
- Creating an Action Button Characteristic
- Creating a List (List Frame Characteristics and List Columns)
- Setting User-Based or Group-Based Characteristic Editing Permissions

Classes, Class Number, and Class Tables

Each object class resides in the EDM Library data model as its own table or group of related tables (for example, te_sachnr is the name of the main class table of the Components class in the default data model). Columns in the class table hold values for most, but not all, characteristics for that object class.

Every class has a unique numeric key value called the class number. For example, class number 1 identifies the Components object class, class number 3 identifies the Package object class, class 111 identifies the Request object class, and so on. The class number is important because it must occur at the beginning of a class characteristic name to identify to EDM Library software the class in which the characteristic displays. When prefixed to a characteristic name, the class number should consist of three characters, with leading zeros if necessary. For example, 001obj_id is the object identifier characteristic for the Components object class (class 1), and 055partition holds the partition name for objects in the Production Library class (class 55).

Searching for objects in Admin > Object Classes lets you display important information about classes residing in the data model, such as the class number and class table, so that you have the necessary information when creating new characteristics to include in the class (Figure 46).

Figure 46. Object Class Search Results

	Status	Class Number	Table Name	Class Name
1	S	0	te_dyn	Dynamic Characteristics
2	A	1	te_sachnr	Components
3	A	3	te_pack	Package
4	S	8	te_glb_rastw	Enumerators Dynamic Values
5	A	9	te_shop	Shopping List
6	A	10	te_baust	Mapping

The data model contains many default object classes. You can only create new custom classes if you have administrator privileges, and you acquire a Developer license when connecting to the EDM Server. Once an object class exists, administrators using a Librarian license can add new characteristics or edit characteristics associated with that class. When a developer creates a new class, he or she must assign the management characteristics to the class before other users can create objects in the class, or an administrator can add additional characteristics that make the class useful.

To add a new characteristic to a class, an administrator must specify a specific table in the database and identify the column within the table in which EDM Library applications can store the characteristic information. Creating a new characteristic automatically creates the table column (if it does not exist) in the table when you save the characteristic definition. However, deleting a characteristic does not delete the table column or any data residing in that column from the database.

Standard characteristics must exist as columns in the class table, with a unique column name that is not a reserved word in SQL. List characteristics and dynamic characteristics require storage in the data model in tables separate from the class table. For a list of reserved SQL words, refer to your database documentation.



Note:

When modifying the definition of existing characteristics, to see the effect you often have to reload the data model cache using the **File > Refresh Data Model** or **Tools > Administration > Reload Data Model** menu item in EDM Library Cockpit.

Related Topics

[Management Characteristics](#)

[Class Characteristics and Dynamic Characteristics](#)

[Maximum Number of Characteristics in a Table](#)

[Characteristic Types](#)

[Status Administration](#)

[Release and Revision Control](#)

[Using Compose Mode to Edit Characteristic Definitions](#)

[Opening a Characteristic Information Window](#)

Management Characteristics

For object classes where users can create and edit objects within the class (most classes within the data model) certain management characteristics must reside in the base table of the object class and follow certain naming conventions. These management characteristics have the same purpose in each object class. Except for the `<class_no>obj_id` master key that identifies objects in the class, management characteristics do not always have to be made visible in the search window or information window. Management characteristics that are visible can be placed anywhere and do not have to reside on the same tab in the search window or information window.

[Table 8](#) lists management characteristics for the Components class (which has a class number of “1”) and the table name and column in the data model where those characteristics reside. [Figure 47](#) shows where the management characteristics are on the **General** tab of a component information window. Other classes have similar characteristics whose name differs by the class number (for example, `003obj_id` as a key identifier for class 3 [Package class], `021bearbeit` holds the name of the last user who modified an object in class 21 [variant BOM], and so on).



Note:

Every class must have a <class_number>obj_id characteristic to store the unique identifier of each object within that class. The unique ID for any given class must always reside within a characteristic named according to the <class_number>obj_id naming convention (for example, 002obj_id, 003obj_id, and 004obj_id).

Table 8. Management Characteristics (Components Class)

Characteristic Name	Purpose	Class Table Name	Class Table Column
001obj_id	A unique identifier that acts as the master key for the object. A user usually types a value when creating an object, although the system can auto-generate the value or form a value by combining the values of other characteristics if appropriate for the class.	te_sachnr	obj_id
001obj_status	The status of the object. A user usually chooses a value from a predefined option list, but the value can also be generated automatically as an object moves through a defined process.	te_sachnr	usr_sts
001obj_txkg	The name of the catalog group to which the object belongs. The system supplies the value (which is non-editable) based on the catalog group where the object is created.	te_bnk	bnk_txt
001co_user	The name of the user who has an object checked out (empty when an object is checked in). The system supplies the value, which is non-editable. The characteristic must exist to use check out/check in functionality.	te_sachnr	co_user
001co_status	Specifies whether an object is checked out or checked in. The system supplies the value, which is non-editable. Characteristic must exist to use check out/check in functionality.	te_sachnr	co_status
001ersteller	Name of the user that created the object. The system	te_sachnr	ersteller

Table 8. Management Characteristics (Components Class) (continued)

Characteristic Name	Purpose	Class Table Name	Class Table Column
	supplies the value, which is non-editable.		
001erst_date	Creation date of the object.	te_sachnr	erst_date
001bearbeit	Name of the user who last modified the object. The system supplies the value, which is non-editable. Characteristic must exist to use history tracking functionality.	te_sachnr	bearbeit
001obj_datum	Date when the object was last modified. The system supplies the value, which is non-editable.	te_sachnr	datum

Figure 47. Management Characteristics on the General Tab of a Component Information Window

Modify Components~Multiplexer: COMP-10005

COMP-10005 Comp → Components → Digital → Glue Logic → Multiplexer

Add New Object Modify Duplicate Delete Sh

001obj_status **001texkg**

Status	U Under Construction
Part Number	COMP-10005
Description	3 to 8 Decoder/Demultiplexer
Long Description	
Catalog Group	Multiplexer

001obj_id

001co_user

Checked Out By: **001erst_date** Check-Out Status: 0 Checked in

Creator: bgraves Creation Date: 08/14/2007 13:20:18

Last Modified By: admin Last Modified Date: 11/14/2011 16:36:07

Sync Exclude: 1 Yes

001bearbeit **001ersteller** **001obj_datum**

OK Save

When a developer creates a new object class, it is customary to create the `<class>obj_id` manually to reflect the format requirements of the unique identifier for that class (for example, the length of identifier, whether the identifier must match a certain pattern, and so on).

To apply status administration to a specific object class, the class must have a `<class_number>obj_statu` characteristic. When creating the `<class_number>obj_statu` characteristic, enter status values (the defaults are A, D, R, U, or X) in the **Top** tab of the characteristic in the Option List list characteristic. EDM Library software also uses the default value (**Other** tab) for status administration. Objects whose status is set to the default value (usually U) can always be edited. Objects with a different status can only be edited by users with EDIT permission (refer to “[User-Dependent Status Administration](#)” on page 126).

A developer can create the remaining management characteristics by copying the characteristic from a class in the default data model (for example, from the Components class) and then changing only the information in the characteristic definition that applies to the new class. Copying a management characteristic requires making the following minimal changes:

- Changing the name of the characteristic, the class number, and the reference class on the **Top** tab of the Characteristic object to reflect the class number of the new class.
- Changing the name of the class table on the **Table** tab of the characteristic definition.
- Changing other placement parameters of the characteristic (such as whether the characteristic is visible in a search window or information window, the tab on which the characteristic displays, and so on) is optional.

Remaining characteristics can be configured to meet the individual needs of the user community.

Related Topics

[Classes, Class Number, and Class Tables](#)

[Class Characteristics and Dynamic Characteristics](#)

[Characteristic Types](#)

[Status Administration](#)

[Release and Revision Control](#)

[Using Compose Mode to Edit Characteristic Definitions](#)

[Opening a Characteristic Information Window](#)

[Characteristic Object - Top Tab](#)

[Characteristic Object - Table Tab](#)

[Characteristic Object - Placement Tab](#)

Class Characteristics and Dynamic Characteristics

Characteristics can be either class characteristics or dynamic characteristics. Class characteristics are associated with an object class and display in objects that reside in all the catalog groups of an object class. Dynamic characteristics are input fields that only display in certain subcatalog branches of the class hierarchy.

For example, a resistor and a capacitor are both objects in the Components class, but reside in different subcatalogs. A resistor has a resistance characteristic, and a capacitor has a capacitance characteristic, which are unique to the resistor and capacitor subcatalog groups. Each dynamic characteristic has a different associated Unit value.

[Table 9](#) shows the configuration differences between class characteristics and dynamic characteristics.

Table 9. Class and Dynamic Characteristic Configuration Differences

	Characteristic Name	Class	Table
Class Characteristics	<Class-No.><Name> (for example, '001obj_id')	<Class-No.> (for example, '1')	te_<class-table> (for example, 'te_sachnr')
Dynamic Characteristics	000<Name> (for example '000inductance')	0	te_dyn

Defining one or several dynamic characteristics does not automatically assign them to a catalog group. To add dynamic characteristics to a specific catalog, you must open a Catalog group in modify mode and edit the Catalog Characteristics list on [Catalog Group Object - Top Tab](#). Inside the information window of the sub-list, activate the Inherit flag to have the dynamic characteristic also show in all sub-catalogs below the modified catalog object.

Related Topics

[Classes, Class Number, and Class Tables](#)

[Management Characteristics](#)

[Maximum Number of Characteristics in a Table](#)

[Characteristic Types](#)

[Status Administration](#)

[Release and Revision Control](#)

[Using Compose Mode to Edit Characteristic Definitions](#)

[Opening a Characteristic Information Window](#)

Maximum Number of Characteristics in a Table

To avoid performance degradation when connecting to a database and using EDM Library applications, an administrator should make sure that no table holding custom characteristics exceeds 500 columns.



Note:

When customizing a database by creating custom characteristics, make sure that the te_dyn table, or any te_<class> table, does not exceed 500 columns (characteristics). Exceeding the limit can result in performance degradation or possible error messages from the Oracle or PostgreSQL server software.

In the default EDM data model, most te_<class-table> tables have less than 50 default columns, enabling an administrator or developer to add up to 450 columns of custom class characteristics. If the te_dyn

Core Object Class Administration

Maximum Number of Characteristics in a Table

table holding dynamic characteristics is full and has reached the 500 column recommended maximum, create another table in the EDM database for more dynamic characteristics by doing the following:

- Creating a copy of the 001dyn_id characteristic with the following changes:
 - Change the name of the characteristic (for example, to *001dyn_id1*).
 - Specify another table name inside the **Table** tab sheet (for example, *te_dyn1*).
- Filling the table with the obj_id values of te_dyn using the following SQL command from an sqlplus shell window:

```
SQL> insert into te_dyn1 (obj_id) (select obj_id from te_dyn);
```

A quick way to determine how many characteristic columns a table has is to do an advanced search on the Characteristic object class, with the name of a table specified on the **Table** tab and all other search fields left empty. If your database supports multiple languages, be sure to select only one language to avoid duplicate entries in the search results. **Figure 48** shows the results of searching the characteristics class with the *te_sachnr* table specified in the search criteria (*te_sachnr* is the class table for the Components object class).

Figure 48. Search Results Showing the Column Count in the *te_sachnr* Table

The screenshot shows the Xpedition EDM Library Cockpit interface. On the left, there's a navigation tree with categories like Comp, Design, Library, Compliance, Library Administration, Design Config, and Admin. Under Admin, 'Characteristics' is selected. A callout box points to this selection with the text 'Select the Characteristics object class.' In the main pane, a search results table is displayed under the 'Characteristics' tab. The table has columns: ALL, Status, Characteristic, Ref. Class, Charact. Type, Value Type, Tab Sheet, Information Text, Table ..., and ...'. There are 11 rows listed. A callout box points to the 'Table Name' field in the search criteria section with the text 'Specify the table name.' Another callout box points to the bottom-left corner of the table area with the text 'The te_sachnr table has 44 characteristic columns.' A third callout box points to the status bar at the bottom left with the text '44 hit(s)'. The status bar also shows 'EDM Library Services' and 'LIB Library'.

The 500 characteristic limit applies to both Oracle or PostgreSQL EDM databases.

Related Topics

- [Classes, Class Number, and Class Tables](#)
- [Characteristic Object - Table Tab](#)
- [Management Characteristics](#)
- [Opening a Characteristic Information Window](#)

Characteristic Types

Every characteristic has a type value. Each characteristic type presents itself differently in the a search window or information window from characteristics of a different type.

To define the characteristic type, you must choose one of the following characteristic type values from a dropdown list on the [Characteristic Object - Top Tab](#) when creating the characteristic:

- Standard
- Action Button
- Body Property
- Pin Property
- Bit Status
- List Frame
- Text Frame
- BLOB

[Standard Characteristic Type](#)

[Action Button Characteristic Type](#)

[Body Property and Pin Property Characteristic Types](#)

[Bit Status Characteristic Type](#)

[List Frame Characteristic Type](#)

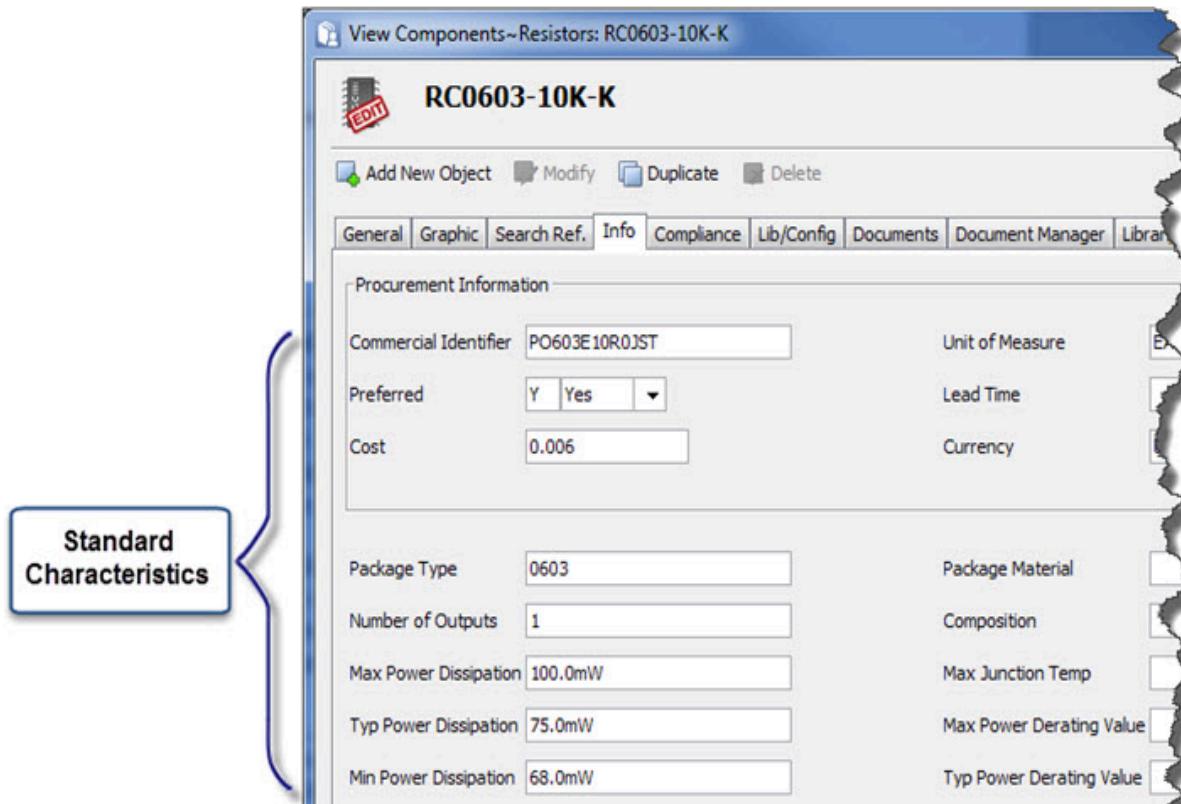
[Text Frame Characteristic Type](#)

[BLOB Characteristic Type](#)

Standard Characteristic Type

A standard characteristic presents the user with a field where they can type in text or choose a value from an option list. Use a standard characteristic type for single-value data (one column, one row). Possible data value types are Char, Bit, Int, Long, and Double. Most characteristics in the data model are standard characteristics.

Figure 49. Standard Characteristics in an Information Window



Related Topics

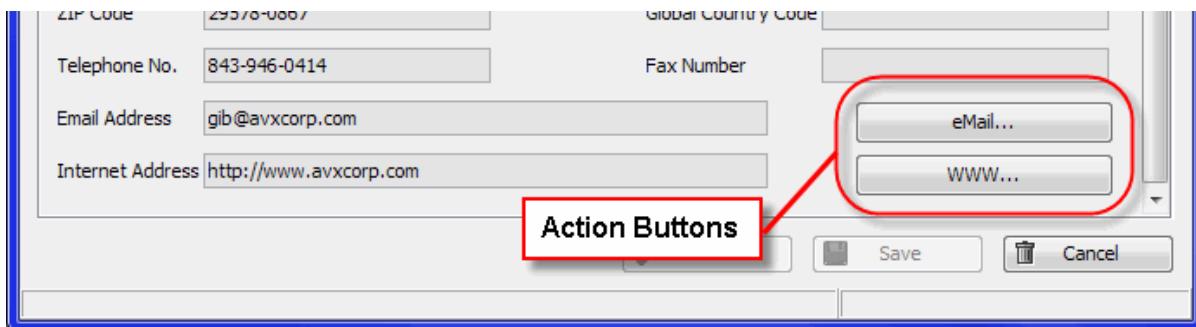
[Creating a Standard Characteristic](#)

Action Button Characteristic Type

Action button characteristic types display as a button in the EDM Library Cockpit user interface that, when clicked, launch an external program. Action buttons can be in the search results list and the information window. Example uses for action buttons are to display a datasheet for a component, launch a browser and go to a web page, or to send email.

The Default Value and Java Method input fields on the **Other** tab of the characteristic definition contain the syntax an EDM Library application uses to launch the program or Java method when the user clicks the action button. Any program you can execute from a shell can be called with an action characteristic type.

Figure 50. An Action Button Characteristic



Action button characteristic types can also display as a column characteristic in a list.

Related Topics

[Characteristic Object - Other Tab](#)

[Creating an Action Button Characteristic](#)

Body Property and Pin Property Characteristic Types

Body Property and Pin Property characteristic types are no longer used for new characteristics, and only remain for backward compatibility with older databases.



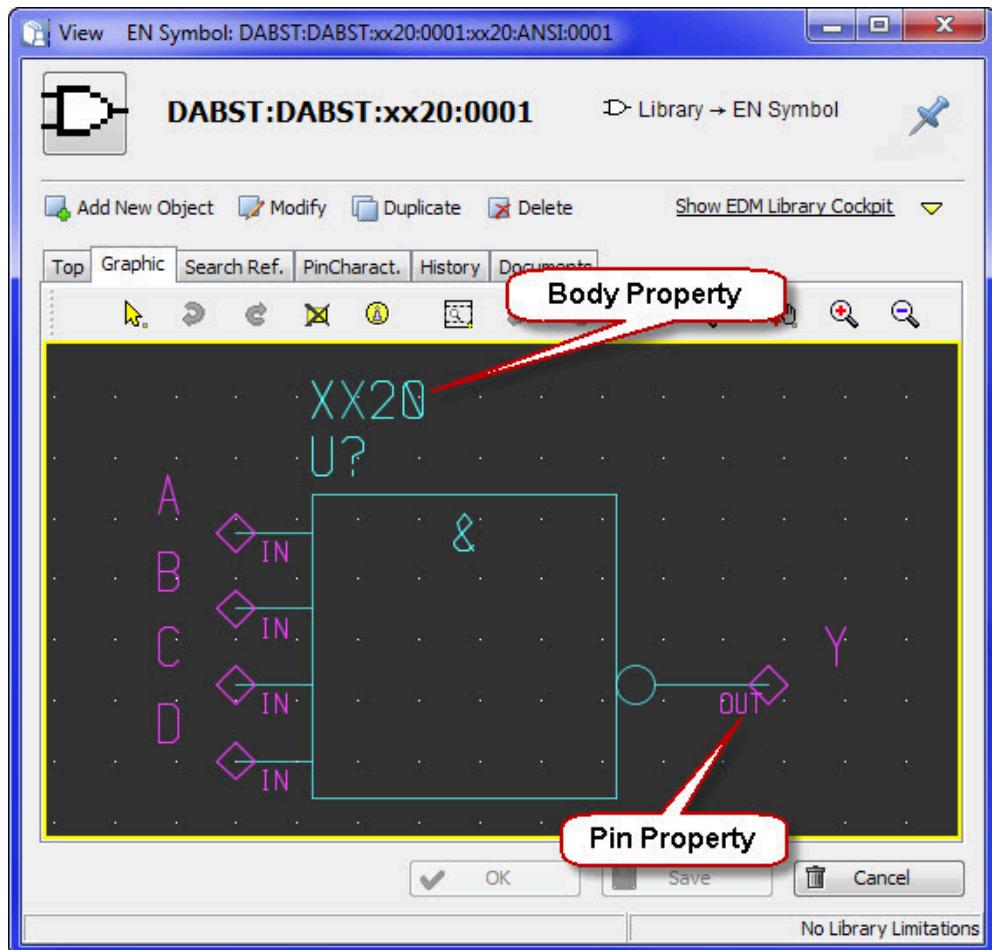
Note:

An administrator should not manually create a characteristic with a type of body or pin property. In older software releases, body and pin characteristic types were associated with graphical symbols created using the original DMS interface or imported from the file system. The EDA library loader programs no longer create or use body property and pin property characteristic types.

In the past, Interface objects and Symbol objects in the EDA library module were the only classes to use body property and pin property characteristics. Examples of body property characteristics are COMP, REF, or VALUE properties. Pin property characteristic types are allocated to interface pins.

The following figure shows an example of body property and pin property characteristics on imported graphics from an older data model.

Figure 51. Pin Properties and Body Properties

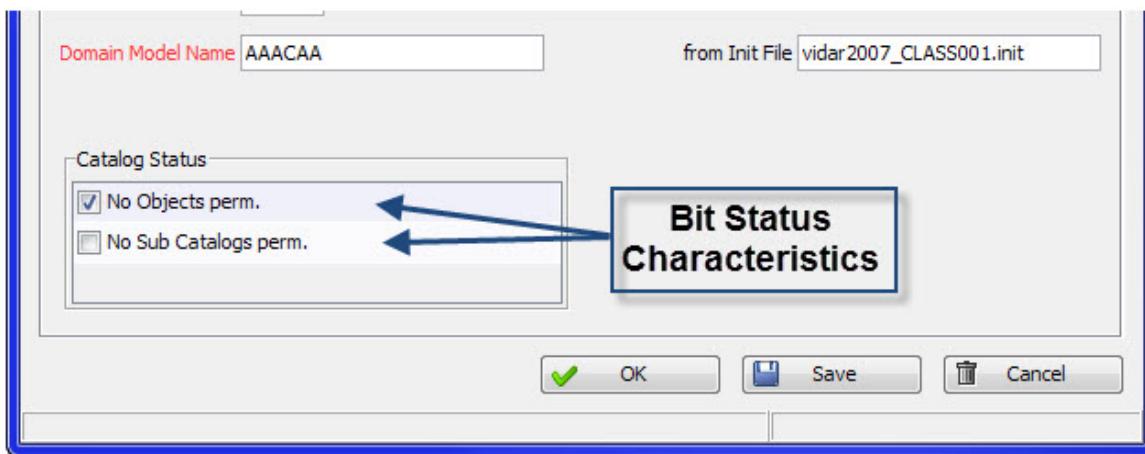


Bit Status Characteristic Type

A bit status characteristic type is an array of true/false values, displayed in a list of checkboxes.

Examples of a bit status characteristic type are the No Objects Perm. and No Sub Groups Perm. characteristics on the **Admin** tab of a Catalog Group information window.

Figure 52. Example of Bit Status Characteristics



A bit status characteristic must be an Integer or Long data type, with corresponding option list values, and it must have the Option List status bit checked on the **Status** tab of the characteristic definition. In addition, the Information Width and Information Height attributes on the **Placement** tab of the characteristic definition must be set correctly to completely display the contents of the bit status characteristic.

Related Topics

[Characteristic Object - Status Tab](#)

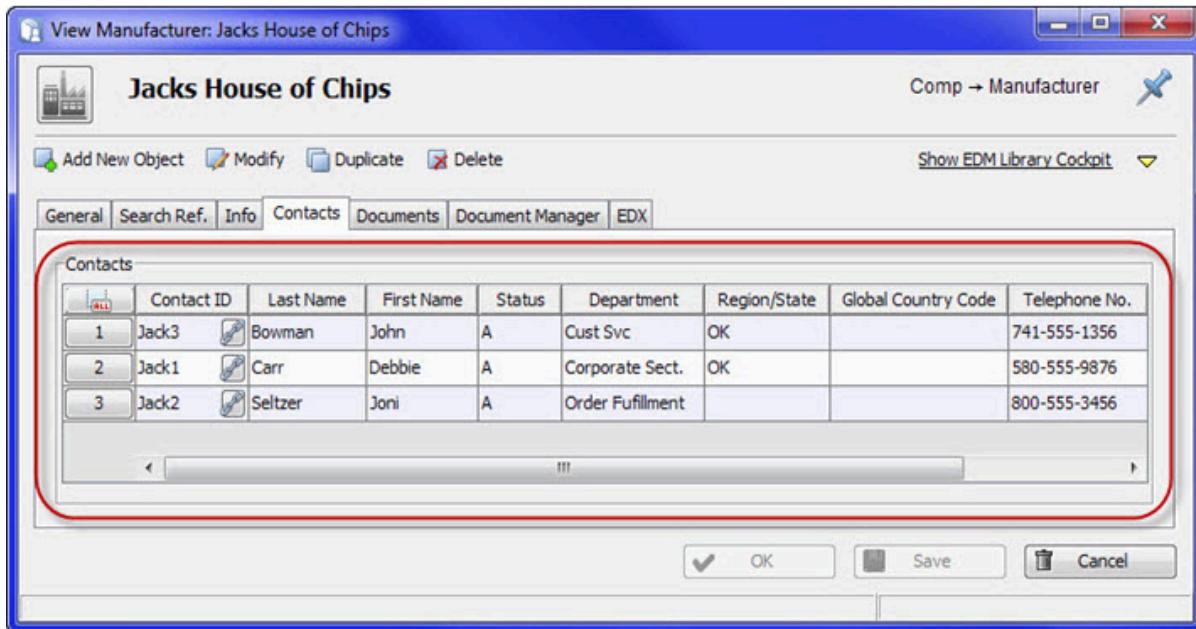
[Characteristic Object - Placement Tab](#)

List Frame Characteristic Type

A list frame displays data in an information window in a tabular list format with several columns and rows.

The following figures shows the Contacts list frame on the **Contacts** tab of a Manufacturing information window.

Figure 53. Example List Frame



Displaying data as a list frame requires a characteristic of the type list frame and an array of other characteristics that form the columns within the list. Other examples of a list frame are an assignment of several manufacturers and their data to a component, or the language-dependent assignment of information or search texts. EDM Library permits construction of list frames that are hierarchical (that is, that have lists within lists). Columns within a list frame can be standard characteristics, bit status characteristics, or action button characteristics.

Figure 54. List Frame Structure

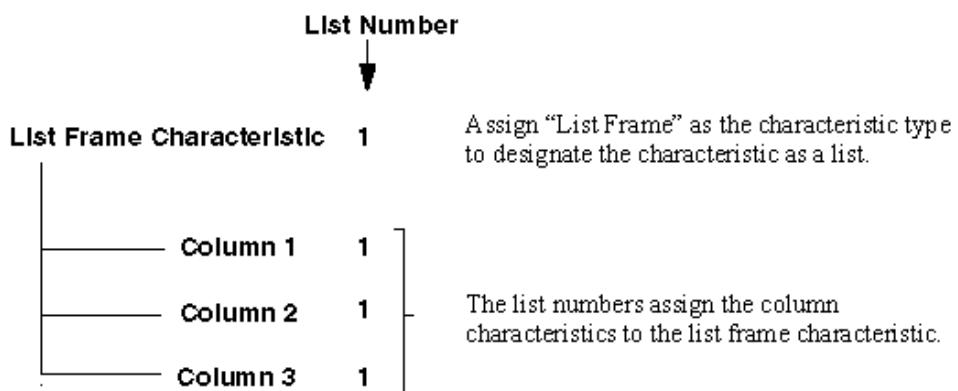


Figure 55. Corresponding Structure of the Database Table

Main Key List Characteristic		Column Characteristics			
Id		Column 1	Column 2	Column 3	Column 4
LM 1		X 11	X 12	X 13	X 14
LM 1		X 21	X 22	X 23	X 24
LM 2		X 31	X 32	X 33	X 34
LM 2		X 41	X 42	X 43	X 44
:	:	:	:	:	:

Figure 56. Section of an Information Window With a List Characteristic

#	Catalog Group	Option	Text	Sort	Ref. Characteristic	Reference Value
1		A	Approved/Released	3		1 Rx
2		D	In Development	2		
3		R	Restricted	4		1 Rx
4		U	Under Construction	1		
5		X	Obsolete	5		

As shown in Figure 56, the list characteristic contains the following parts:

1. **List Frame Characteristic** — Comprises an array of characteristics. A header, column characteristics and list entries are assigned to the list frame characteristic.
2. **Column Characteristics** — Defines the individual column names in the list (for example, Catalog Group and Text) and are assigned to the list frame characteristic using a list number. All column characteristics of the same list characteristics have the same list number.
3. **Header** — Contains the information text of the column characteristics. This is not identical to the name of the column characteristic.
4. **List Entries** — Data values assigned to the column characteristics.

Related Topics

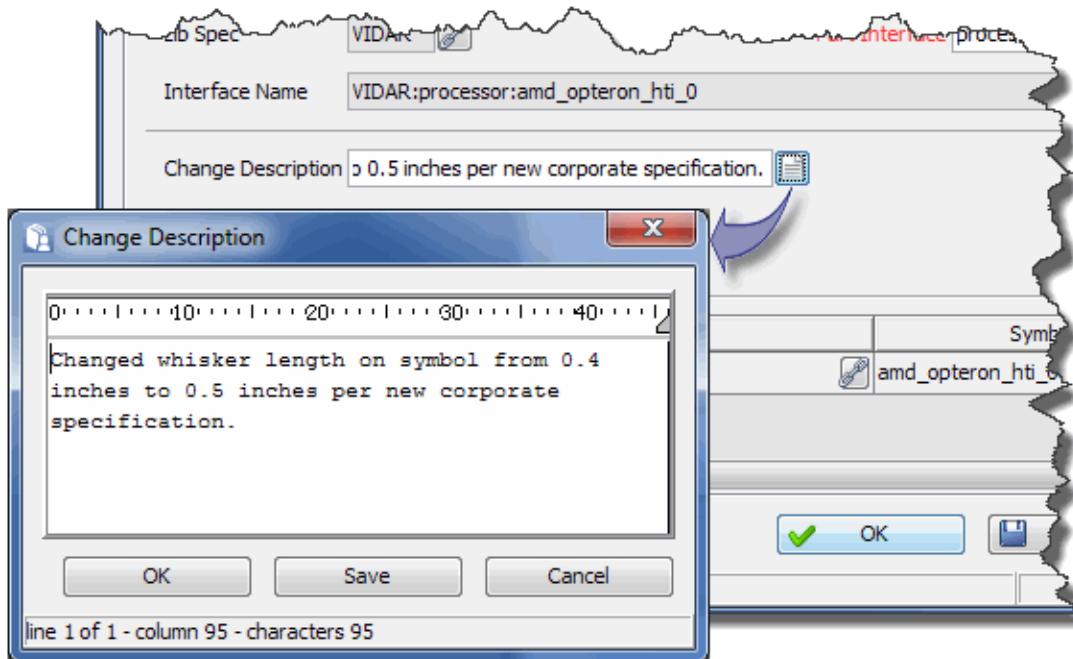
[Creating a List \(List Frame Characteristics and List Columns\)](#)

Text Frame Characteristic Type

Text frame characteristic types are for characteristics that contain long text streams as data values. The text displays in an edit window with scroll bars. A text frame characteristic can hold up to 2,000 characters.

In the user interface, clicking a dropdown button to the right of a text frame characteristic opens the text frame.

Figure 57. Example of a Text Frame Characteristic



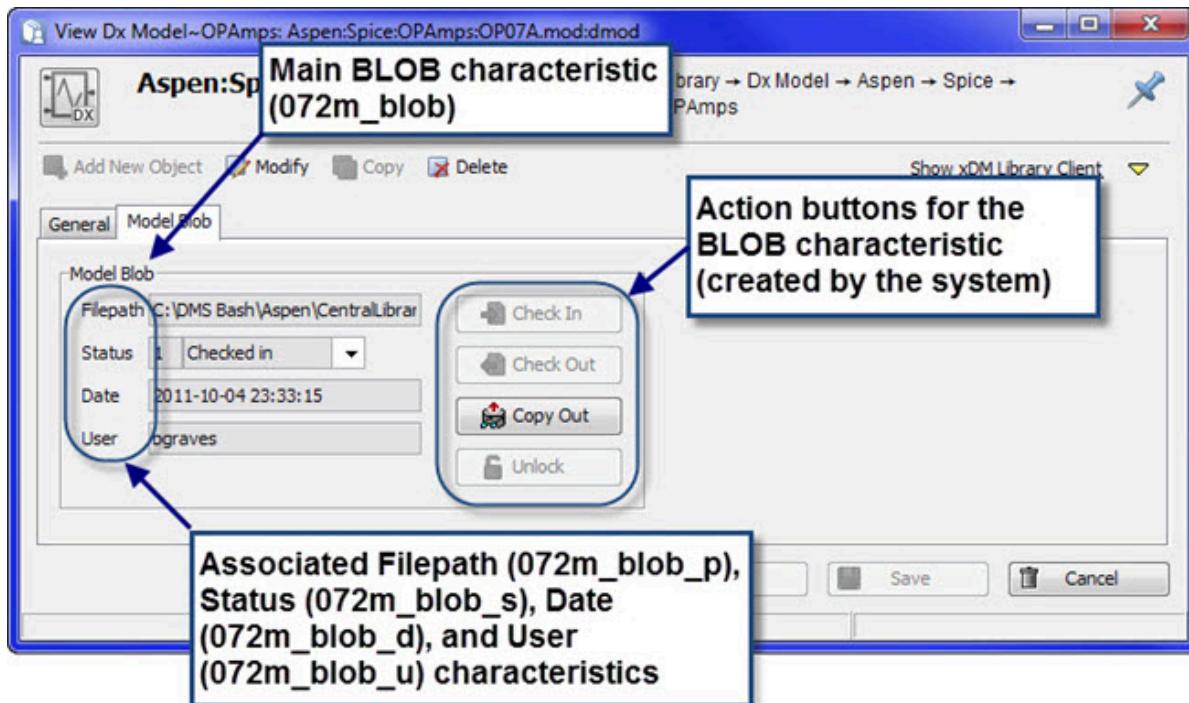
BLOB Characteristic Type

A large object that is in binary format, such as a PDF file, schematic sheet, or a graphic file can also be stored in a database table as a BLOB (Binary Large Object). A BLOB can be any single file up to a maximum size of 4 GB.

Storing files in the database as BLOBs requires a main characteristic of type 9 (BLOB) that has a characteristic name of <blob_char_name> and one or more standard characteristics associated with the main BLOB characteristic by name. At least one associated characteristic is required to define the path to the file on the file system, while other optional associated characteristics contain the status of the file (whether the file is checked in or checked out), the user who checked the file in or out, and the date of the last check-in or check-out.

[Figure 58](#) shows an example of a BLOB characteristic and other standard characteristics associated with the BLOB characteristic. The EDM Library software automatically provides **Check In**, **Check Out**, **View**, and **Unlock** buttons for each main BLOB characteristic.

Figure 58. Example of a BLOB Characteristic



Configuring a BLOB characteristic type requires the following settings:

- For the main BLOB characteristic (072m_blob in Figure 58), the characteristic type must be set to 9 (BLOB) and the value type must be set to 6 (BLOB).
- Create a file path characteristic for each BLOB characteristic. If the BLOB characteristic name is 072m_blob, then the file path characteristic must have the name 072m_blob_p.
- Create Status as an integer characteristic, which defines the status of the associated file. A value of 0 means the file is checked out, while a value of 1 means the file is checked in. When not configured, it is always possible to check a file in or out. When configured, it is only possible to check out a file when status is "checked in" and vice versa.

So, if the BLOB characteristic name is 072m_blob, you would create the status characteristic with the name 072m_blob_s.

- Create Date as a date characteristic, where the date is set to when this file was last checked in or out. For example, if the BLOB characteristic name is 072m_blob, then the date characteristic has the name 072m_blob_d.
- Create User as a char characteristic, which holds the name of the user who last checked the file in or out. So, if the BLOB characteristic name is 072m_blob, then create the user characteristic with the name 072m_blob_u.

Status Administration

The `<class_no>obj_statu` characteristic defines the current processing status of an object in a given object class. The status setting denotes the different ways in which users or programs can use the object.

Status administration is tightly coupled to [Release and Revision Control](#).

[Default Status Values](#)

[Characteristic Visibility Through the Status Value](#)

[Hierarchical Status Management Rules](#)

[User-Dependent Status Administration](#)

Default Status Values

EDM Library defines a set of default values for the object status. Every status value may not apply to or be used by every object class.

By default, the search results list displays objects with a certain status in a different color. The `<SDD_HOME>/dms/java/config/DFConnector.properties` file controls the color settings in Xpedition EDM Library Cockpit.

- **U** — The object is under construction. A librarian is currently developing the view, which is not yet finished. While under construction no one other than the librarian should access or use the object. When creating a new characteristic, setting the status value to "U" means that the characteristic is not visible in the user interface. By default, objects with status U are blue inside the search results list.
- **D** — The object is in development. While still in a development state, the view is accessible to design engineers as a non-released part. A design engineer that uses a part still in development recognizes that the part could change and require an update to the schematic. A part with a development status might not have complete data to support the entire design process, but does contain the views necessary to use the part in a schematic. By default, objects with status D are magenta inside the search results list.
- **R** — Use of the object is restricted to testing, to preliminary use, or only for use in qualified projects. By default, objects with status R are red inside the search results list.
- **A** — The object is approved and has been released. It has been created correctly and tested. The component has been proven in practice. By default, objects with status A are green inside the search results list.
- **X** — The object is obsolete. Do not use the object. By default, objects with status X are gray inside the search results list.

When making changes to the status colors that display in EDM Library Cockpit, you can either:

- Edit the `DFConnector.properties` file located in the `<SDD_HOME>/dms/java/config` directory (which affects all users of that tree).
- Copy the `DFConnector.properties` file to a different location and edit your `dmsdesktop.bat` (or similar startup file) to point to the new file location.

Related Topics

- [Characteristic Visibility Through the Status Value](#)
- [Hierarchical Status Management Rules](#)
- [User-Dependent Status Administration](#)
- [Default Status Characteristic Values \[Xpedition EDM Library Overview\]](#)

Characteristic Visibility Through the Status Value

Of the possible status values, characteristics only use the U (under development), A (approved/released), or S (system) settings. If the status value of a characteristic definition is U, the characteristic is not visible in the user interface. If the status value is A or S, the characteristic is visible in the user interface.

[Figure 59](#) and [Figure 60](#) show an example of how the Status value can control visibility of the Glossary characteristic in the Component search window.

When changing the value of the status value in a characteristic definition, use the **File > Refresh Data Model** or **Tools > Administration > Reload Data Model** pulldown menu item in EDM Library Cockpit to see the effect.

Figure 59. Glossary Characteristic Is Visible

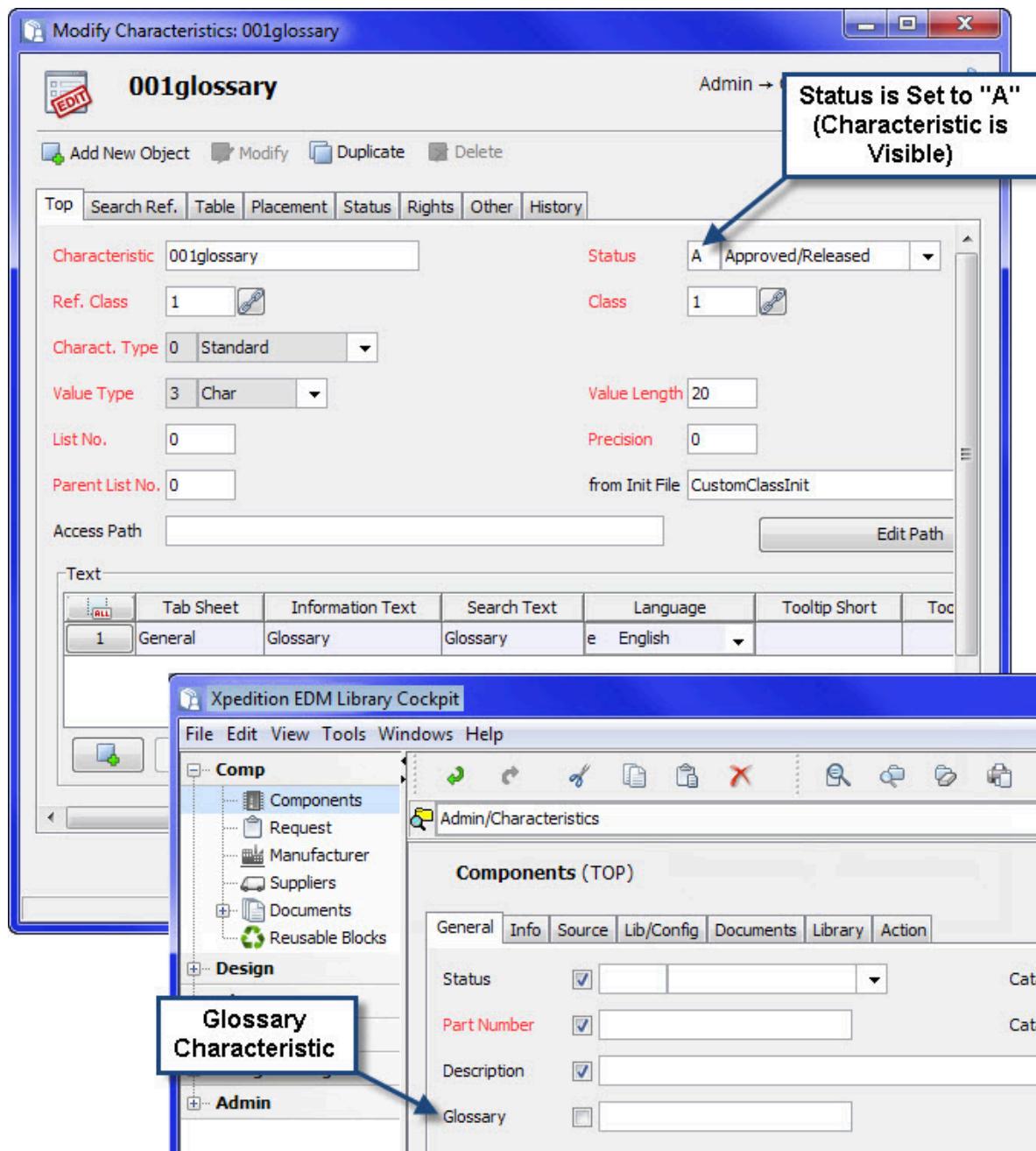
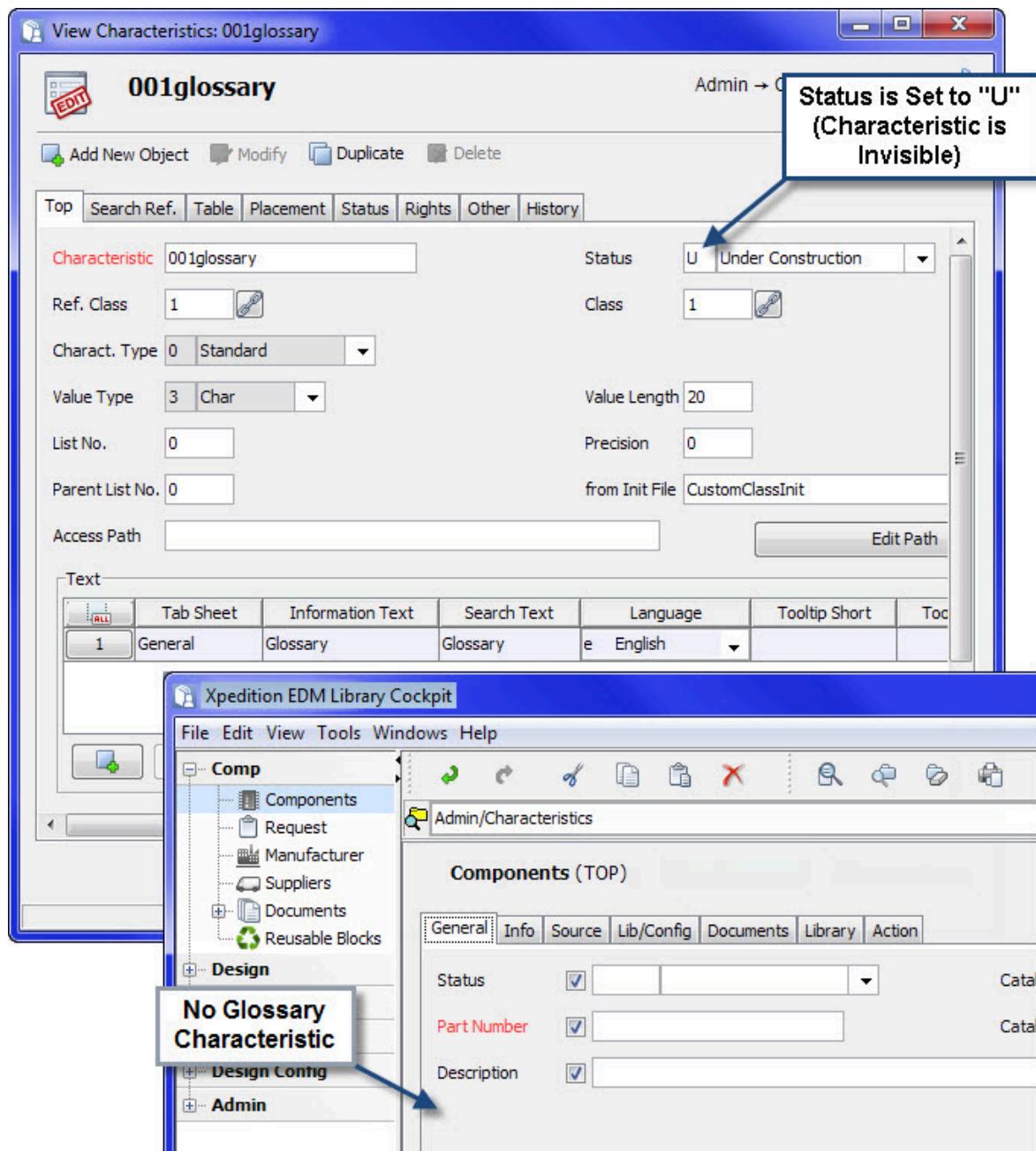


Figure 60. Glossary Characteristic Is Invisible



Related Topics

[Default Status Values](#)

[Hierarchical Status Management Rules](#)

[User-Dependent Status Administration](#)

Hierarchical Status Management Rules

Hierarchical status management rules are most often used for objects representing EDA data.

The following rules apply to status management:

- **Rule 1.** Status A has priority over status R; status R has priority over status D; status D has priority over status U, and status U has priority over status X ([Figure 61](#)).

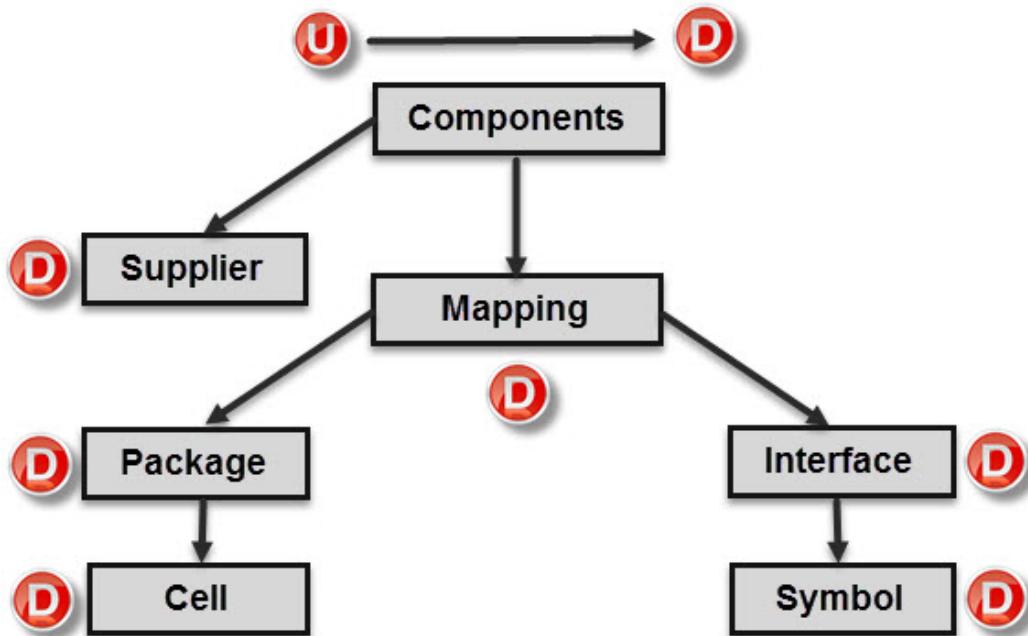
Figure 61. Status Hierarchy



- **Rule 2.** An object can only be set to a higher priority if all reference objects in lower-level object classes (see hierarchy) have been set to that priority. For example, before a component can be set to a status value of R, all referenced objects must also have been set to R.

As an example, a librarian is ready to upgrade the status of a Component object from U to D, as shown in [Figure 62](#). This is only possible if all lower-level reference objects (Mapping, Supplier, Package, Interface, Cell, and Symbol) are also set to D.

Figure 62. Status Management Example



Related Topics

[Default Status Values](#)

[Characteristic Visibility Through the Status Value](#)

[User-Dependent Status Administration](#)

[Status Management Rules \[Xpedition EDM Library Overview\]](#)

User-Dependent Status Administration

EDM Library enables you to grant edit rights to the status characteristic on a per user basis.

The **Rights** tab of the characteristic definition defines individual edit rights. Granting individual edit rights to the status characteristic determines which users can modify the status of objects in that object class.

Implementing user-dependent status administration requires meeting the following preconditions:

- A <class_no>obj_statu characteristic must exist in the associated object class.
- The Release Status status bit must be enabled in the **Status** tab of the class and characteristic.
- The Edit Allowed status bit must not be activated in the tab of the <class_no>obj_statu characteristic. If the Edit Allowed status bit is activated, all users can edit objects, irrespective of the setting on the **Rights** tab.

The following example ([Figure 63](#)) shows user-dependent status administration in greater detail. The example uses the following assumptions:

- A company has six EDM Library users and one administrator.
- The administrator controls access permissions for users and has unrestricted access to all data. The EDM administrator checks library interfaces and is the only user authorized to create the mapping.
- Users 1 and 2 are librarians that manage the logical (user 1) and physical (user 2) representation of components.
- User 3 is a procurement manager that controls the release of all data relating to suppliers.

- Users 4, 5, and 6 are design engineers. They create the components required for their designs and include them in their part lists.

Figure 63. Example of User-Dependent Status Management

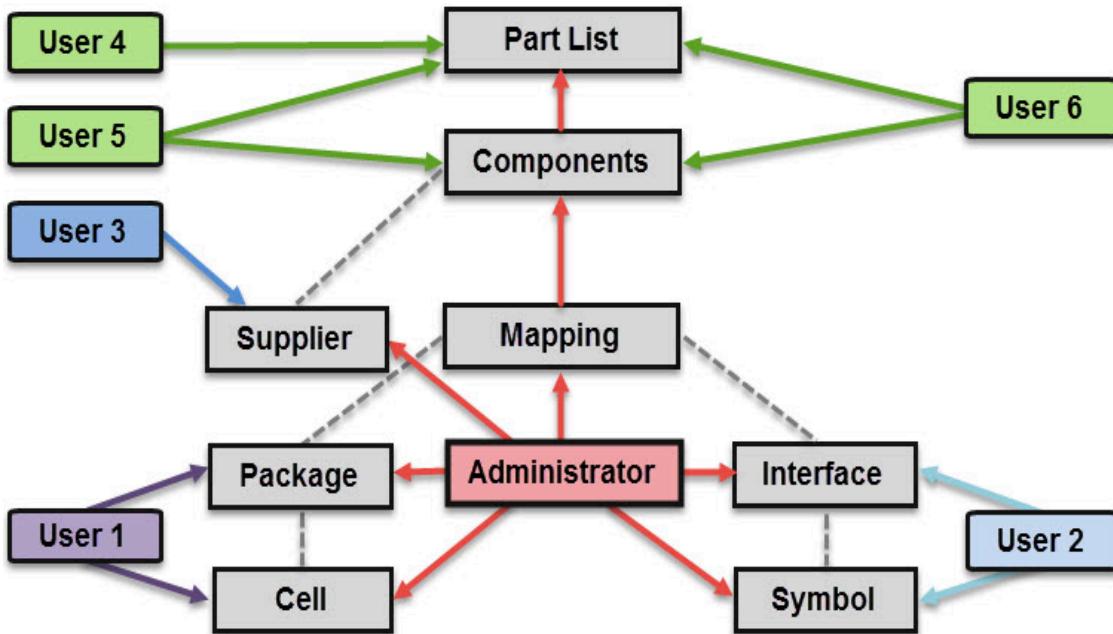


Table 10 shows the EDIT permission settings on the **Rights** tab of the <class_no>obj_statu characteristic definitions necessary to implement the status management shown in [Figure 63](#).

Table 10. Entries in the Rights Tab of the Status Characteristics

Object Class	Characteristic	User
Part List	009obj_statu	User 4
		User 5
		User 6
		Administrator
Components	001obj_statu	User 5
		User 6
		Administrator
Supplier	090obj_statu	User 3
		Administrator
Mapping	010obj_statu	Administrator

Table 10. Entries in the Rights Tab of the Status Characteristics (continued)

Object Class	Characteristic	User
Package	003obj_statu	User 1
		Administrator
Geometry	004obj_statu	User 1
		Administrator
Interface	011obj_statu	User 2
		Administrator
Symbol	002obj_statu	User 2
		Administrator

Related Topics

[Characteristic Object - Rights Tab](#)

[Default Status Values](#)

[Characteristic Visibility Through the Status Value](#)

[Hierarchical Status Management Rules](#)

Release and Revision Control

The `<class_no>obj_statu` characteristic is important when implementing status management rules for an object. Release and revision control functionality, if implemented, goes beyond the status characteristic to provide capabilities to manage the versions and release state of database objects.

The following is a summary of release and revision control capabilities:

- Total reproduction of any old version of an object
A new version of an object replicates a valid predecessor version of the object. Every new version is a full object.
- Method where a set of process steps describes a one-way or step-by-step workflow
Optionally, each process step can be separated into a second level of process steps. Every version must explicitly pass every defined process step to reach the highest step.
- Automatic copying or deletion of aggregated objects along with the container object

[Versioning Types](#)

[Characteristics Required for Release and Revision Control](#)

[Object Key Values For Different Version Types](#)

[Versioned Objects in a Process Workflow](#)

[Aggregated Object Handling](#)

[Key Reference Attributes](#)

Versioning Types

EDM Library implements versioning on a class-by-class basis. Each class can store objects using either no versioning, standard versioning, or major/minor versioning. **Release** and **Revision** items are present in the search results list popup menu if the object class uses versioning.

The following summarizes the different types of versioning schemes:

- **No Versioning** — Does not attach a version number to the key identifier of an object. A user can use the release capability to release objects to the next status level but, because objects are not subject to versioning, there is no revision control.

Most objects classes in the data model do not use versioning. When the primary way to create EDM Library objects is by importing non-versioned file system objects (such as central library symbols and cells), then the EDM Library object class does not usually have versioning.

- **Standard Versioning** — Permits the release of a hierarchy of objects, release of an individual object, and the ability to revision an object using the **Release Hierarchy**, **Release**, and **Revision** popup menu items.
 - **Release** — Moves an object to the next process step.
 - **Revision** — Creates a new version of that object (that is, a copy of the current object with an incremental version designator).

- **Major/Minor Versioning** — Adds the ability to split an object version into a major object version and a minor object version. With major/minor versioning, EDM Library provides the capability to prune, or delete, minor versions of an object when they are no longer needed.

Implementing major/minor versioning requires initializing an object class for major/minor versioning and configuring the object class to permit the release of an individual object as described in “[Object Classes Object - Status Tab](#)” on page 52. In addition to the popup menu items used for standard versioning, major/minor versioning adds the **Revision Minor**, **Revision Major**, and **Prune** subitems below the **Revision** popup menu item.

The default data model implements major/minor versioning for objects in the Documents, Symbol, and Interface classes. It is possible to extend major/minor versioning to other object classes.

Related Topics

[Release and Revision Popup Menu Items \[Xpedition EDM Library Overview\]](#)

[Releasing an Object \[Xpedition EDM Library Overview\]](#)

[Revisioning an Object \(Standard Versioning\) \[Xpedition EDM Library Overview\]](#)

[Revisioning an Object \(Major/Minor Versioning\) \[Xpedition EDM Library Overview\]](#)

[Pruning an Object \(Major/Minor Versioning\) \[Xpedition EDM Library Overview\]](#)

[Characteristics Required for Release and Revision Control](#)

Characteristics Required for Release and Revision Control

Implementing release and revision control requires that certain characteristics be present in the object class. Activating the status bits associated with major/minor versioning on the **Status** tab of a class object automatically creates many of the required characteristics

Table 11. Characteristics Required by Release and Revision Control

Characteristic	Definition
Both standard and major/minor versioning	
<class_no>obj_id	A key value for the object. With standard versioning, the key value consists of the object name (<class_no>snr), and the object's version (<class_no>intvers), separated by a colon. With major/minor versioning, the key value consists of the object name (<class_no>snr), and the object's major version (<Class-No>intvers), and minor version (<class_no>iminvers).
<class_no>vers	The version of the object in either character or integer format. With standard versioning, <class_no>vers represents the only version number associated with the object. With major/minor versioning, <class_no>vers becomes the major version.
<class_no>intvers	The major version integer (for system-internal use only).
<class_no>snr	The name of the object, labeled part number by default.

Table 11. Characteristics Required by Release and Revision Control (continued)

Characteristic	Definition
<class_no>proc	The process step, first level.
<class_no>aktobj	A help characteristic that describes whether an object is valid or invalid.
Standard versioning only	
<class_no>st	The process step, second level.
<class_no>seq	A sequence designation (that is, a counter of object modifications).
<class_no>adat	A start date for the object.
<class_no>edat	An end date for the object (if this characteristic contains a value, the version is invalid).
<class_no>ctext	A characteristic to hold change description information. The user typically uses this field to enter text that describes the reason for the object change or modification.
Major/minor versioning only	
<class_no>minvers	The minor version of the object in either character or integer format.
<class_no>iminvers	The minor version integer (for system-internal use only).
<class_no>freeze	Whether an object is frozen for prune (0=free for prune, 1=frozen for prune). An object frozen for prune is not deleted when the user performs a prune action.
<class_no>lastvers	Identifies the latest version of an object that has been created.

These characteristics must be defined for release and revision control to work, but can be hidden from a user's view. The default data model displays many, but not all, of the release and revision control characteristics on either the **General** or **Major/Minor Version** tab.

[Figure 64](#) shows the **General** tab on a Variant BOM object in the default data model. [Figure 65](#) shows the **Major/Minor Version** tab on a Symbol information window in the default data model. The windows are in edit mode, but notice how the system sets values for the version characteristics.

Figure 64. Variant BOM Information Window (Standard Versioning)

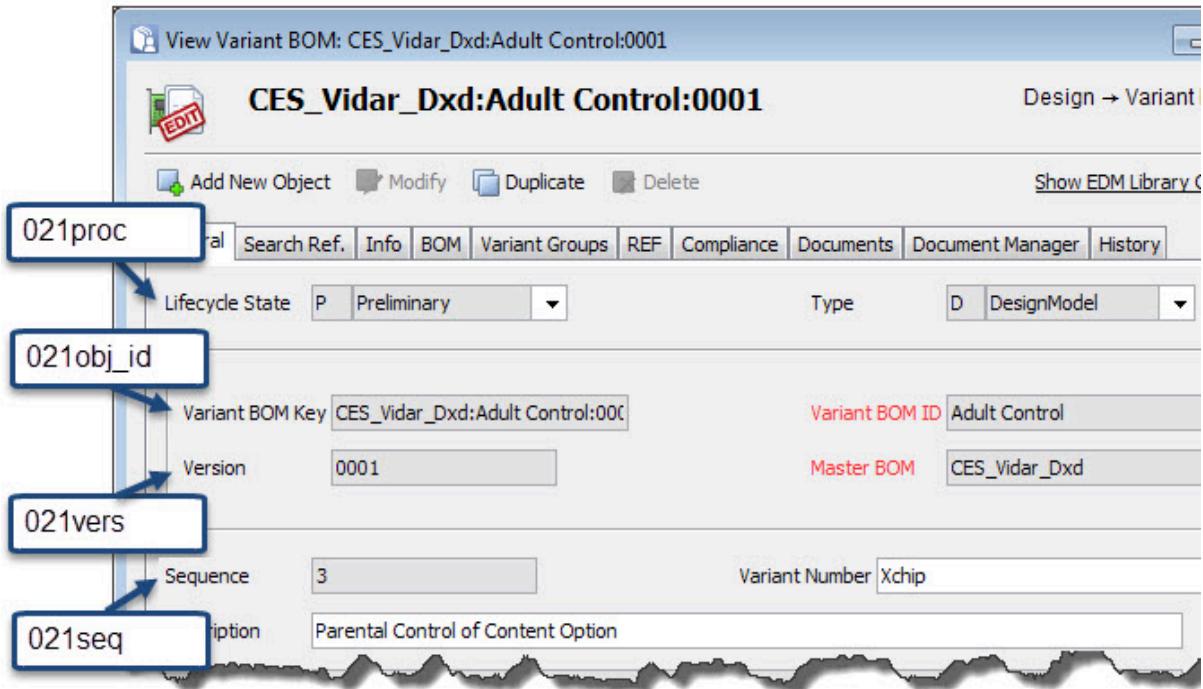
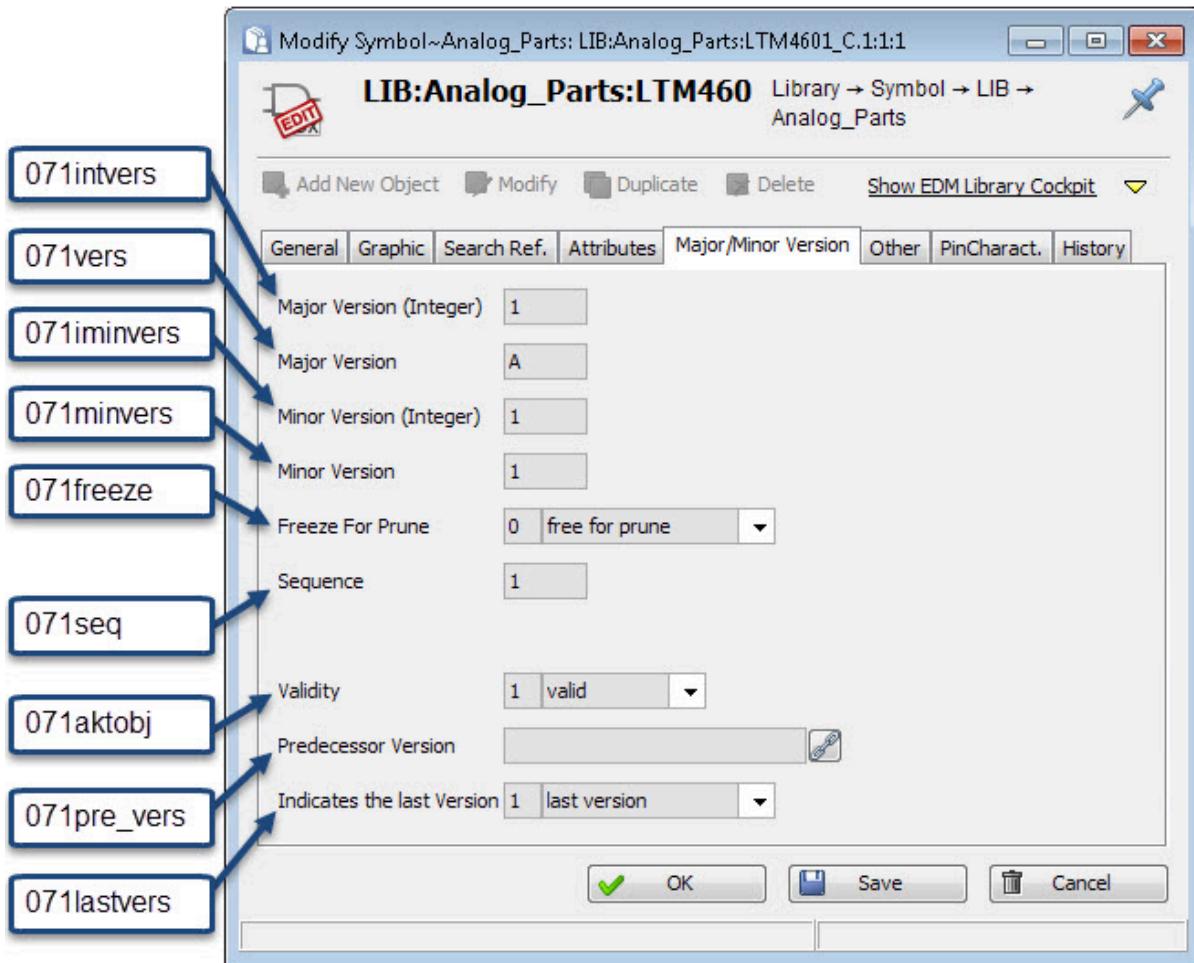


Figure 65. Symbol Information Window (Major/Minor Versioning)



Related Topics

[Object Key Values For Different Version Types](#)

[Characteristic Object - Status Tab](#)

Object Key Values For Different Version Types

The object key provides a unique identifier for each object. The value of the key used to identify a specific object depends on the versioning type being used by the class.

Table 12. Object Key Values for Different Versioning Types

Versioning	Key Value
No Versioning	You do not need a <class_no>snr characteristic and the <class_no>obj_id value is also the unique key. The user creating the object usually enters the value for <class_no>obj_id.

Table 12. Object Key Values for Different Versioning Types (continued)

Versioning	Key Value
Standard Versioning	<p>The key value should be a concatenation of the object name (<code><class_no>snr</code>) and the object version (<code><class_no>intvers</code>). For example:</p> <pre>obj1:0001</pre> <p>A user enters the value for the <code><class_no>snr</code> characteristic (obj1 in the above example), with the system automatically managing the value for <code><class_no>intvers</code> and the resulting <code><class_no>obj_id</code> value.</p>
Major/Minor Versioning	<p>The key value should be a concatenation of the object name (<code><class_no>snr</code>), the major version (<code><class_no>intvers</code>), and the minor version (<code><class_no>iminvers</code>). For example:</p> <pre>obj1:01:03</pre> <p>A user enters the value for the <code><class_no>snr</code> characteristic (obj1 in the above example), with EDM Library managing the major and minor version numbers and the resulting <code><class_no>obj_id</code> value.</p>

Versions and object names must adhere to these rules:

- The object name is comparable to the key of an object without version control. The object name is analogous to a container identifier for all object versions.
- The version designator (character or integer) must represent a self-contained object.

Figure 66 show the relationship between containers and object keys with standard versioning. The figure shows two containers, named obj1 and obj2. Each container has several versions of each object, where a key value in the form object:version denotes each object. Figure 67 shows the same objects with major/minor versioning.

Figure 66. Containers and Objects (Standard Versioning)

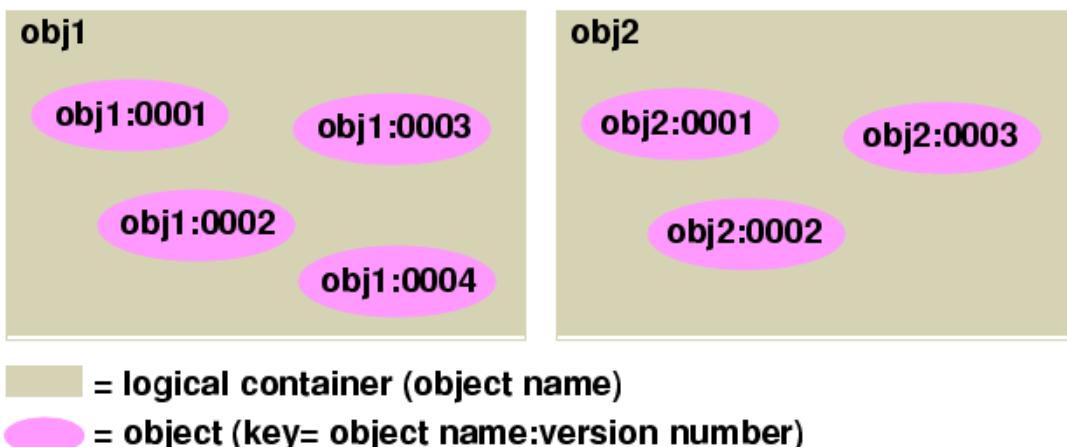
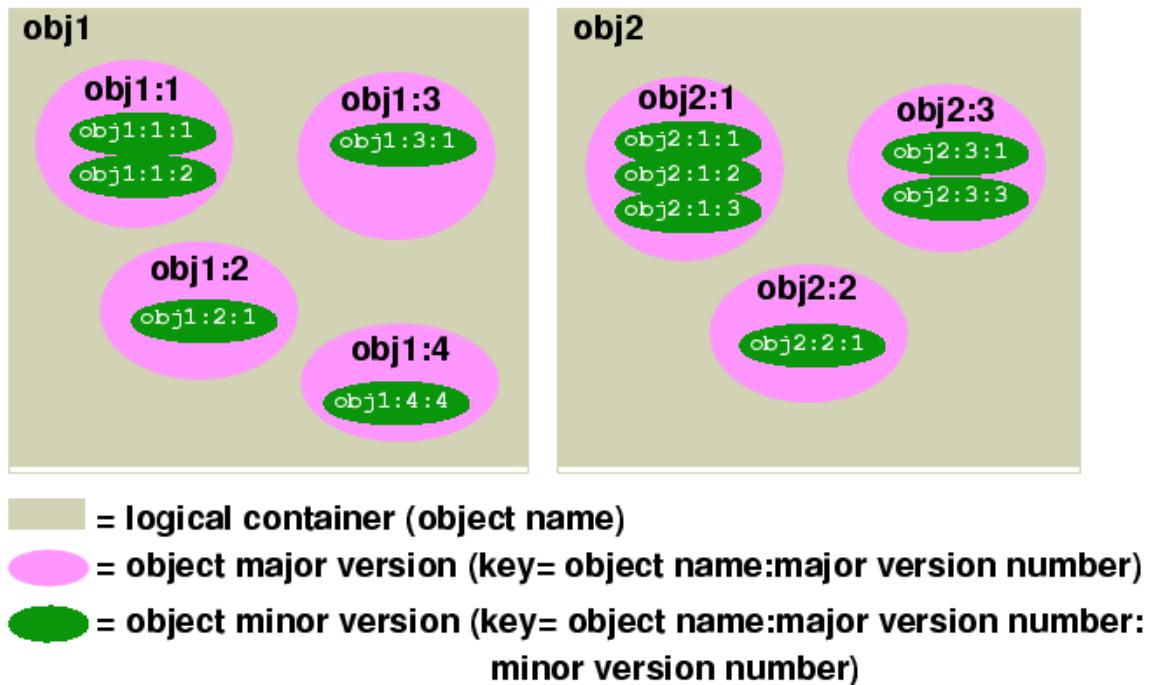


Figure 67. Containers and Objects (Major/Minor Versioning)



Versioned Objects in a Process Workflow

A process workflow contains a series of process steps, where every process step can contain only one valid version of an object.

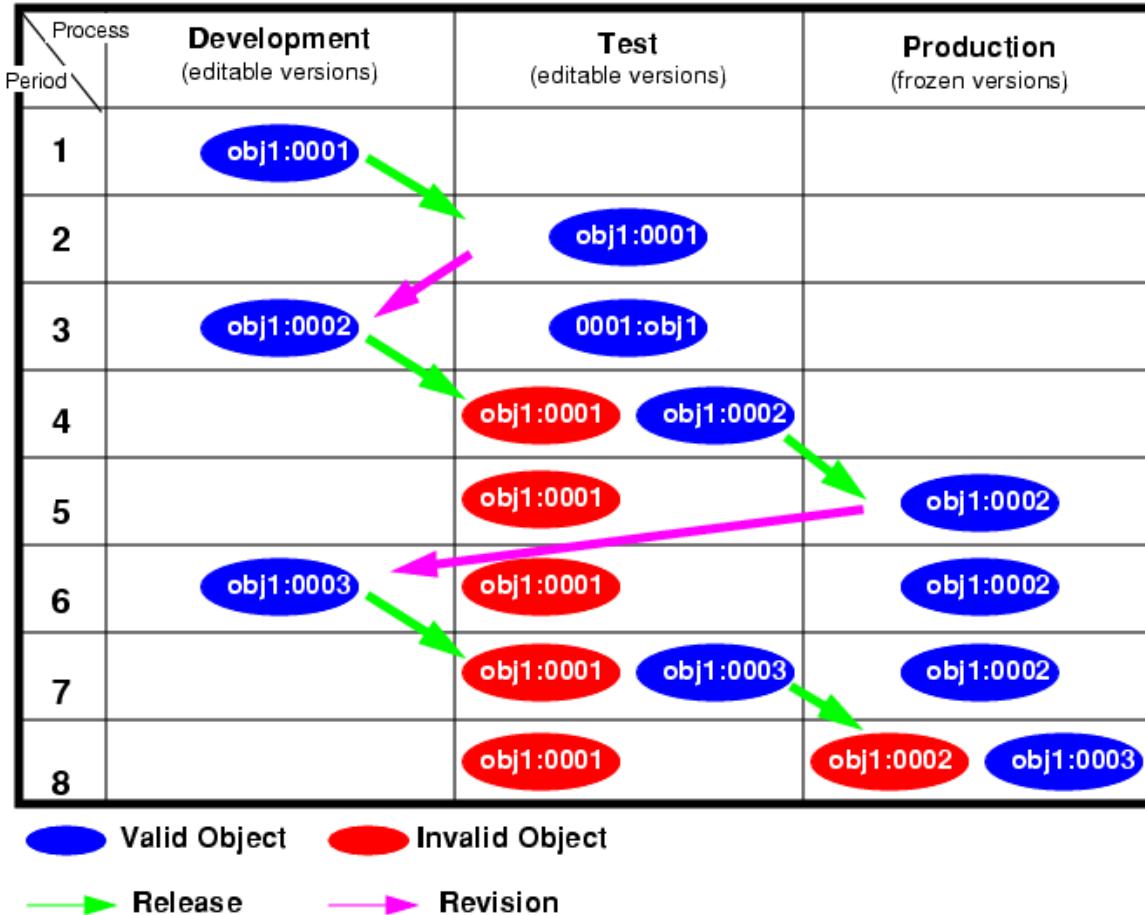
Versioned objects moving through a process workflow must follow these rules:

- Every version must pass every defined process step to reach the highest step.
- A version of an object can only be moved (that is, released) to the next process step and cannot skip process steps.
- A copy is required to move a version to a lower process step.
- The start and end date define the validity of a version.
- If the end date is set, the version is invalid.
- If a versioned object is moved to a process step that already contains a version of the same object, the object that was moved last is considered the valid version.
- Only valid versions of an object can be released or copied to a new version.

- Only the latest version of an object and objects from the latest process step can be copied to a new version.
- Objects at the highest process level are frozen. Changing an object at the highest process level requires creating a new revision copy of the object.

Figure 68 shows how an object might move through a one way, step-by-step workflow.

Figure 68. An Example Workflow (Standard Versioning)



Related Topics

[Release and Revision Popup Menu Items \[Xpedition EDM Library Overview\]](#)

[Releasing an Object \[Xpedition EDM Library Overview\]](#)

[Revisioning an Object \(Standard Versioning\) \[Xpedition EDM Library Overview\]](#)

[Revisioning an Object \(Major/Minor Versioning\) \[Xpedition EDM Library Overview\]](#)

[Pruning an Object \(Major/Minor Versioning\) \[Xpedition EDM Library Overview\]](#)

Aggregated Object Handling

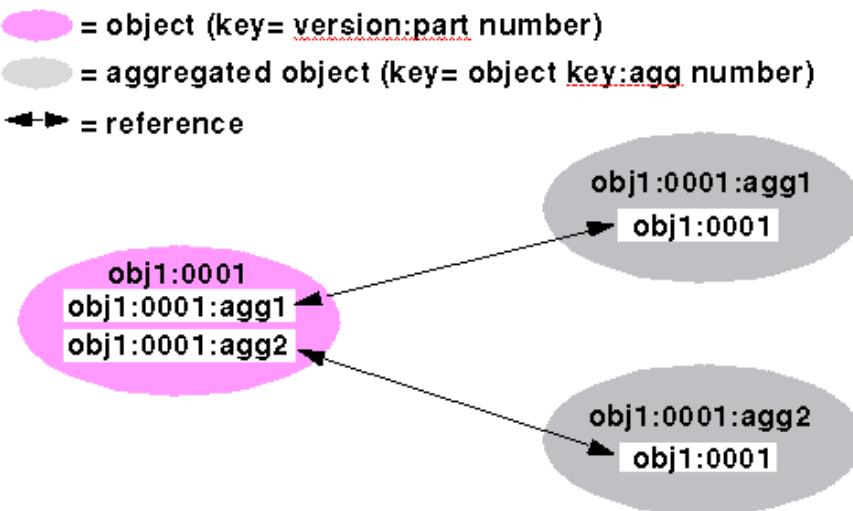
Aggregated objects are copied and deleted automatically with their container object.

Aggregated objects have the following requirements:

- Every aggregated object needs a reference to its container object.
- The container object must have a list of references to its aggregated object.
- The key of the aggregated object must contain the key of the container object.

Figure 69 shows a diagram of how a container object references aggregated objects and how aggregated objects reference container objects.

Figure 69. Aggregated Object Handling



Key Reference Attributes

EDM Library does not support part number references, but does support key references.

Key references have these attributes:

- The reference to a version object is the version itself, not the container (part number).
- The object can only view the version.
- EDM Library does not provide an automatic update of references to versioned objects after a revision or release.

Using OK macros, right-clicking, or any other user action can update the reference.

Reference Characteristics

Reference characteristics take over their value by referencing the data stored in another characteristic in a different object class.

When creating a reference characteristic, you define the reference class on the **Top** tab of the characteristic definition and select one of the following three reference types on the **Status** tab:

- [Object Reference](#)
- [Multiclass Object Reference](#)
- [Common Object Reference](#)

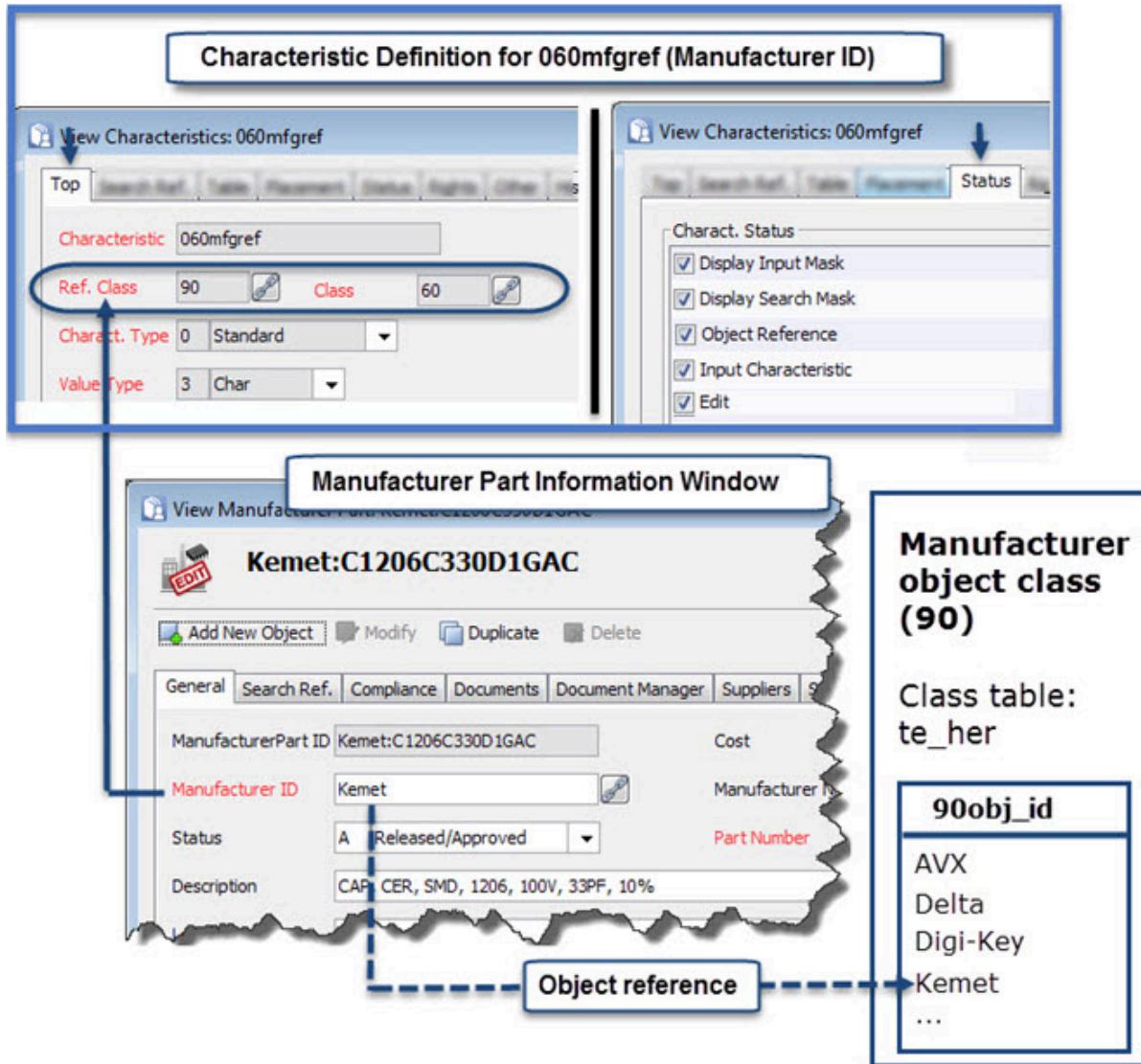
The following subsections provide examples of each reference type.

Object Reference

A reference characteristic configured with a basic object reference takes the value from a characteristic in one other object class. From a data model perspective, the basic object reference creates a one-to-one relationship.

In [Figure 70](#) the Manufacturer ID characteristic on the **General** tab of a Manufacturer Part object information window (class number 60) is defined as an object reference to the `obj_id` inside the Manufacturer class (class number 90).

Figure 70. Object Reference Overview



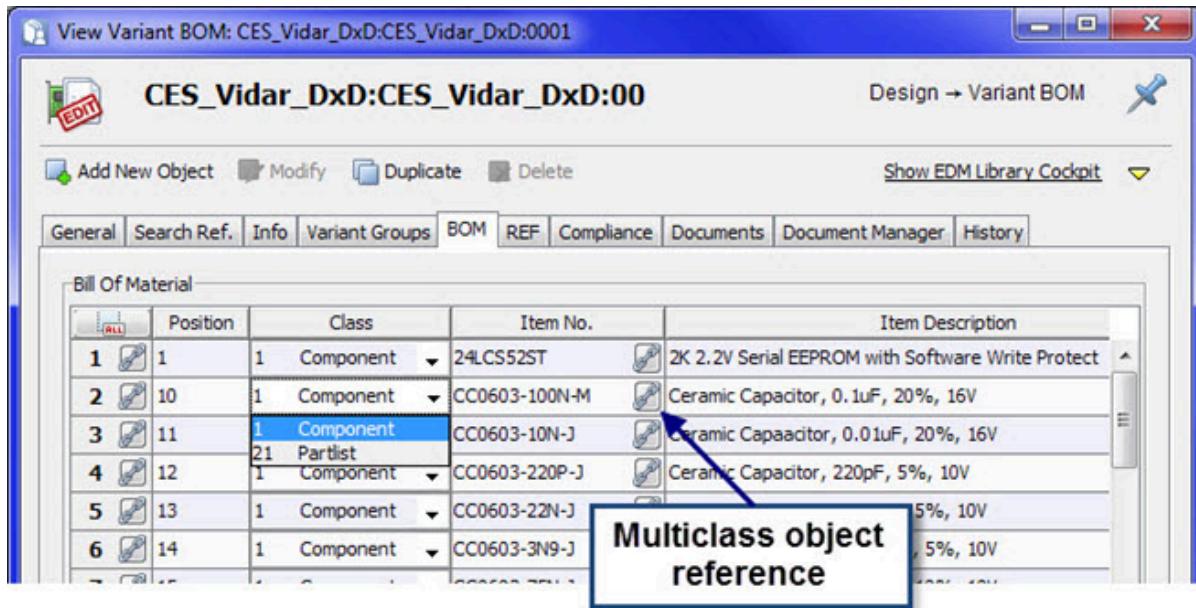
Defining the 060mfgref (Manufacturer ID) characteristic as an object reference has two effects:

- Enables data value checking. In this example, if the user types a value for Manufacturer ID and then saves the Manufacturer Part object, the software checks if the value exists in the Manufacturer object class.
- Creates a reference button next to the input field that enables a user to enter a value by using “Send To” functionality. A user can click the reference button to open a search window of the referenced class, perform a search, and double-click a row in the search results list or use the **Send to** popup menu item to return the value to the field.

Multiclass Object Reference

A characteristic defined with a multiclass object reference establishes a relationship where the characteristic can have a reference to more than one object class. From a data model perspective, the multiclass object reference creates a one-to-many relationship

Figure 71. Multiclass Object Reference Example



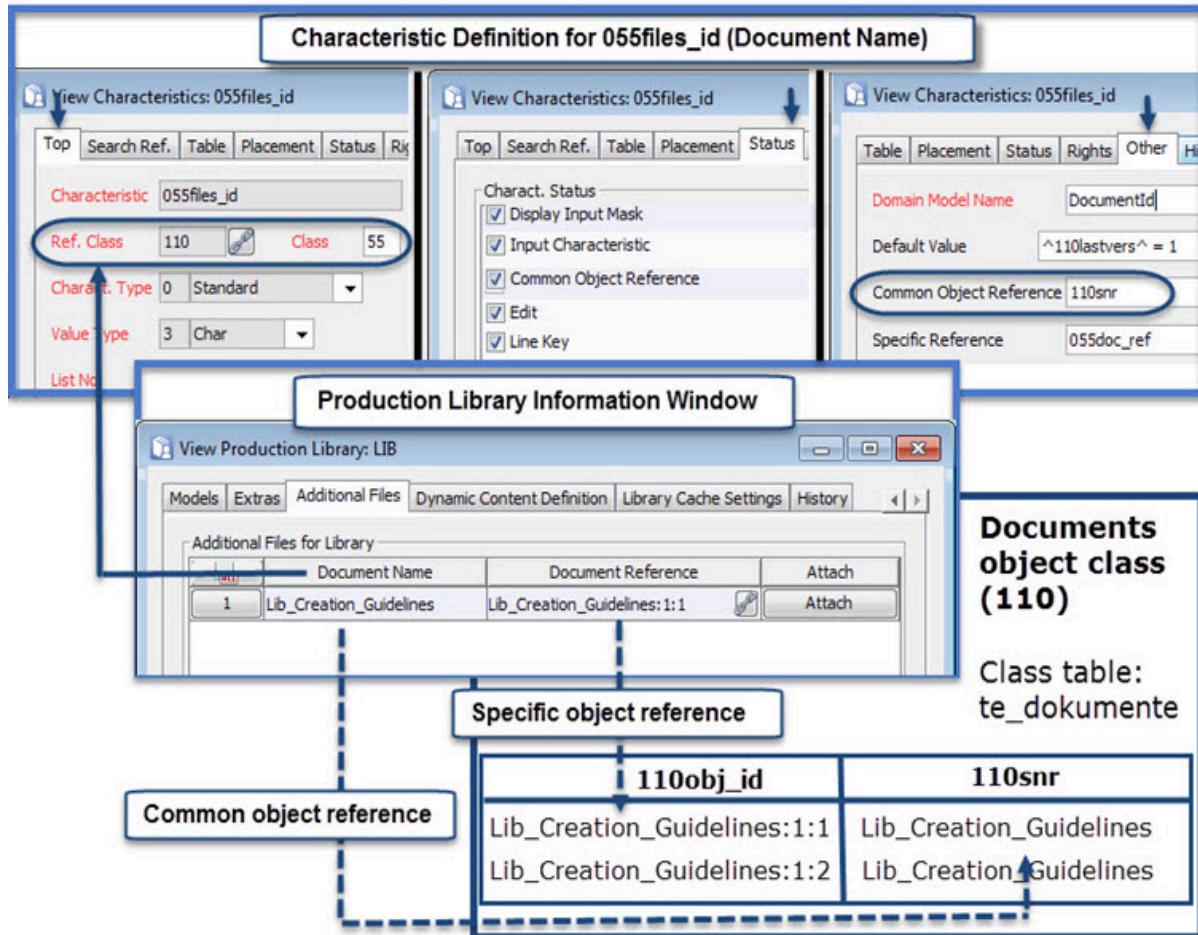
A multi-class object reference requires two characteristics: one that has the Multiclass Object Reference status bit set, and another characteristic that defines the target object class number. The Default Value field of the **Other** tab of the characteristic that has the Multiclass Object Reference status bit set defines the name of the characteristic containing the target object class number. The characteristic with the target object class number must also adhere to the following prerequisites:

- The characteristic must have a value type of Integer with a value length of at least 3.
- The **Top** tab of the characteristic must define an option list and contain the class numbers of all possible referenced object classes.
- The Option List and Mandatory Field (check always) status bits must be checked in the **Status** tab.

Common Object Reference

Different from an object reference or multiclass object reference characteristic, a common reference characteristic does not point to the main key of the versioned target object class but instead can point to any characteristic in the class. In [Figure 72](#), the Document Name characteristic is defined as a common object reference to the 110snr characteristic inside the Documents class. From a data model perspective, the common object reference creates a many-to-many relationship

Figure 72. Common Object Reference Overview



You can set up a common object reference characteristic differently, depending if you are setting up an input characteristic or a characteristic to display data.

Common Object Reference Input Characteristics

Common object reference input characteristics can either be configured as standard characteristics within the main class table of an object class (n:1 relation) or inside a list frame characteristic (n:m relation).

As an example, the Document Name input characteristic (055files_id) defines the reference and displays as a column in the Additional Files for Library list frame on the **Additional Files** tab of a Document object (Figure 72 on page 141). The definition for the Document Name characteristic has:

- **Top tab** — A Class value of 55 (Production Library class) and a Ref. Class value of 110 (Documents class).
- **Status tab** — Bits Display Input Mask, Input Characteristic, Edit, Line Key and Common Object Reference are active.
- **Other tab** — The Common Object Reference field defines the target characteristic to reference as 110snr.

In addition to the common object reference characteristic, a required specific object reference input characteristic pointing to the obj_id of the target class must exist in the list table. In the Additional Files for Library list box in [Figure 72](#) on page 141, this is the Document Reference column (055doc_ref).

Type the name of this specific object reference characteristic in the Specific Object Reference input field inside the **Other** tab of the common object reference characteristic.

Common Object Reference Display Characteristics

The display characteristics of a common object reference are typically configured as list frame columns. Different options can be configured to view either all versions of the referenced objects or a subset of referenced objects, such as the approved objects or the objects with the latest version.

Making use of these options requires at least two display characteristics, as follows:

- A common object reference display characteristic that points to the same column as the common object reference input characteristic and has the same settings as the common object reference input characteristic in the **Top** tab, except for a different list number. The **Status** tab must have different settings. Activate only the Display Input Mask, Display Search Mask, Search Characteristic, Outer Join and Common Object Reference status bits.

The **Other** tab of the common object reference display characteristic also contains the same values as the **Other** tab of the common object reference input characteristic. In addition, restrictions for viewing only a subset of versions of referenced objects can be defined in the Default Value input field inside the **Other** tab of the common object reference display characteristic. When typing into the Default Value input field, you must use the following syntax:

```
^<charactname1>^ = <value> <operator> ^<charactname2>^ = <value> ...
```

- *<charactname>* is the name of a characteristic in the target object class, including the class number prefix.
- *<value>* must be a valid data value for *<charactname>* or a set of values separated by pipe characters (|).

You cannot use wildcards for *<value>*. The pipe character to specify a set of values represents a sorted OR operator. For example, if you specify 'D|A' for *<value>*, the system searches for 'D' first and displays only objects with a 'D' status. If no objects exist with a status of 'D', the system searches and displays objects with status 'A'.

- *<operator>* can be used to combine restrictions for several characteristic values. Possible operators are '| (logical OR) and '&&' (logical AND).

For example, enter the following string to display all latest document versions with a status of 'D' or a status of 'A', if no document version in status 'D' exists.

```
^110lastvers^=1 && ^110proc^ = D|A
```

- A specific object reference display characteristic must be defined to be able to open the target object. This characteristic is configured in the same way as the specific object reference input characteristic, except it must have a different list number in the **Top** tab.

The two display characteristics mentioned above are sufficient to create the common object reference join between two classes to correctly view all or filtered target objects.

You can define additional display characteristics to view data values of the target object class.

Using Compose Mode to Edit Characteristic Definitions

The Xpedition EDM Library Cockpit Compose mode enables a user with administrative privileges to interactively add, delete, and edit characteristic placement parameters (such as the length and height of a field for input) and their location on a search criteria tabbed sheet or an object information tab sheet.

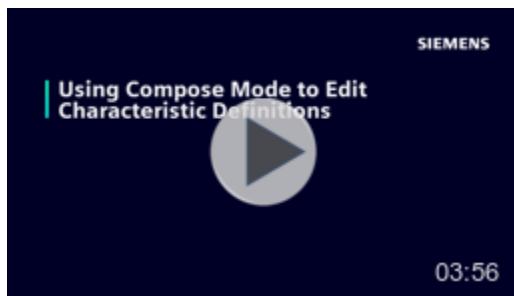
When you enter Compose mode, a grid shows over the advanced search pane, and the software automatically displays Characteristic objects that are in the search criteria pane in the search results list. You can then use the grid to graphically reposition and resize characteristics, and use the search results list to open a characteristic information window to edit characteristic property values, create a new characteristic, or delete a characteristic.

Prerequisites

- You invoked the Xpedition EDM Library Cockpit application and connected to an EDM Server.
- Your account has administrator privileges, with the rights to edit characteristic objects.

Video

- Select an object class and place the user interface into Compose mode.
- Use the Compose mode grid to reposition and resize search criteria area and information window characteristics
- Use the search results list to further modify characteristic parameters, create characteristics, or delete characteristics.
- Close Compose mode.
- Refresh the database to see the changes.



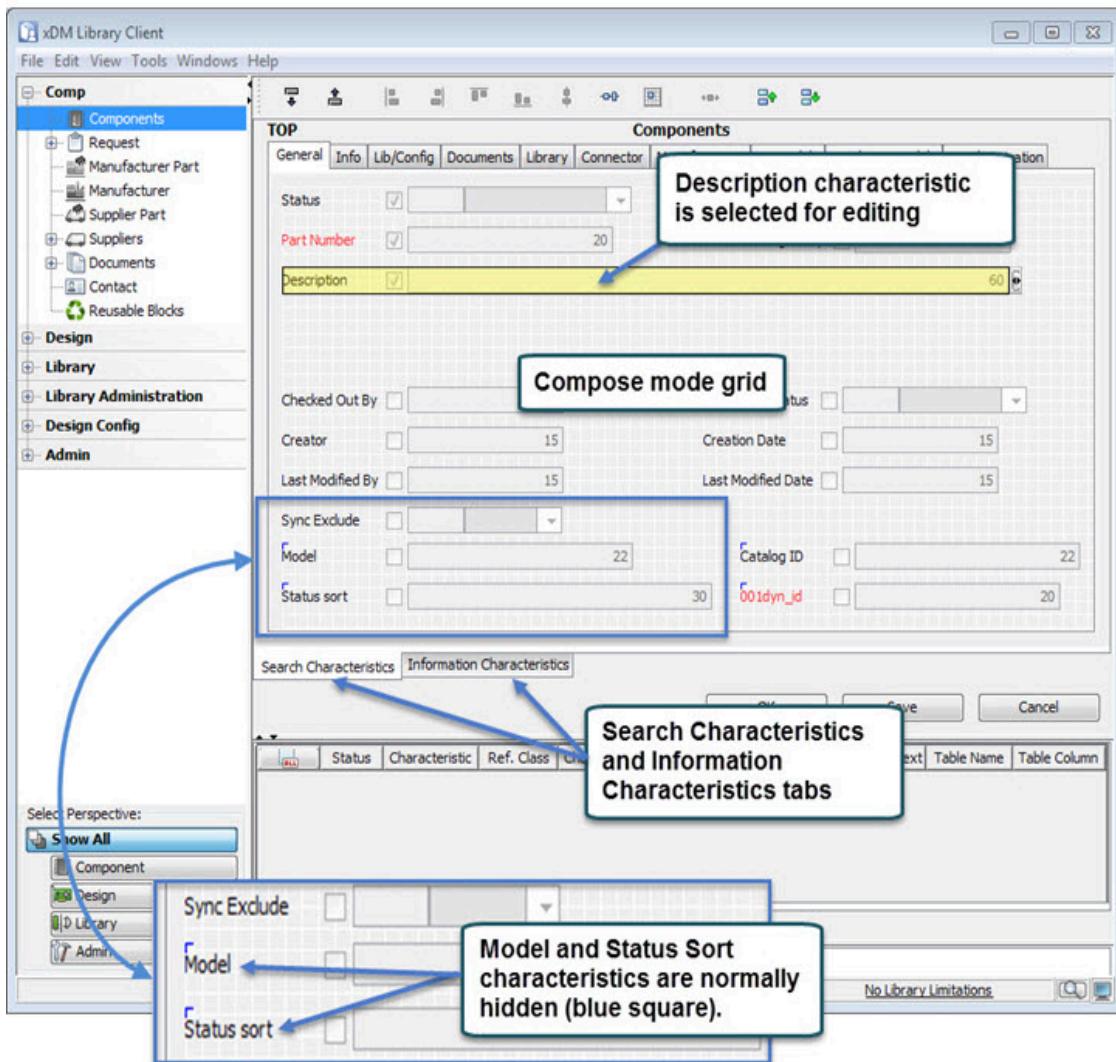
Procedure

1. Select an object class or catalog in the object classification pane.
2. Choose the **Edit > Catalog > Compose Mode** pulldown menu item or press the F12 key to place the EDM Library Cockpit into Compose mode.

A grid and two tabs display in the search criteria pane (Figure 73).

If you are logged in with an account that does not have the rights for Compose mode, the **Compose Mode** menu item and the icons on the composer toolbar are dimmed or absent.

Figure 73. Xpedition EDM Library Cockpit in Compose Mode



3. Click the **Search Characteristics** tab to change characteristics in the Search Criteria area, or click the **Information Characteristics** tab to change characteristics on an object information window.

A blue rectangle above the characteristic name identifies hidden characteristics. For example, the Model and Status Sort characteristics on the **General** tab of a Component object do not display when EDM Library Cockpit is in Restrict mode, but are available for edit in Compose mode.

4. Use the mouse and composer toolbar to change the appearance and placement of a characteristic on either tab:



Note:

Changing the characteristic placement affects all catalog groups. You cannot specify different placements for different catalog groups.

- a. From the search results list, open a Characteristic information window in Modify mode to change non-placement attributes. Save and close the characteristic information window before making grid changes.

You can also use the search results list popup menu to create new characteristic objects or select and delete existing characteristic objects.

- b. Use the mouse to move or resize the characteristic on the grid, and then click the **Save** button to save the grid changes.

If you move a characteristic in the grid and do not save changes before opening the characteristic information window, you lose any repositioning you did in the grid.

5. Click the **Save** button in the Compose mode grid to write the characteristic changes to the database and enable other users to see those changes the next time they connect to the server. To force an immediate reload of the data model cache, use the **Tools > Administration > Reload Data Model** menu item.
6. Click **OK**, choose the **Edit > Catalog > Restrict Mode** pulldown menu item, or the press F3 key to return to Restrict (normal) mode.

Related Topics

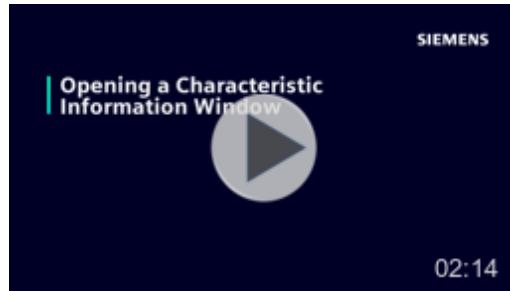
[Composer Toolbar \[Xpedition EDM Library Overview\]](#)

[Opening a Characteristic Information Window](#)

Opening a Characteristic Information Window

Opening a Characteristic information window lets an administrator change the definition parameters of an existing characteristic, or enter new values to create a new characteristic.

The following video visually shows the tasks described in this section:



[Opening a Characteristic Information Window From Another Information Window](#)

[Opening a Characteristic Information Window From the Advanced Search Pane](#)

[Opening a Characteristic Information Window by Searching the Characteristics Object Class](#)

Opening a Characteristic Information Window From Another Information Window

Click the name of a characteristic in an open information window to display the information window for that characteristic.

Restrictions and Limitations

The characteristic you click cannot be an action button because clicking a button performs the action rather than displaying the characteristic information.

Prerequisites

- You invoked the Xpedition EDM Library Cockpit application and connected to an EDM Server.
- You are using an account with administrator privileges.
- To modify a characteristic, you acquired a Librarian or Developer license.

Procedure

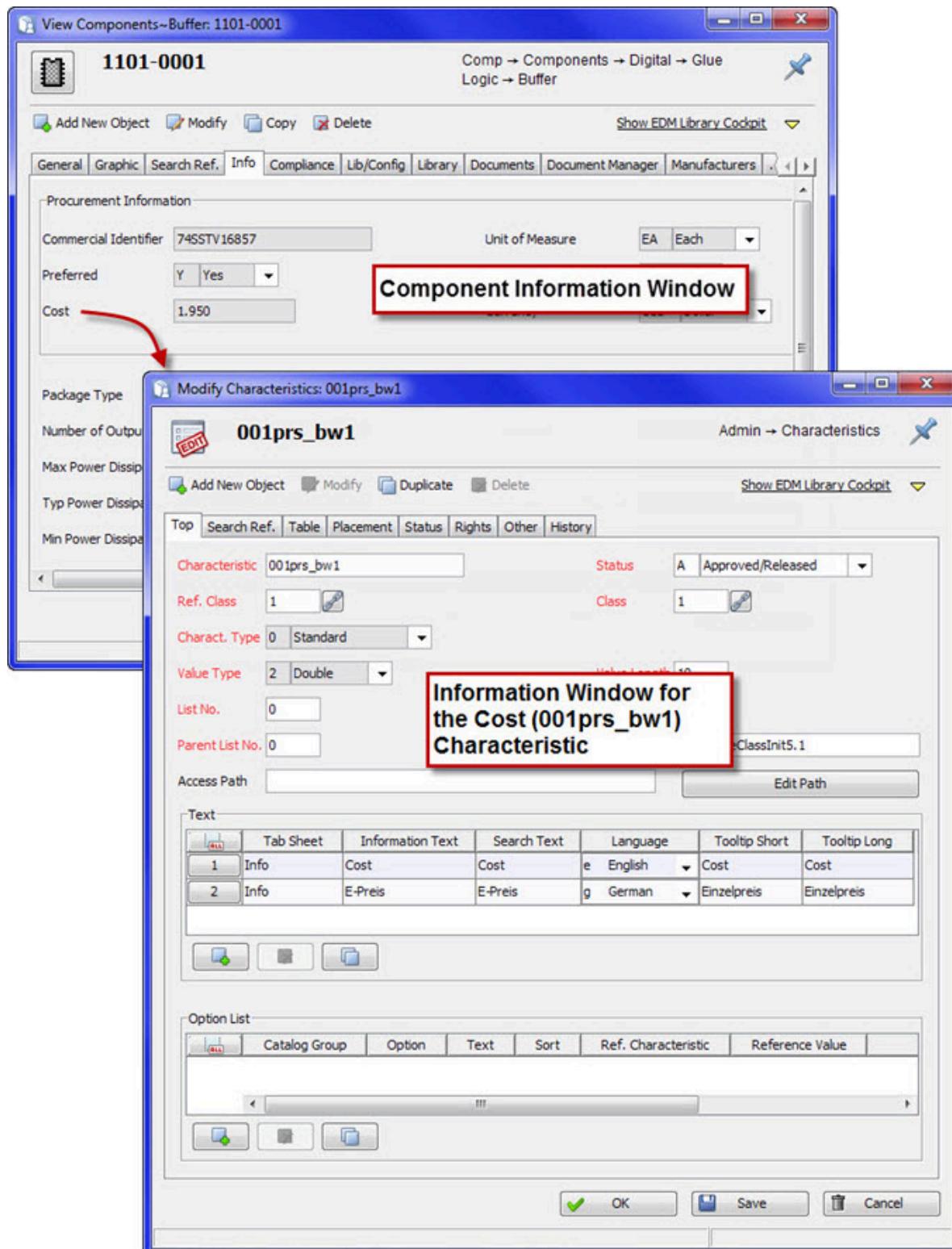
Click the name of a characteristic in an information window or, if the characteristic is a column characteristic in a list frame, right-click the column head and then choose the **Modify Characteristic** popup menu item.

Examples

Figure 74 shows an example of how clicking the Cost characteristic name in an open component information window displays the definition for the Cost (001prs_bw1) characteristic in the Characteristic

information window. An administrator can then make quick changes such as altering the number of permitted input characters or changing the position of the characteristic.

Figure 74. Opening a Characteristics Information Window From Another Information Window



Related Topics

[Characteristic Object - Top Tab](#)

[Characteristic Object - Search Ref Tab](#)

[Characteristic Object - Table Tab](#)

[Characteristic Object - Placement Tab](#)

[Characteristic Object - Status Tab](#)

[Characteristic Object - Rights Tab](#)

[Characteristic Object - Other Tab](#)

[Characteristic Object - History Tab](#)

Opening a Characteristic Information Window From the Advanced Search Pane

Click the name of a characteristic in an advanced search pane to display a window with a list of all characteristics on that search pane tab. Using the list, open the information window of a specific characteristic.

Prerequisites

- You invoked the Xpedition EDM Library Cockpit application and connected to an EDM Server.
- Your account has administrator privileges.
- To modify a characteristic, you acquired a Librarian or Developer license.

Procedure

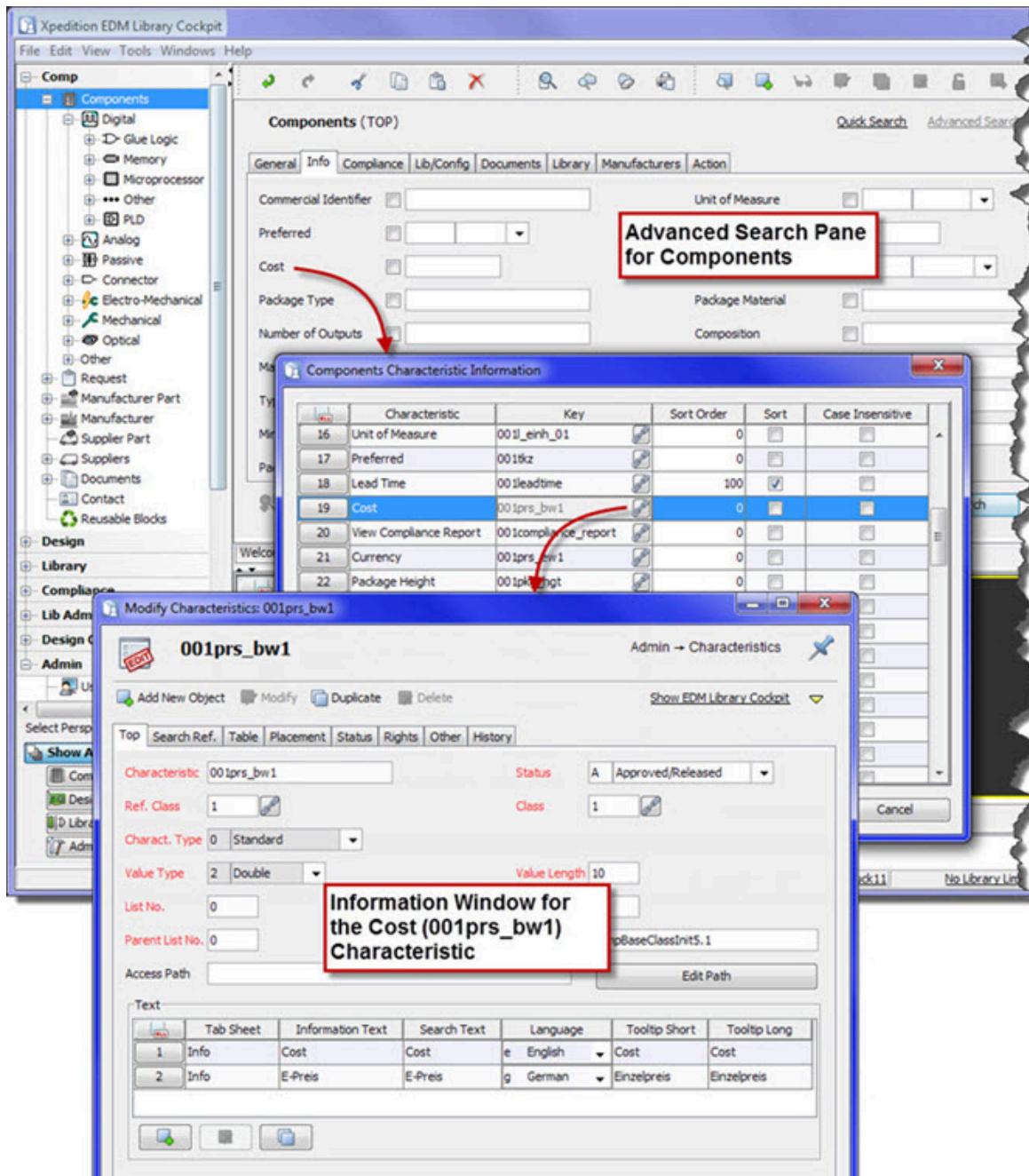
1. Select the object class from the classification hierarchy (for example, **Comp > Components**).
2. Make sure the search criteria pane is in advanced search mode and showing all searchable characteristics. A link at the top of the search criteria pane lets you switch between modes.
3. Click the tab containing the characteristic whose definition you want to display and then click the characteristic name.
A window displays listing all searchable characteristics on that tab.
4. Click the link icon next to the characteristic name.

Examples

Figure 75 shows how clicking the Cost characteristic name in the advanced search pane for components displays a summary of all characteristics on that advanced search pane tab with the characteristic you clicked highlighted. Clicking the reference button next to the characteristic key (001prs_bw1) opens the Characteristic information window.

An administrator can then make changes such as altering the number of permitted input characters, changing the position of the characteristic, and so on before saving the characteristic definition back to the database.

Figure 75. Opening a Characteristics Information Window From an Advanced Search Window



Related Topics

[Characteristic Object - Top Tab](#)

[Characteristic Object - Search Ref Tab](#)

[Characteristic Object - Table Tab](#)[Characteristic Object - Placement Tab](#)[Characteristic Object - Status Tab](#)[Characteristic Object - Rights Tab](#)[Characteristic Object - Other Tab](#)[Characteristic Object - History Tab](#)

Opening a Characteristic Information Window by Searching the Characteristics Object Class

Characteristic definitions reside in the Characteristics object class. You can use the characteristic search window to create a query that returns matching Characteristic objects to the search results list. You can then use the search results popup menu items to open a Characteristic information window.

Prerequisites

- You invoked the Xpedition EDM Library Cockpit application and connected to an EDM Server.
- Your account has administrator privileges.
- To modify a characteristic, you acquired a Librarian or Developer license.

Procedure

1. In the object classification pane, open the **Admin** module and choose **Characteristics**. Alternately, type Admin/Characteristics in the location bar.

The search criteria pane can either be in quick search mode or advanced search mode, depending on your preference settings in EDM Library Cockpit. A link at the top of the search criteria pane lets you switch between modes.

**Tip**

For general information about searching in EDM Library Cockpit and how to formulate a search query, refer to “Searching and Replacing” in the *EDM Library Overview*

2. Type a search query into the search criteria pane. Leave all search criteria fields empty to return a list of all Characteristic objects.

Advanced search mode enables you to enter search criteria into one or more input fields on multiple tabs and place a check mark next to the characteristics that you want as columns in the search results list.

3. Click **Search** to return a list of Characteristic objects that match your query.
4. Select a row from the search results list. Then, with the cursor still in the search results area, right-click and choose an editing function.

Alternately, click the reference button to the left of a row to open an information window in view mode without first having to select the row or use the popup menu.

An information window can either be floating or docked. Your EDM Library Cockpit preferences determine the default.

Related Topics

- [Characteristic Object - Top Tab](#)
- [Characteristic Object - Search Ref Tab](#)
- [Characteristic Object - Table Tab](#)
- [Characteristic Object - Placement Tab](#)
- [Characteristic Object - Status Tab](#)
- [Characteristic Object - Rights Tab](#)
- [Characteristic Object - Other Tab](#)
- [Characteristic Object - History Tab](#)

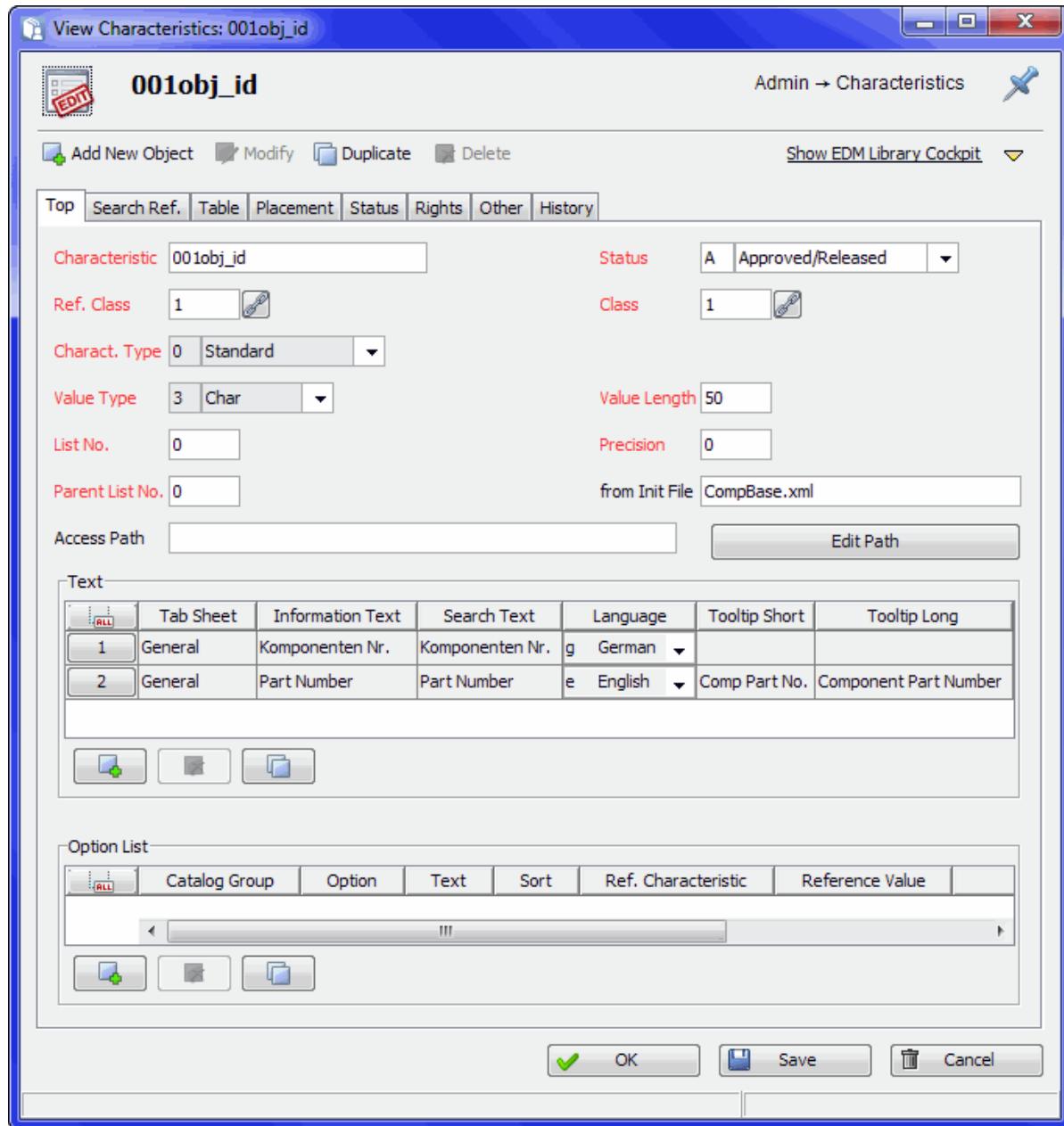
Characteristic Object - Top Tab

To access: Refer to “[Opening a Characteristic Information Window](#)” on page 146

The **Top** tab of a Characteristic information window contains the master data for the characteristic.

Description

Figure 76. Top Tab of a Characteristic



Characteristics

Characteristics on the **Top** tab can be divided into:

- Master data characteristics
- Text list
- Options list

Master Data Characteristics

- **Characteristic** — Specifies the unique identifier for the characteristic. The value consists of an abbreviated name with an associated object class key as a required prefix; for example, 001<name> for a characteristic belonging to object class 1.

You can use integers, lowercase letters, uppercase letters, and underscores in the value. If the characteristic ID contains an invalid character such as a dash or period, an error message occurs when you try to save the Characteristic information window.

The name cannot begin with the characters “smw”.



CAUTION:

Never modify or delete a Characteristic object that has a Status value of S (system). Objects with an S status value are critical to the operation of EDM Library applications. If you modify or delete such an object, it can potentially corrupt database data or render EDM Library applications inoperable.

- **Status** — Can be U (Under Construction), A (Approved/Released), or S (System). EDM Library applications do not display characteristics with a status value of “U”. Because characteristics with an **S** (system) status are important for EDM Library operation, you must never modify or delete them.
- **Ref. Class** — The number of the object class to which the characteristic belongs.

When creating a standard characteristic, enter the same value for a class number in Ref. Class as in Class (for example, [Figure 76](#) shows a class number of “1” in both Ref. Class and Class). Enter different values in Ref. Class and Class for reference characteristics. For a reference characteristic (for example, the datasheet name inside the Components object class), the Ref.Class is 113 and the Class is 1. This means that the characteristic’s value is stored in class 113 and viewed in class number 1.

- **Class** — The number of the object class where a user views the characteristic. If the Class field is empty, EDM Library does not display the characteristic in any class.
- **Charact. Type** — The characteristic type. Select one of the following values from a list:
 - 0 Standard
 - 1 Action Button
 - 2 Body Property
 - 3 Pin Property

- 4 Bit Status
- 5 List Frame
- 7 Text Frame
- 8 (Reserved for future use)
- 9 BLOB

The default value is 0 (Standard). For more information about the various characteristic types, refer to “[Characteristic Types](#)” on page 112).

- **Value Type** — The value classification type to assign when storing the characteristic value in the database. Values must conform to the specified format. The value type setting also restricts the operations a user can perform when comparing or searching for characteristic values. The following list describes the legal value types and their lengths:
 - **1 (Integer)** — Store the numeric data value as an integer. This type has a default maximum number of five places for compatibility with database systems that operate with signed integer numbers (maximum of 32768). When connected to an Oracle database, EDM Library software handles an integer number value as a long integer (which permits up to ten places).
 - **2 (Double)** — Store the numeric value as a real number with a decimal point. The maximum number of characters, not including the decimal point, is 15. When checking the length setting of a double value type characteristic, the software adds the Value Length value (the number of integers to the left of the decimal point) and the Precision value (the number of integers to the right of the decimal point) to make sure the total does not exceed 15.
 - **3 (Char)** — A standard character string value. The maximum length depends on the character set you are using (refer your database system documentation for the maximum number of storable characters in a string).
 - **4 (Long)** — A long integer providing more places (a maximum of ten) than the integer data type.
 - **5 (Date)** — A date conforming to the date format defined on the Preferences dialog box. To ensure proper date format input, an administrator often sets the Time/Date Selection Box status bit on the **Status** tab of the characteristic definition, which enables a user to use the Date Chooser dialog box to enter date information.
 - **6 (BLOB)** — Store the value as a Binary Large OBject in the database. The maximum size of a binary large object is 4 GB.
- **Value Length** — The maximum number of characters in a characteristic value. The Value Length value cannot exceed the maximum length for a given value type. When the value type is double, Value Length denotes only the number of characters permitted to the left of the decimal point (the Precision value denotes the number of characters permitted to the right of the decimal point).

- **List No.** — The assignment number of the list to which the characteristic belongs. Characteristics with identical list numbers belong to the same list. Characteristics that are not part of any list have a value of 0 (the default).

When creating a new custom list frame and column characteristics, you must use a List No. value that is not already used by any other characteristic definitions within that class. To avoid conflict with lists in the default data model, values between 1000 and 1999 are reserved for custom list frames. To see values already used, search for Characteristic objects that have a List No value greater than 0.

- **Precision** — The number of digits permitted to the right of the decimal point for characteristics of value type double. For all other value types, Precision must be 0 (the default). EDM Library restricts the maximum number of decimal places to 13.
- **Parent List No.** — The list number of the parent list. The software uses this value when creating a hierarchical list structure. A 0 indicates the characteristic does not belong to a list or a hierarchical list.
- **from Init File** — If loading an initialization file created the characteristic, the system writes the name of the initialization file in this field. When creating a custom characteristic, entering a value in the **from Init file** field enables the **batchadmin** program to automatically write the characteristic definition to that initialization file.
- **Access Path** — A search path through the data model to the characteristic for an API client that uses the Path Query function. A value in the Access Path field is only required for characteristics used to view data values from referenced (foreign) object classes. The **Edit Path** button next to the Access Path input field provides an easy way to associate a search path to such a characteristic.

Once a data model is constructed and access paths are defined in a database, an administrator can use the **data_model_checker** command to check the integrity of access path definitions. The command checks an entire database and cannot be used for an individual characteristic.

“[Creating a Characteristic to View Data From a Foreign Class](#)” on page 204 contains an example of how to create an additional view characteristic and define its access path.

Text List



Note:

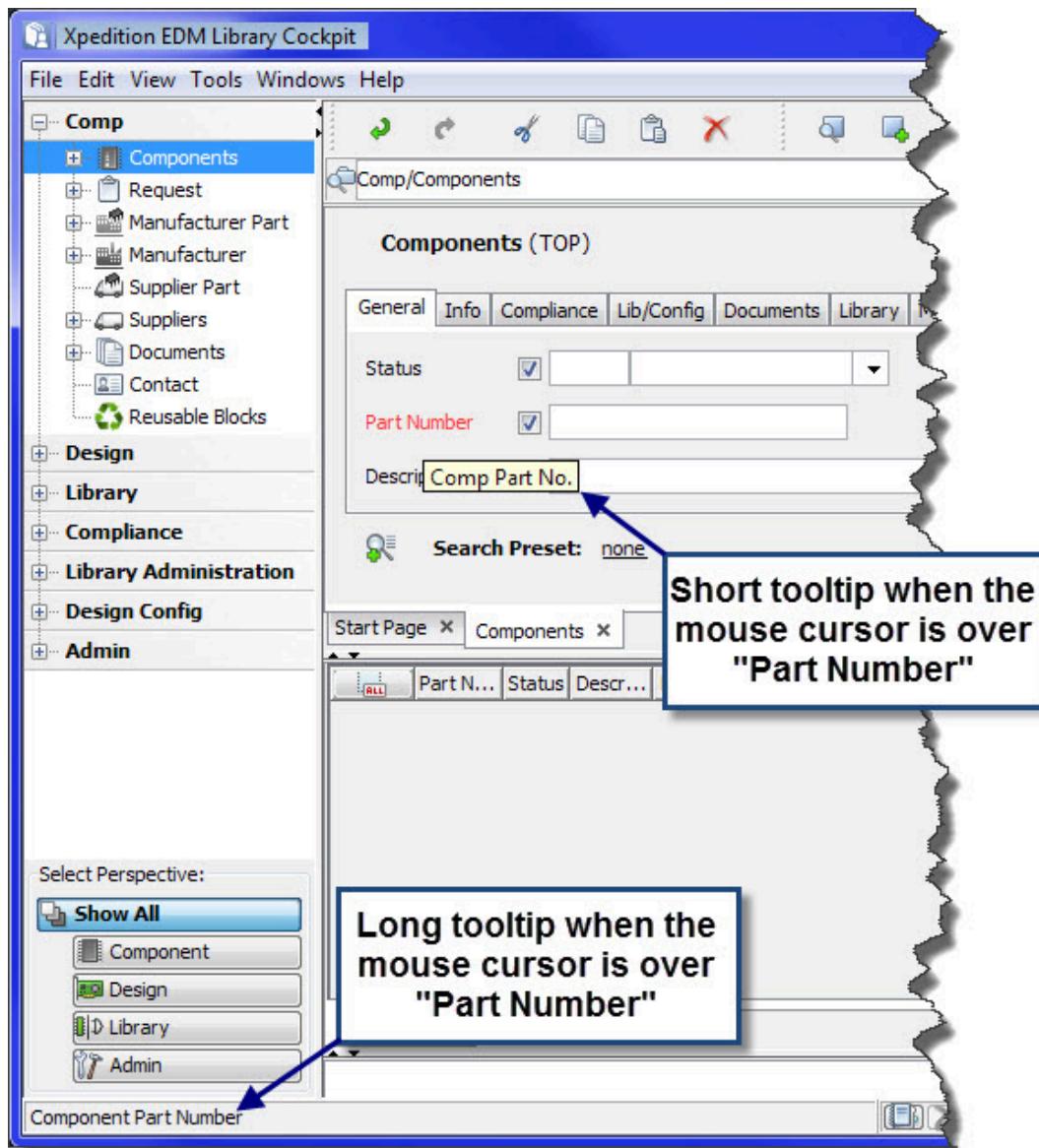
VX.2 and later releases only support English language labels and ignore non-English rows.

The Text list box in the middle of the **Top** tab facilitates the management of terminology in different languages. Each row in the Text List characteristic contains the following columns:

- **Tab Sheet** — The name(s) of the tab sheet(s) that displays the characteristic.
- **Information Text** — The name of the characteristic shown in the information window.
- **Search Text** — The name of the characteristic shown in the search window.
- **Language** — The language of the Information Text, Search Text, and Tab Sheet columns. The default setting (and only value the software recognizes) is “e” (English).

- **Tooltip Short and Tooltip Long** — Information that displays as popup text or in the message bar when placing the mouse cursor over the name of a characteristic (Figure 77). The Tooltip Short text displays at the mouse cursor. The Tooltip Long text displays in the message line.

Figure 77. Tooltips in the Xpedition EDM Library Cockpit Window



Options List

You can define a list of choices from which to select a value for the characteristic. These choices are called option list values, and display in the Option List list box. There is no restriction in the number of values. Option list values can be individually specified for dynamic characteristics of different catalog groups. To activate a characteristic as an option list characteristic and set up the dropdown list with the specified enumerated values, you must enable the Option List bit in the [Characteristic Object - Status Tab](#).

The Option List characteristic contains the following columns:

- **Catalog Group** — The key value of the catalog (for example, AAAB; refer to “[Catalog Group Key](#)” on page 84). This value is empty for most option list characteristics, which means that all characteristics within this object class will have the same option list values. However, you can specify different option list values for dynamic characteristics associated with different catalog groups. [Table 13](#) shows an example for catalog-dependent (dynamic) option list values.

Table 13. Dynamic Option List Values

Catalog Group	Option	Text	Sort
AAAA	U	Under Construction	1
AAAA	D	In Development	2
AAAA	A	Approved	3
AAAB	D	In Development	1
AAAB	A	Approved	2
AAAB	X	Obsolete	3

- **Option** — Contains a letter or number abbreviation to use as an enumerated value.
- **Text** — A description of the option list value.
- **Sort** — An additional characteristic used to display the option values in the selection list in a different order than alphabetical order. To specify a sort order, specify an integer value in the Sort column.
- **Ref. Characteristic** and **Reference Value** — Denote the reference characteristic and reference value.
- **Level** — Specifies the checks to perform when the characteristic has a specific option value. Used for Status characteristics such as 001obj_statu. The following level choices are available:
 - **1 (Released)** — Check mandatory fields when a user changes the object status to this value. [Figure 78](#) shows an example where the software should check all mandatory fields if the user changes the status of a component to ‘A’ or ‘R’.

Figure 78. Option List of the 001obj_statu Characteristic

The screenshot shows a software interface titled 'Option List'. It displays a table with columns: Catalog Group, Option, Text, Sort, Ref. Characteristic, Reference Value, and Level. The data is as follows:

	Catalog Group	Option	Text	Sort	Ref. Characteristic	Reference Value	Level
1		A	Approved/Released	3			1 Released
2		D	In Development	2			
3		R	Restricted	4			1 Released
4		U	Under Construction	1			
5		X	Obsolete	5			

- **2 (Move to next process step)** — Use if you have implemented the release and revision control feature. If this flag is not set, an object must go through all process states before it can be shifted to the next process level. By setting this flag, an object can be directly shifted to the next process level.
 - **3 (1 + 2)** — Combine the first two options.
 - **4 (1+2+auto ref. updates)** — Combine option three and the automatic update of the status of referenced objects. For example, level 4 is defined for the 'A' status value of a component. The component has a reference to its datasheet. A user changes the component status from 'U' (Under Construction) to 'A' (Approved). Clicking the **OK** button then causes EDM Library to check all mandatory fields. If all mandatory fields contain values and are set correctly, EDM Library sets the component and its referenced datasheet to 'A'.
- **from Init File** — If loading an initialization file created the characteristic, the system writes the name of the initialization file in this field.

Related Topics

- [Characteristic Object - Search Ref Tab](#)
- [Characteristic Object - Table Tab](#)
- [Characteristic Object - Placement Tab](#)
- [Characteristic Object - Status Tab](#)
- [Characteristic Object - Rights Tab](#)
- [Characteristic Object - Other Tab](#)
- [Characteristic Object - History Tab](#)

Characteristic Object - Search Ref Tab

To access: Refer to “[Opening a Characteristic Information Window](#)” on page 146.

The **Search Ref.** tab of a Characteristic information window displays references between the Characteristic object and class and Catalog Group objects.

Description

The **Search Ref.** tab of a Characteristic information window provides a way to visualize and navigate references between the Characteristic object and the Class and Catalog Group objects that use the characteristic.

Selecting a branch in the reference tree in the left pane displays a list of references to the Class or Catalog object in the right pane. Clicking the reference button next to the name of a referenced object in the list opens the information window of that object.

For example, [Figure 79](#) shows the **Search Ref.** tab for the 001obj_id (Part Number) characteristic. Selecting **Class > Catalog Groups** in the left pane shows the catalog groups in class 1 (Components) that use the characteristic in the right pane. Because 001obj_id is the key characteristic for the Components class, all objects in all component catalog groups starting with the top of the component hierarchy (catalog key AA) use the 001obj_id characteristic and are in the list. To open the information window for a particular catalog group, click the reference button next to the catalog group name.

Figure 79. Search Ref Tab of a Class Characteristic

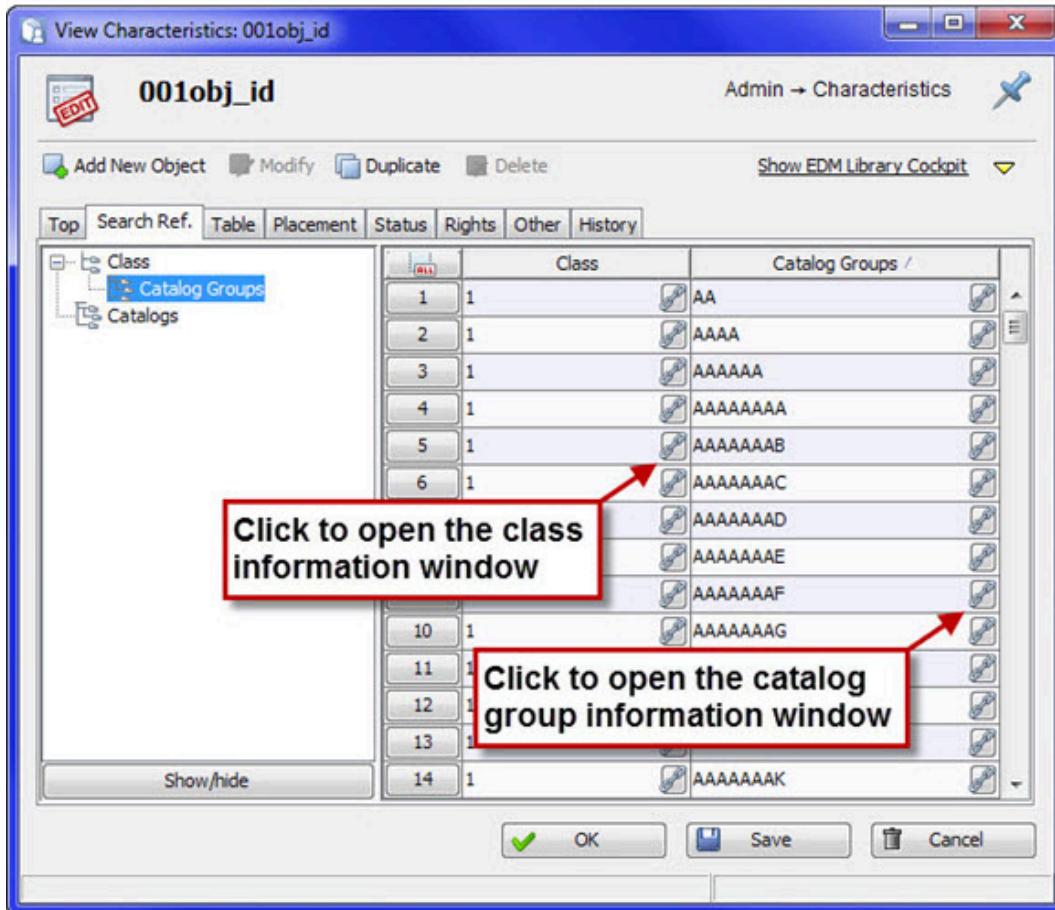
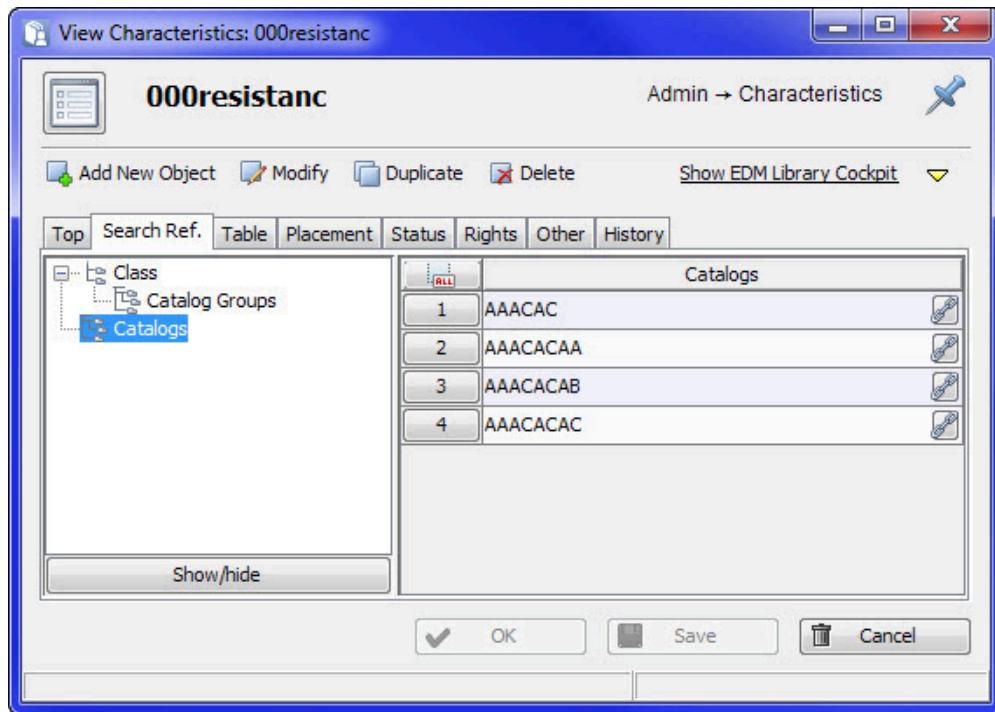


Figure 80 shows another example of the **Search Ref** tab, but this time for the dynamic 000resistanc (Resistance) characteristic. Because 000resistanc is only in the catalog groups for resistor components, the list of referenced catalogs is much shorter.

Figure 80. Search Ref Tab of a Dynamic Characteristic



Characteristics

Because the **Search Ref.** tab is only for navigation, there are no data input characteristics on the tab. The columns in the reference list in the right pane depend on the object type chosen from the hierarchy in the left pane.

Usage Notes

To show or hide branches in the reference tree on the left side of the **Search Ref.** tab, use the **Show/Hide** button at the bottom of the pane. The show/hide branch settings are persistent from session to session and remain in effect for all objects of that type until you change them.

Related Topics

[Class Characteristics and Dynamic Characteristics](#)

[Characteristic Object - Top Tab](#)

[Characteristic Object - Table Tab](#)

[Characteristic Object - Placement Tab](#)

[Characteristic Object - Status Tab](#)

[Characteristic Object - Rights Tab](#)

[Characteristic Object - Other Tab](#)

[Characteristic Object - History Tab](#)

Characteristic Object - Table Tab

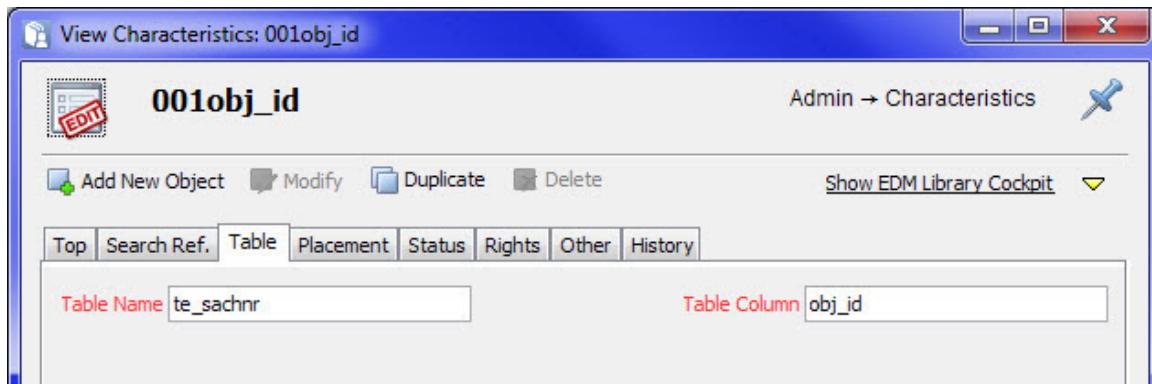
To access: Refer to “[Opening a Characteristic Information Window](#)” on page 146.

The **Table** tab of a Characteristic information window specifies the name of the table and the column in the table of the data model where EDM Library should store the values for the characteristic.

For more information, refer to “[Classes, Class Number, and Class Tables](#)” on page 104.

Description

Figure 81. Table Tab of a Characteristic



Characteristics

- **Table Name** — Specifies the name of the table in the database where the software stores the characteristic values. Each object class resides in a table in the database. List characteristics, which require a separate table, are an exception.

Characteristics that are common for all objects of an object class reside in the table associated with that class (for example, to add another input field inside the Components object class, you would use the te_sachnr table). Catalog-specific characteristics reside in the te_dyn table.

- **Table Column** — Corresponds to the name of a column in the database table and must be unique for each characteristic.
If you are creating a new characteristic, saving the characteristic object automatically creates the table column in the object class table. Deleting a characteristic does not delete the table column or any data residing in that column from the database.



CAUTION:

Do not use a table name or column name that is a reserved keyword in SQL (for example, SELECT, UPDATE, and so on). For a list of reserved words in SQL, refer to your database documentation.



CAUTION:

The combination of the Table Name and Table Column should not exceed 30 characters. This restriction is necessary because, when searching, Oracle forms the index names by concatenating <table_name>_<column_name> and only permits a maximum of 30 characters per index. If the resulting Oracle index exceeds 30 characters, Oracle can issue an ORA-00972 error.

Related Topics

[Classes, Class Number, and Class Tables](#)

[Characteristic Object - Top Tab](#)

[Characteristic Object - Search Ref Tab](#)

[Characteristic Object - Placement Tab](#)

[Characteristic Object - Status Tab](#)

[Characteristic Object - Rights Tab](#)

[Characteristic Object - Other Tab](#)

[Characteristic Object - History Tab](#)

[Creating a Standard Characteristic](#)

Characteristic Object - Placement Tab

To access: Refer to “[Opening a Characteristic Information Window](#)” on page 146.

The **Placement** tab of the Characteristic information window defines the position of the characteristic and its input field in the search and information windows, and the sort and list order.



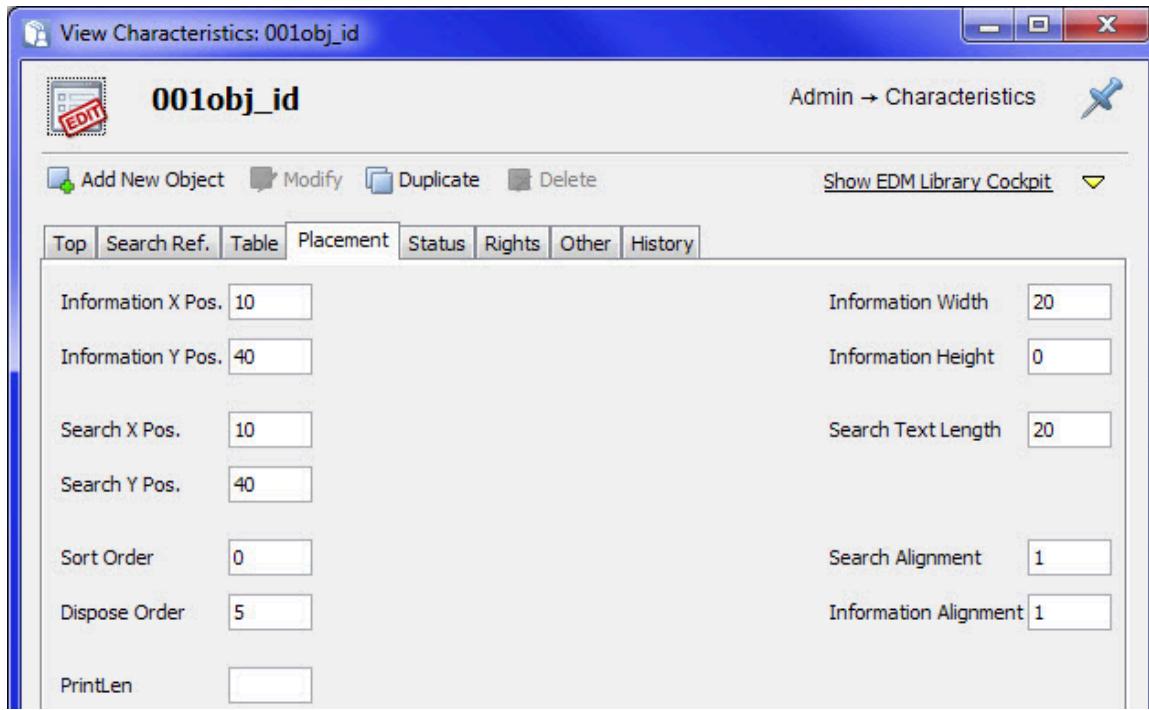
Note:

In the Xpedition EDM Library Cockpit application, compose mode enables a privileged user to interactively add, delete, and edit characteristic placement and input parameters (such as the length and height of an input field) on a search criteria tabbed sheet or an object information tab. When using compose mode, the software automatically enters values on the **Placement** tab as you graphically position and size characteristics.

Attempting these same tasks by manually editing values on the **Placement** tab of a Characteristic object is often a process of trial and error to get the placement correct. For more information about compose mode, refer to “[Using Compose Mode to Edit Characteristic Definitions](#)” on page 143.

Description

Figure 82. Placement Tab of a Characteristic



Characteristics

- **Information X Pos** — Defines the X-coordinate of the characteristic in the information window in pixels. The reference point is the top left corner of the tab. Entering 0 automatically positions the characteristic in EDM Library Cockpit. The default value is 10.
- **Information Y Pos** — Defines the Y-coordinate of the characteristic in the information window in pixels. The reference point is the top left corner of the tab. Entering 0 causes EDM Library Cockpit to automatically position the characteristic. The default value is 10.
- **Information Width** — The value depends on if the characteristic is a list characteristic, an action characteristic, or a standard characteristic.
 - For list and action characteristics, the value is the width of the text characteristic in pixels.
 - For standard characteristics, the entry corresponds to the length of the information text; that is, the number of digits of the input field in the information window. The default value is 20.
- **Information Height** — The height of the list or button in pixels. Only applicable if the characteristic is a list characteristic (type 5) or an action characteristic. Otherwise, the software ignores the value.



Note:

If the characteristic is a list or an action characteristic, you must enter a value in this field; otherwise, no list or action button displays.

When the characteristic type is defined as a list frame characteristic, entering a positive number for the information width and information height displays the list frame according to these values. Scroll bars display if the list frame characteristic exceeds the size of the entire information window. To avoid the scroll bars, specify “-1” for information width or height to have EDM Library Cockpit automatically scale the list frame according to the current size of the information window.



Note:

Do not use a -1 value for the height of a list frame characteristic that resides in a tab containing dynamic characteristics if the dynamic characteristics are placed automatically (that is, the dynamic characteristic has an x,y position of 0,0). Because of the position setting, EDM Library Cockpit places the dynamic characteristic at the bottom of the tab, and the -1 list frame Information Height setting causes EDM Library Cockpit to cover the dynamic characteristic with the edges of the list frame. When a tab contains both, a list frame characteristic and an automatically positioned dynamic characteristic, always specify a positive value (such as 200) for the Information Height value of the list frame characteristic.

- **Search X Pos** — The X-coordinate of the characteristic in the search window in pixels. The reference point is the top left corner of the tab. Entering a value of 0 causes EDM Library Cockpit to automatically position the characteristic. The default value is 10.
- **Search Y Pos** — The Y-coordinate of the characteristic in the search window in pixels. The reference point is the top left corner of the tab. Entering a value of 0 causes EDM Library Cockpit to automatically position the characteristic. The default value is 10.
- **Search Text Length** — The number of digits of the input field in the search window. The default value is 20.
- **Sort Order** — The sort order of the characteristics in the selection list. The characteristic with the lowest number is sorted first. A positive number indicates an ascending sort. A negative number indicates a descending sort. The default value is 0, which indicates that characteristics in the selection list are not sorted.
- **Dispose Order** — The position of individual characteristics in the search results list. Characteristics are arranged in ascending order of numeric values.

The software also uses the Dispose Order value for the tab order. A value for Dispose Order of 0 to 100 displays the characteristic in the first tab. A Dispose Order value of 101 to 200 displays the characteristic in the second tab, and so on.

There are two ways to format the search results list:

- By entering values for Sort Order and Dispose Order in the characteristic definition.
- By clicking a characteristic label in the search window.

The values entered for Sort Order and Dispose Order on the **Placement** tab are permanently stored in the database, whereas the values defined in the table that displays when clicking a characteristic label in the search window are only valid for the current session and are reset when restarting EDM Library Cockpit.

- **Search Alignment** — Controls the column alignment of the characteristics in the search window, as shown in [Figure 83](#). The default value for Search Alignment is 1.

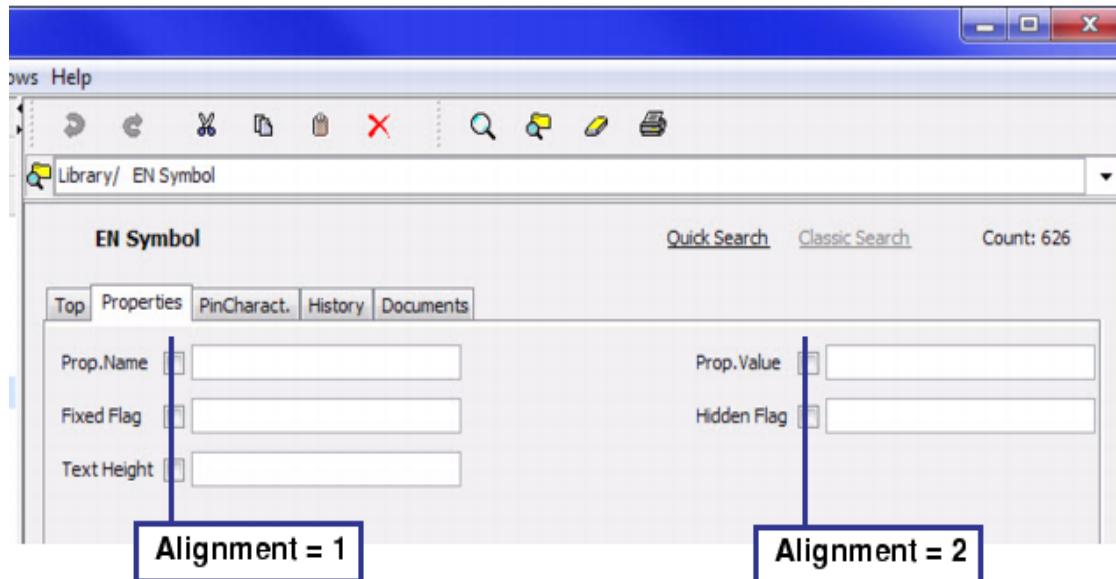
When entering a value for Search Alignment, note the following rules:

- The value must be an integer. All static characteristics with the same numeric value are then displayed in a group with the same x-coordinate for the input field width in the search window. The width is determined by the longest characteristic label.
- Numbering must start with 1 and be continuous (2, 3, 4, and so on) for further characteristic groups. Static characteristics with a value of 1 are in the leftmost column on the tabbed page, characteristics with a value of 2 are in the second leftmost column, and so on.
- EDM Library Cockpit ignores alignment numbering of action characteristics.
- All dynamic characteristics use a common method of search alignment and information alignment that is one number above the setting for the static characteristic.

For example, if static characteristics on a tabbed page use search and information alignment values of 1 (left column) and 2 (right column), then setting the value for Search Alignment and Information Alignment to 3 (the lowest free alignment number) fits dynamic characteristics immediately below the last static characteristic in each column.

Dynamic characteristics must not use a search and information alignment number already used by static characteristics on the tabbed page.

Figure 83. Search Alignment Example



- **Information Alignment** — Controls the alignment of the characteristics in the information window. The definition is identical to alignment in the search window. The default value is 1.
- **PrintLen** — Specifies the length of the characteristic (that is, the number of characters output per line) when printing the characteristic. The software inserts a line break after reaching the number of characters specified by PrintLen.

Related Topics

[Using Compose Mode to Edit Characteristic Definitions](#)

[Characteristic Object - Top Tab](#)

[Characteristic Object - Search Ref Tab](#)

[Characteristic Object - Table Tab](#)

[Characteristic Object - Status Tab](#)

[Characteristic Object - Rights Tab](#)

[Characteristic Object - Other Tab](#)

[Characteristic Object - History Tab](#)

[Creating a Standard Characteristic](#)

Characteristic Object - Status Tab

To access: Refer to “[Opening a Characteristic Information Window](#)” on page 146.

The **Status** tab of a Characteristic information window contains several checkboxes that set the properties and defines the behavior of the characteristic. Each checkbox on the **Status** tab corresponds to a status bit that is either on (checked) or off (unchecked).

Description

Checking or unchecking specific settings on the **Status** tab control the areas shown in [Table 14](#). Status bits with a “---” label represent settings no longer used by the software.

Table 14. Characteristic Status Settings

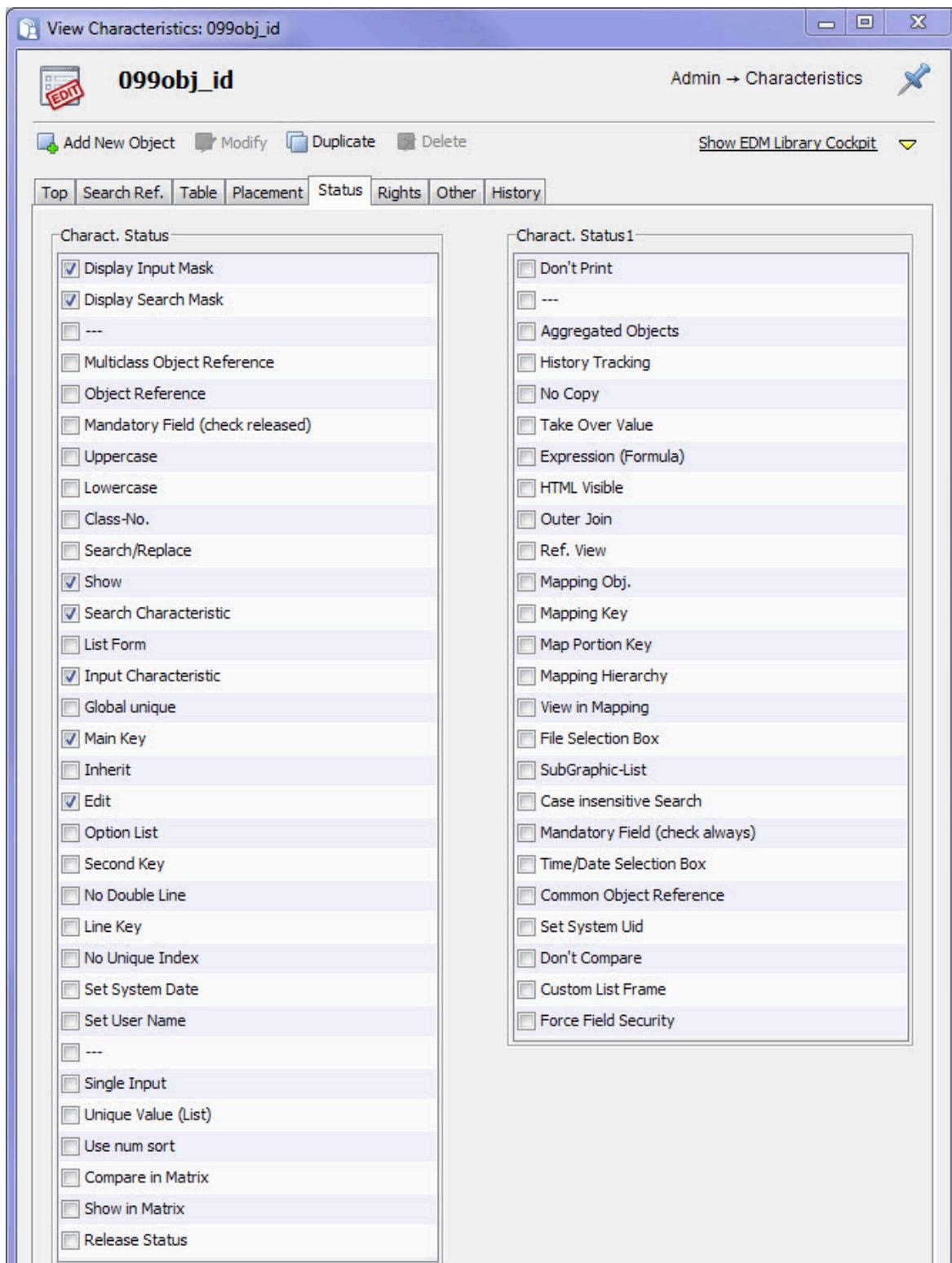
To Control:	Check or Uncheck:
Display	Display Search Mask, Display Input Mask, Show, Show in Matrix, Don't Print, HTML Visible, Ref View
Appearance	Option List
Characteristic functionality	Input Characteristic, Search Characteristic, Object Reference, Main Key, Second Key, Multiclass Object Reference, Global Unique, Line Key, List Form, Use num sort, Compare in Matrix, Sub Graphic List, History Tracking, Aggregated Objects, Release Status

Table 14. Characteristic Status Settings (continued)

To Control:	Check or Uncheck:
Input functionality	Inherit, Edit Allowed, Single Input, Mandatory Field, Uppercase, Lowercase, Search/Replace, Set User Name, No Copy, Set System Date, File Selection Box, Expression (Formula)
Database functionality	Class-No., No Double Line (select distinct), No Unique Index, Unique Value (List), Outer Join

Figure 84 shows the **Status** tab of a Characteristic information window.

Figure 84. Status Tab of a Characteristic



The 056smt_stat bit status system characteristic assigns labels to status bits in the left column of the **Status** tab, while the 056smt_stat1 system characteristic assigns labels to the status bits in the right

column. Do not change the status bit labels because updating software on your server can overwrite your label customizations.

Characteristics

- **Display Input Mask** — Checked, displays the characteristic in the information window.
If the [Characteristic Object - Rights Tab](#) grants a user VIEW or EDIT permission, the characteristic can still display in the information window when viewed by that user despite Display Input Mask being unchecked on the **Status** tab. Specific user rights always override the Display Input Mask setting.
- **Display Search Mask** — Checked, displays the characteristic in the advanced search window. You must also check Search Characteristic when Display Search Mask is checked.
- **Multiclass Object Reference** — Defines if an Object Reference characteristic (see the next status bit) can have a reference to more than one object class. Checked, the characteristic value can derive from different object classes. Unchecked, the characteristic value is not a multi-class object reference.

For more information about an multiclass object reference characteristic, refer to “[Reference Characteristics](#)” on page 138.

- **Object Reference** — Checked, specifies if the content of this characteristic is a reference to the key of another object class. Supply the reference by giving the key of the referenced class in the Ref. Class field on the **Top** tab.

For more information about an object reference characteristic, refer to “[Reference Characteristics](#)” on page 138 and “[Creating an Object Reference Characteristic](#)” on page 216.

- **Mandatory Field (check released)** — Checked, specifies that the characteristic must have a data value to promote the object to an approved or released state. The system only does the check if the object class has a status characteristic (for example, 001obj_statu).

If the characteristic definition defines an option list and has Mandatory Field (check released) checked, the system also checks if the characteristic value is in the defined set of option list values. If not, the user cannot release the object.

Unlike Mandatory Field (check always), a characteristic with the Mandatory Field (check released) status bit checked is not red in the EDM Library Cockpit user interface until after releasing the object.

- **Uppercase and Lowercase** — Controls the input format of the data value.

Checking Uppercase converts characters to uppercase as they are typed into the search field or data field of the characteristic, thereby preventing a user from entering lowercase letters. Unchecking Uppercase defers to the value of the Lowercase bit.

Checking Lowercase converts the characters to lowercase as they are typed into the search field or data field, thereby preventing a user from entering uppercase characters. If Lowercase is unchecked, the input format defers to the value of Uppercase.

If both Lowercase and Uppercase are unchecked, a user can type both lowercase and uppercase letters into the search field or data field.

When saving the characteristic definition, the system issues an error if both Uppercase and Lowercase bits are checked.

- **Class-No.** — Checked, inserts the class key and the version in front of the characteristic contents. Unchecked, leaves the data value unchanged.

The Class No. bit is necessary because of central tables in the data model that hold the keys of different object classes (for example, the te_obj table). But a key must have a unique value. Putting the class number in front of each key value makes the key unique.

You must check **Class-No.** under these conditions:

- You create a key (that is, an obj_id characteristic) and the TE_OBJ flag is set for the corresponding object class.
- You create a reference to another characteristic that has the Class-No. flag checked.
- You create a list frame characteristic key and the Class-No. flag is checked for the obj_id of the class that will host the list frame.

You do not need to check **Class-No.** for non-key characteristics and that do not reference other characteristics that have Class-No. set.

- **Search/Replace** — Checked, specifies that a bulk modification (search and replace) operation can modify the characteristic value.



Note:

Never set the Search/Replace bit for a key characteristic (<class_no>_objid). By definition, a key characteristic must always have a unique value. Enabling bulk modification of key characteristic values can invalidate the uniqueness of the characteristic.

- **Show** — Checked, adds a checkmark next to the characteristic in the advanced search window, thus making the characteristic a column in the search results list by default.

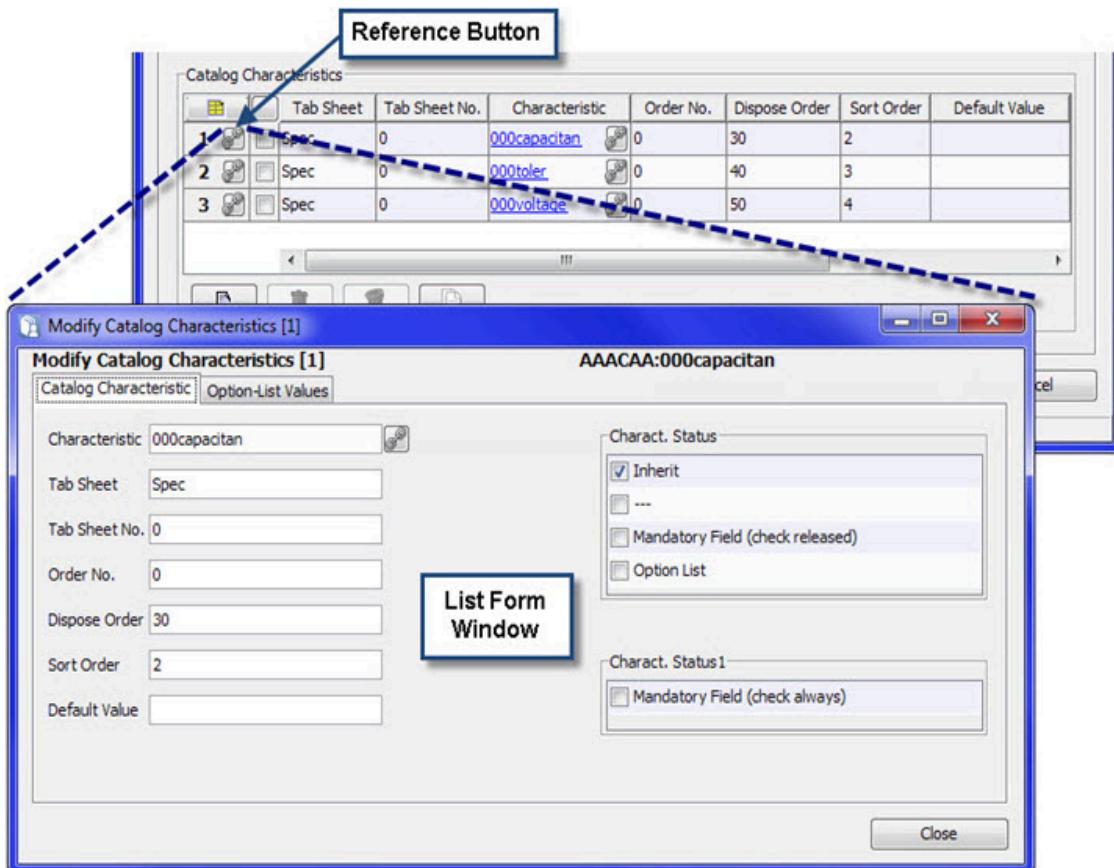
Do not check Show on list frame characteristics. On an action button characteristic, checking Show displays the button inside the search results list.

- **Search Characteristic** — Specifies that a user can include the characteristic in a search. You must check Search Characteristic when Display Search Mask is checked.
- **List Form** — Checked, defines a list form for a list frame characteristic and displays a reference button to the left of each line of the list (Figure 85). Clicking the button opens the form where a user can input data for each line. This feature is useful if the list contains many columns and you do not want a user to have to scroll across the list to input data.

Unchecked, represents the standard list frame characteristic without a list form.

You can also use list forms to create hierarchical lists (a list inside a list) as described in “[Creating Hierarchical Lists \(List Frames Within List Frames\)](#)” on page 240.

Figure 85. List Frame Characteristic Input Form



- **Input Characteristic** — Checked, specifies that this characteristic is for input and that the system stores the data value a user enters in the database table.

Always check Input Characteristic when Edit is checked. Otherwise, it will be confusing if a user can edit the field but nothing is stored.

- **Global Unique** — When checked on a characteristic that is not a main key, specifies that the characteristic data value must be unique within the object class. Unchecked means that duplicate characteristic values can occur within the characteristic database table.

For example, the Components object class contains two characteristics named Part Number and Datasheet. You want a unique Datasheet value for each component so that two components cannot have the same datasheet. To satisfy this condition, set the Global Unique bit for the Datasheet characteristic.

Note that using the Global Unique bit for characteristics stored in a central table (for example, the te_obj table) that holds all database objects facilitates a data value that is unique within the whole database.

- **Main Key** — Checked, makes a characteristic the main key of the object (for example, all object IDs). Check Main Key for all characteristics named obj_id and all list frame keys.

Do not check Main Key for standard characteristics such as input fields, action buttons, text frame characteristics, or list columns.

- **Inherit** — Checked, specifies that a subgroup inherits this characteristic. The setting is only valid for dynamic characteristics allocated to individual catalogs, (that is, the characteristics belonging to the object class number 0). Characteristics allocated to an object class different from 0 are always inherited.
- **Edit** — Checked, permits editing of the data field of this characteristic in the information window by any user with edit class rights.

If unchecked, you can still allocate user-dependent or group-dependent edit permissions as described in “[Setting User-Based or Group-Based Characteristic Editing Permissions](#)” on page 241.

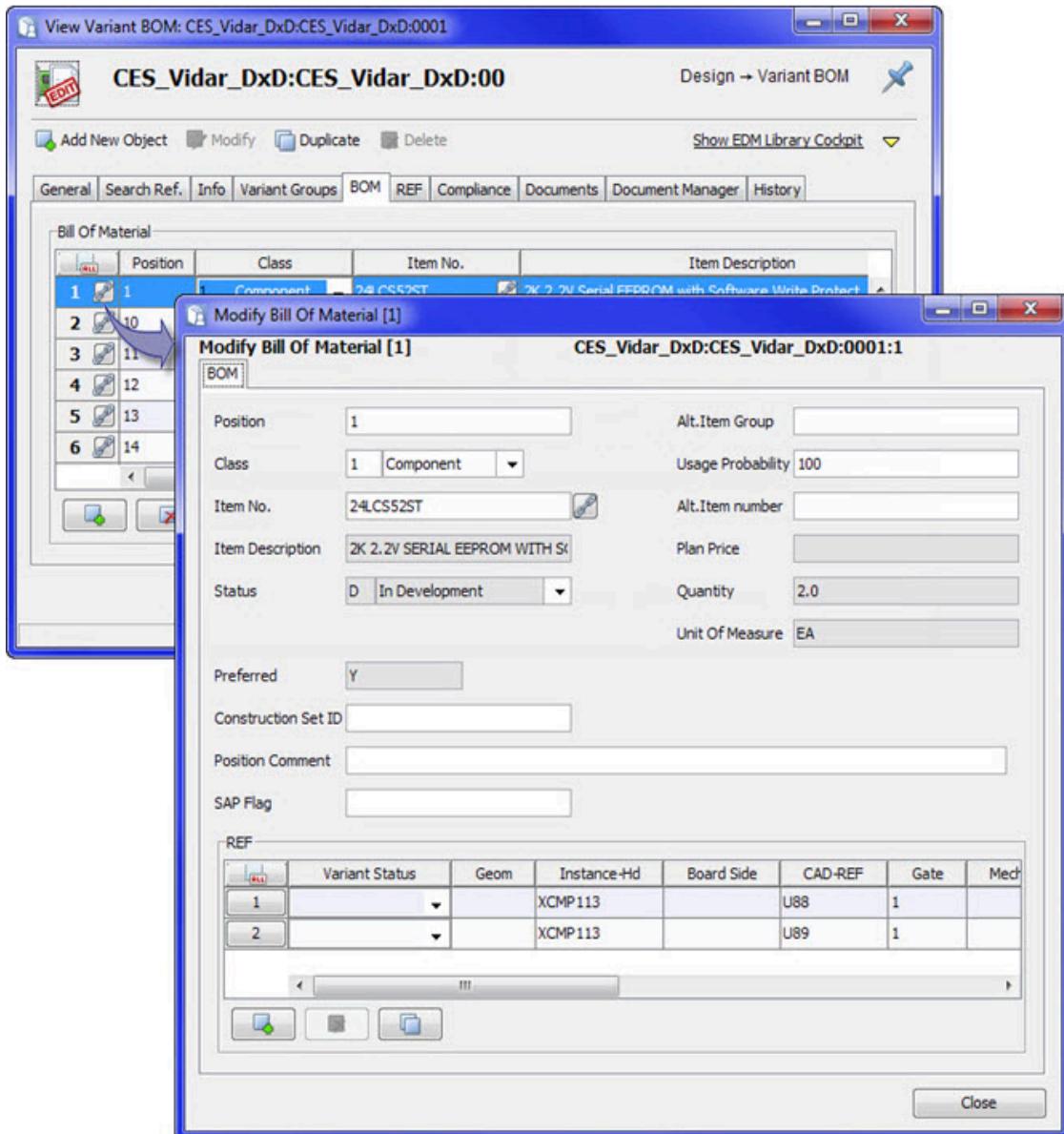
A special case occurs for the Status (obj_statu) characteristic. Unchecking Edit on the **Status** tab of the obj_statu characteristic does not permit unauthorized users to modify objects with an A (Approved/Released) status value. In this example, you should specify EDIT rights for only users authorized to change the status in the **Rights** tab of the obj_statu characteristic.

Checking **Edit** for action button characteristics enables a user to click the button. If unchecked, a user cannot click the action button.

- **Option List** — Checked on a standard characteristic type displays the option list choices (specified on the **Top** tab) in a dropdown list.
Defining option list values does not restrict data input to only those predefined option values.
- **Second Key** — Checked, makes the characteristic the second key of an object. The system uses the bit to display data from two different tables of an object class within one list frame inside an object information window.

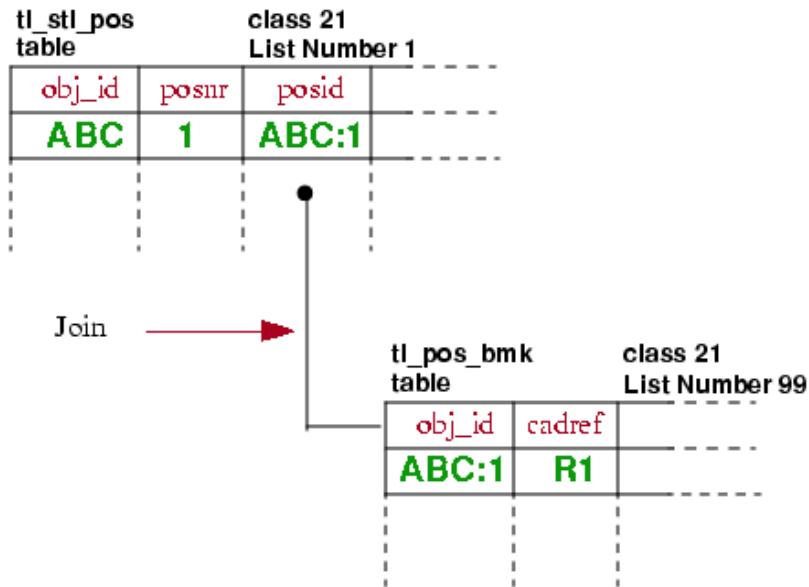
The Variant BOM object class provides an example of characteristics that use Second Key. The **BOM** tab ([Figure 86](#)) of the Variant BOM information window contains a list of all BOM positions with a sub list form containing another list of REF designators.

Figure 86. BOM Tab of a Variant BOM With Sub Positions



EDM Library stores data about the positions in the tl_stl_pos table, and stores data about the REF sublist in another table named tl_pos_bmk. [Figure 87](#) shows the structure in the database tables.

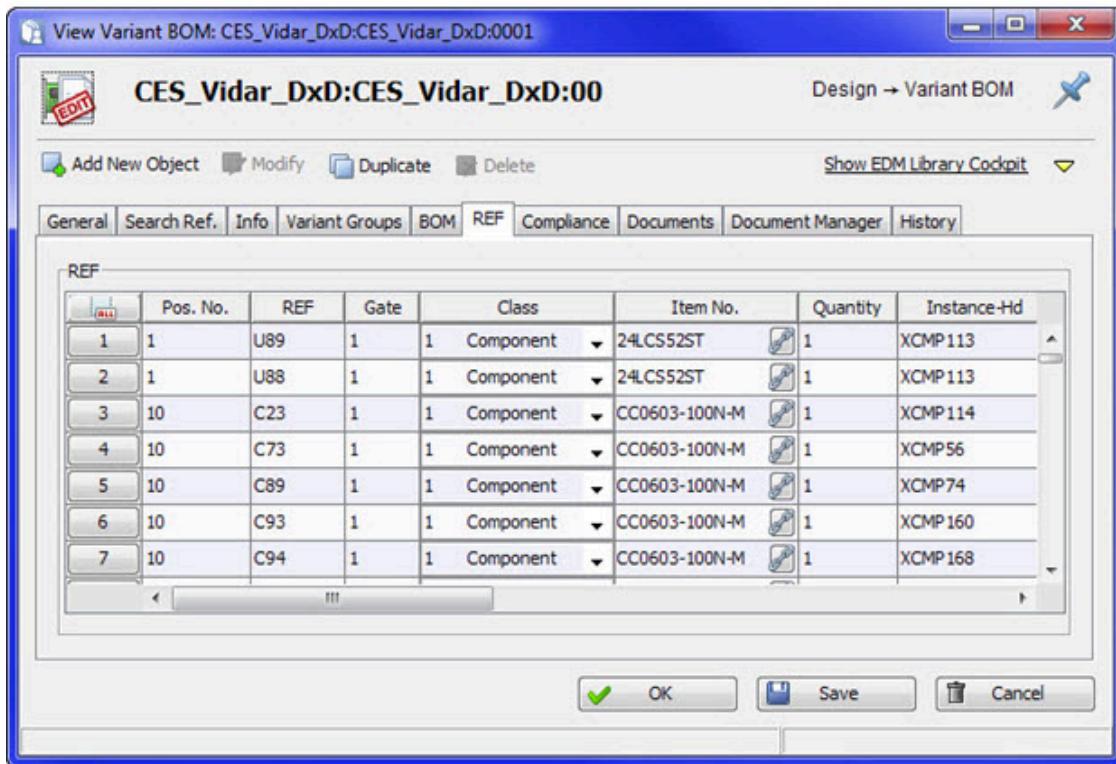
Figure 87. Structure of Pos and REF Tables



These two lists are used for data input.

The **REF** tab (Figure 88) of a Variant BOM information window contains a view of all REF designators of all positions in a variant BOM. To accomplish this view, the **REF** tab displays data from the **tl_stl_pos** table and the **tl_pos_bmk** table within the **REF** list. In a standard list, this is not possible because such a list in the information window can only display data from one table.

Figure 88. REF Tab of a Variant BOM

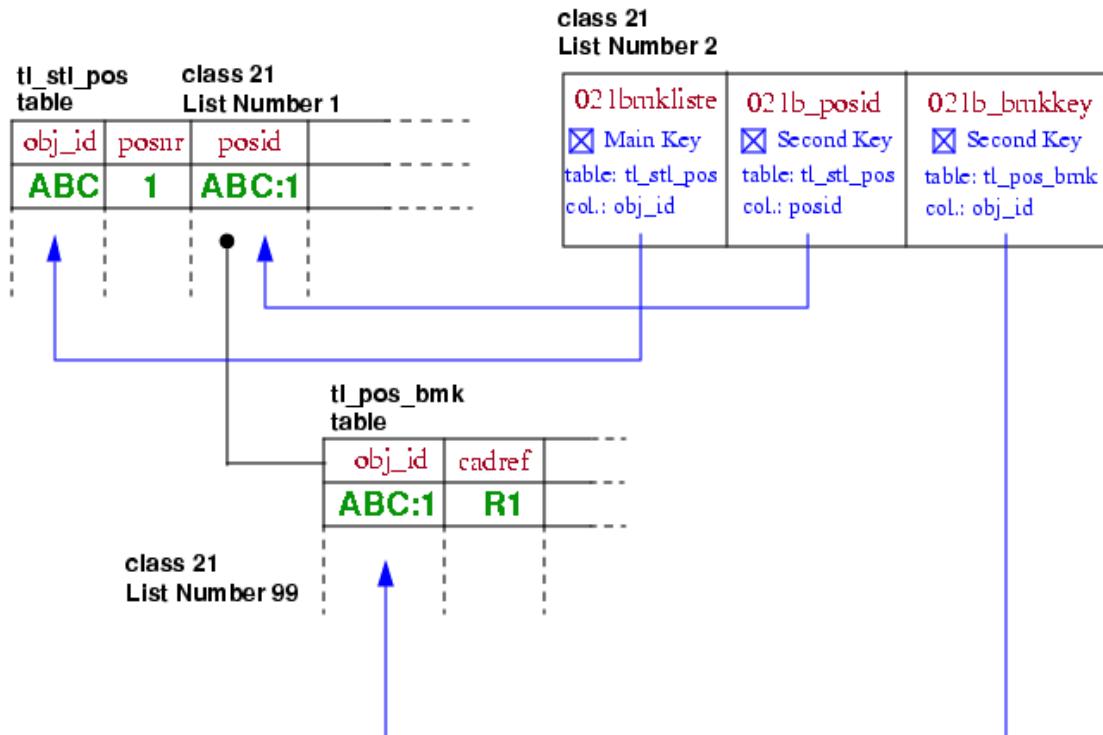


Therefore, you can use the Second Key status bit to display data from two different tables in one list, and it must be set for two characteristics pointing to the joined columns of both lists (tl_stl_pos.posid and tl_pos_bmk.obj_id in the example of [Figure 87](#)).

In addition, the list making use of Second Key must be joined to the obj_id of the object class.

[Figure 89](#) shows the structure and configuration of the main characteristics of the REF list.

Figure 89. Complete Structure for the REF Tab



Having joined all tables as shown in [Figure 89](#), additional columns from both tables (tl_stl_pos and tl_pos_bmk) can be referenced by list number 2 for display.



CAUTION:

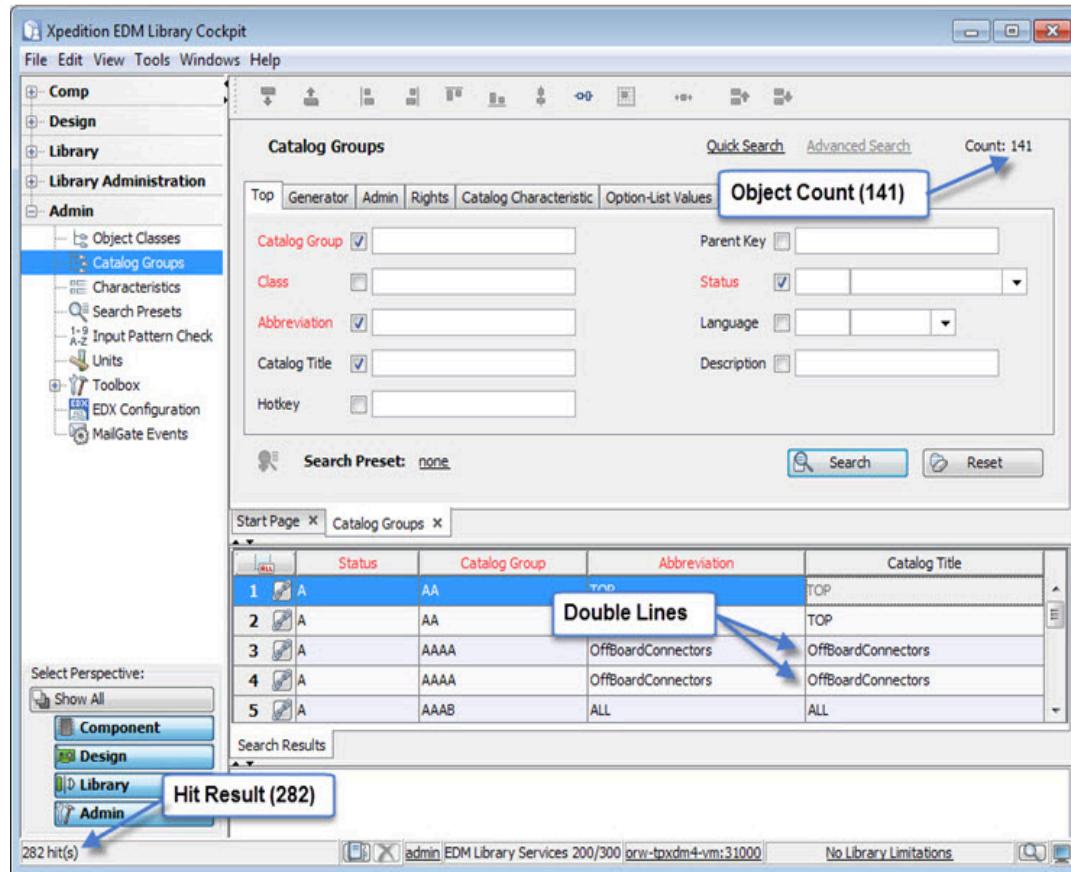
Do not activate the Input Characteristic status bit for any characteristic of list number 2, as it could damage your database.

- **No Double Line** — Checked, specifies that a characteristic can only display in the search results list once. This bit is useful when specific search criteria causes several characteristics to show up more than once in the search results list (that is, the same line occurs twice). The No Double Line bit suppresses those duplicate lines.

The following example shows how No Double Line affects the search results list:

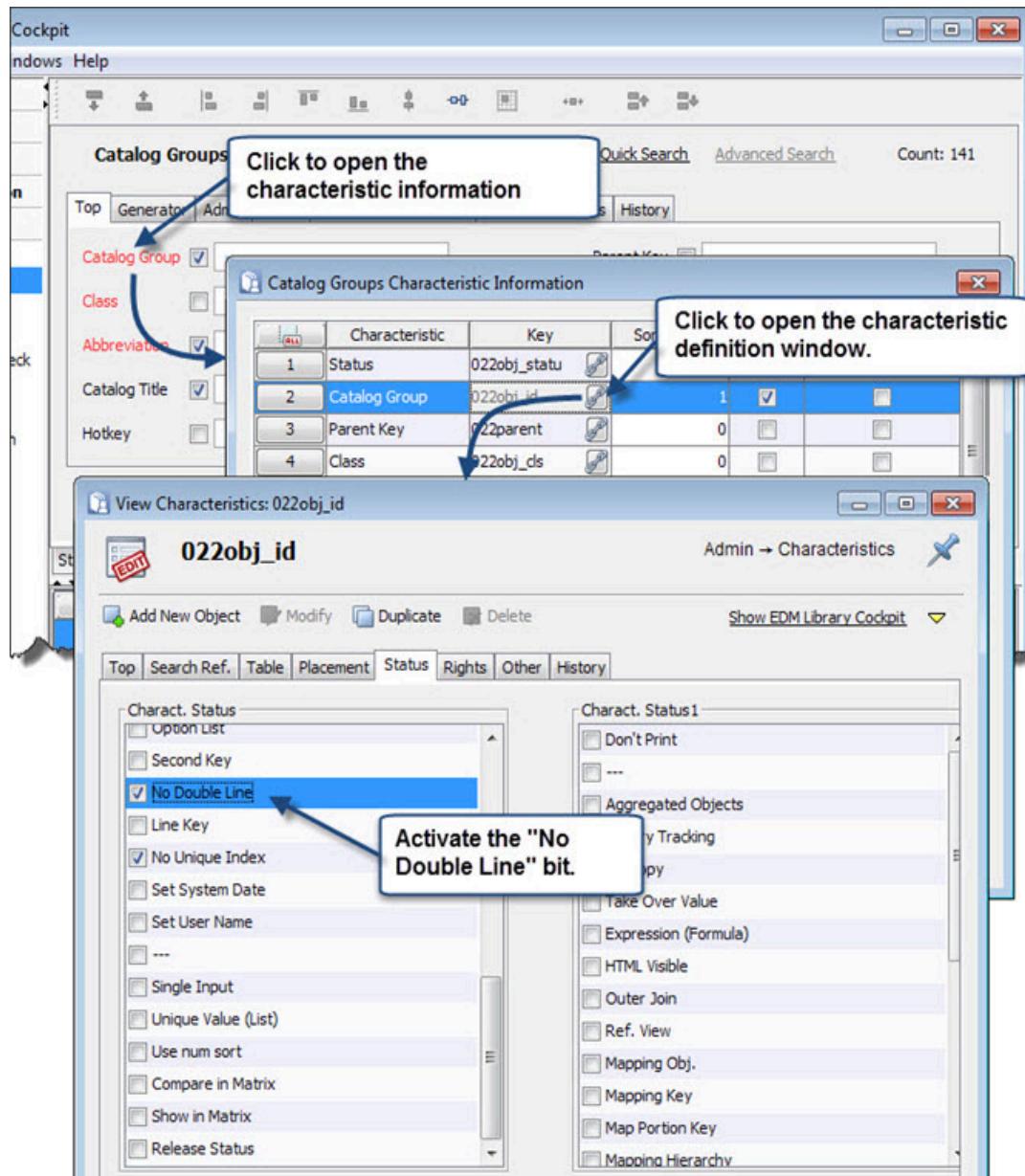
- In the advanced search pane, checking only the Catalog Group, Abbreviation, and Catalog Title search characteristics produces the search results in [Figure 90](#). There are only 141 Catalog Group objects in the database, but there are 282 lines in the search result area. Many of the lines in the search result area are duplicates.

Figure 90. Example of Double Lines in a Search Result



- Clicking the Catalog Group characteristic label in the advanced search pane displays the Characteristic information window in [Figure 91](#). Clicking the reference button next to the active '022obj_id' characteristic opens the characteristic definition in modify mode. Bringing the **Status** tab forward shows that the No Double Line status bit is unchecked.

Figure 91. Characteristic Information for a Catalog Group



- Checking No Double Line, saving the characteristic definition and then repeating the search shows that there are no longer double lines in the search results.
- **Line key** — Checked, identifies the characteristic as a line key for a list. A line key must have a unique value. Unchecked, means that this is a standard characteristic and is not used as a line key for any list.

The Line Key characteristic in a list can contain a unique value specified by the user (1, 2, 3, and so on). Alternately, you can configure a line key characteristic to take over values that are unique when concatenated by checking Take Over Value and specifying the characteristics to concatenate in the **Other** tab.

For example, Figure 92 shows how to set up a Line key characteristic.

Figure 92. Example Line Key Setup

Nameslist		
first_name	last_name	Listkey (*line key flag active, Default Value = ^first_name^:^last_name^)
John	Smith	JohnSmith
John	Miller	JohnMiller

If Line Key is checked, and both the first_name and last_name columns are included as a concatenation inside the Default Value input field on the **Other** tab, the combinations “John:Smith” and “John:Miller” are unique.

- **No Unique Index** — Checked, does not use the characteristic to uniquely identify the object.
The No Unique Index bit is necessary for list frame characteristics. Checking facilitates inputting several lines of data into the list. Only one line can be input into the list if No Unique Index is unchecked.
- **Set System Date** — Checked, automatically enters the current date and time as a characteristic value. The data model uses Set System Date to write a timestamp into the Created at and Modified at characteristics in the **History** tab.
The characteristic definition for the Created at input field has both the Set System Date and the Single Input bits set. Thus, the system generates the date of object creation once and never again changes the value. The Modified at input field has only Set System Date checked and gets modified with the current date and time whenever a user or program edits the object.
- **Set User Name** — Checked, automatically enters the name of the user who modified the object as a characteristic value. The data model uses Set User Name to supply a value for the Created by and Modified by characteristics in the **History** tab.
The characteristic definition for the Created By input field has both the Set User Name and the Single Input bits set. Thus, the system enters the name of the user who created the object once and never again changes the value.
- **Single Input** — Checked, permits the value of the characteristic to be defined only once (usually when creating the object), and then does not permit further changes. Characteristics that use Single Input include the Created by and Created at characteristics on the **History** tab.
- **Unique Value (List)** — Checked, specifies that the characteristic value in a list must be unique and not be a characteristic value in any other column. Unchecked means that the value does not have to be unique and can occur multiple times.
- **Use num sort** — Checked, uses numeric sorting and only applies to characteristics of the Char type.

For example, a characteristic with value length of 3 is defined for an object class, with a value type of Char. The characteristic name is CAD Ref. Four objects are created with the following values for the CAD Ref characteristic: C1, C2, C10, and C11. If Use num sort is not checked, the

software sorts these values in the search results list as C1, C10, C11, C2. When Use num sort bit is checked, the software sorts the values as C1, C2, C10, C11.

- **Compare in Matrix** and **Show in Matrix** — These status bits are obsolete and no longer read by the system.



Note:

Compare in Matrix and Show in Matrix only affected the older DMS Classic interface (now discontinued) and do not have an effect in the EDM Library Cockpit interface.

- **Release Status** — Checked, specifies that the object cannot be opened for editing if it has reached a release status. Unchecked, permits editing of an object independent of the release status.

Checking **Release Status** has two effects:

- Once an object has reached a release status, it can no longer be opened in modify mode as defined in the enumerated (option) values list inside the **Top** tab of the status characteristic.
- The software checks all object references to other object classes to see if the referenced objects also have a release status. A referenced object must also have a Release Status defined and be an input characteristic.

For example, [Figure 93](#) shows the status values defined on the **Top** tab in the Component status (001obj_statu) characteristic. The 001obj_statu characteristic has five option values (A, D, R, U, and X). Setting the Level value to '1' define that the A and R status values denote a release state.

Figure 93. Top Tab of Status Characteristic Definition

	Catalog Group	Option	Text	Sort	Ref. Characteristic	Reference Value	Level
1		A	Approved/Released	3			1 Released
2		D	In Development	2			
3		R	Restricted	4			1 Released
4		U	Under Construction	1			
5		X	Obsolete	5			

On the **Status** tab:

- Option List is checked.
- Edit is unchecked, but the **Rights** tab allocates EDIT rights for administrators (or component engineers).
- Release Status is checked.

The characteristic definition is now configured so that a normal user without administrator or component engineer privileges can only edit objects if the objects are in the “D” status. As soon as an administrator or component engineer sets the status of an object to “A”, the software checks mandatory fields and most users cannot open the object in modify mode. Administrators and component engineers can always change the Status value.

- **Don't Print** — Checked, does not print the data value with the characteristic.
- **Aggregated Objects** — Checked, copies and deletes reference objects along with the main object. Aggregated objects consist of a source object and other references to this object. Copying or deleting an aggregated object also copies or deletes the referenced objects.
- **History Tracking** — Checked, keeps a record of past characteristic values in the History Tracking class when history tracking is enabled for the object class. Characteristics in the class must have the History Tracking status bit set before initializing history tracking (refer to “[Enabling History Tracking](#)” on page 324).

History tracking has these conditions and limitations:

- The software only tracks columns of a list frame characteristic if a line key exists.
- The software cannot track characteristics contained in the te_obj table.
- If te_obj characteristics exist, the software takes the value for the 093user (Modified By) characteristic from the te_obj table.
- **No Copy** — Checked, prevents copying of the characteristic data value when copying an object, and uses the default value instead.



Note:

Do not set the No Copy bit for any obj_id characteristic in any object class that uses versioning. The release and revision control functionality creates new object revisions and predecessor objects by copying the original object. If the No Copy bit is active for the main key of a revised class, release and revision control does not work.

- **Take Over Value** — Checked, permits assigning the data value of another characteristic to this characteristic when creating the object. If the user has edited the characteristic, the edited value is lost and the software copies the original source data.

For example, an administrator enables the Take Over Value bit for the comment characteristic in the Components class and enters ^obj_id^ in the Default Value field inside the **Other** tab. The system now copies the component number into the comment field when a user saves the Component object. Take Over Value lets you enter several

characteristic names or a prefix/suffix string in the Default Value field; for example: COMPANY.^characteristic1^.^characteristic2^.DESIGN

- **Expression (Formula)** — Checked, enables the system to use a formula to calculate the characteristic value. The formula must be specified as the Default Value on the **Other** tab.

For more information about expressions or calculating data values, refer to “[Creating a Characteristic With a Value That is an Arithmetic Function](#)” on page 212.

- **HTML Visible** — No longer used. The default value is unchecked.
- **Outer Join** — Checked, joins two tables of the same class to produce valid search results by ignoring empty ID columns of the joined table. Outer Join is used only used in search actions.

Once a developer defines an object class and enters data values, they might create additional tables for the class (for example, tables to support list frame characteristics). At first, these tables are empty and contain no data. If a user activates one of the characteristics of the table in a search (for example, a list column), the result is zero matches because the ID of the filled table is joined with the ID of the empty table. The Outer Join functionality ignores empty ID columns causing the user to still see objects in the search results list.

You should activate Outer Join when creating a new list frame characteristic within a database that already contains objects. You must set Outer Join for the list frame characteristic itself (that is, the key characteristic of the list and not for the list columns).

- **Ref. View** — Checked, makes the characteristic visible inside the **Search Ref.** tab of an object with dynamic references to another class. Setting Ref. View only has an effect if the Object Reference status bit is also checked.



Note:

Before setting the Ref. View bit, dynamic reference search must be properly configured.

- **Mapping Obj, Mapping Key, Map Portion Key, Mapping Hierarchy, and View in Mapping** — These bits configure mapping in EDM Library. An example of configurable mapping can be found in the **Mapping** tab of a Mapping object. Each of these bits enables the following:

- **Mapping Obj** — Checked, places the characteristic in the list on the left of the mapping or as a Mapping object. Unchecked, specifies that this characteristic is not a Mapping object.
 - **Mapping Key** — Checked, specifies the characteristic is a mapping pin.
 - **Map Portion Key** — Checked, marks the characteristic as a portion of the mapping.
 - **Mapping Hierarchy** — Checked, specifies that this is the mapping hierarchy list on the right side of a mapping. Unchecked, specifies that the list is a standard list.
 - **View in Mapping** — Checked, specifies that the characteristic is viewed in the mapping. Unchecked, does not display the characteristic in the mapping.
- **File Selection Box** — Checked, displays a file open button to the right of the characteristic that, when clicked, displays a file selection dialog box. The File Selection Box function does not work when the Object Reference bit is checked.



Note:

In Windows, a user can only use the file selection box to select filenames. In Linux, a user can also use the file selection box to select a directory path.

- **SubGraphic-List** — This bit is no longer used by the software or data model, but should remain unchecked for new characteristics.
- **Case insensitive search** — Checked, specifies that case-insensitive searching is active (the default). Unchecked, returns only a search result that exactly matches the case of the search criteria. You can enable case-insensitive searching only for characteristics whose value is a string.
- **Mandatory Field (check always)** — Checked, specifies that a data value must be assigned to the characteristic before saving the object.

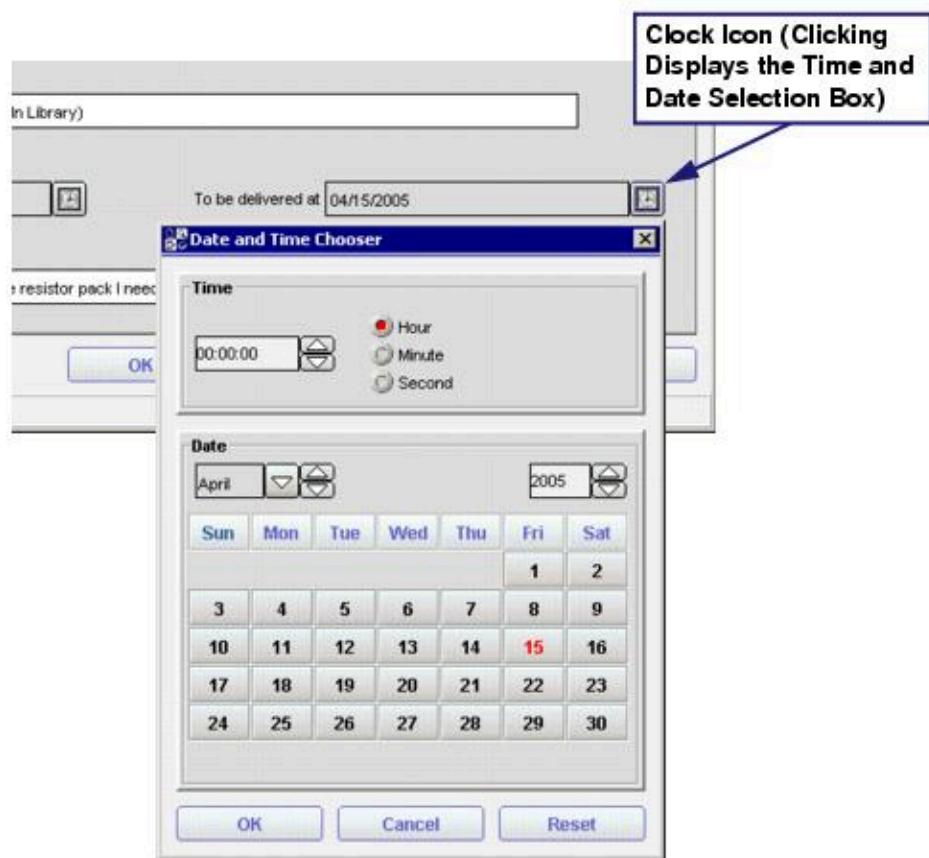
The system only checks the mandatory field if the object class defines a status characteristic (for example, 001obj_statu). Different from the Mandatory Field (check released) status bit, the software always checks for a characteristic data value when saving the object (assuming, of course, that the class status characteristic also exists), regardless of whether the object is in a release status. If no data value is assigned, a user cannot save the object.

With Mandatory Field (check always) set, the characteristic name shows as red in the EDM Library Cockpit user interface, alerting the user that it is a mandatory characteristic that requires a value.

If a characteristic defines an option list, and the characteristic has the Mandatory Field (check always) status bit set, the system also checks if the specified data value is contained in the set of defined option list values. In this case, it is not possible for a user to enter data values different from the option list values.

- **Time/Date Selection Box** — Checked, displays a clock icon next to the characteristic (Figure 94). Clicking the clock icon displays a time and date selection box to use to enter time and date values.

Figure 94. Time and Date Selection Box



Enabling the Time/Date Selection Box bit ensures that a user always enters time and date information in the correct format. The EDM Library software retrieves the correct format of the time/date value from either the default characteristic value, if there is one, or from the `NLS_DATE_FORMAT` environment variable setting.

- **Common Object Reference** — Checked, sets up a common reference to objects of a versioned class for standard or list column characteristics.
For more information about a common object reference characteristic, refer to "[Reference Characteristics](#)" on page 138.
- **Set System Uid** — Checked, automatically generates a UUID (Universal Unique Identifier) for a characteristic. This status bit only applies to list frame characteristics that have the Line key status flag set. The user does not have to enter a line key manually in a list if Set System Uid is set for the corresponding Line Key characteristic.
- **Don't Compare** — Checked, exempts the characteristic from a compare operation (refer to "Comparing Two or More Objects" in the *EDM Library Overview*).
- **Custom List Frame** — Checked on a list frame characteristic permits a plug-in application to populate that list frame column with information from an external source. For more information about using custom list frame functionality, refer to "Custom List Frame" in the *Xpedition EDM Library Guide for API Developers*.

- **Force Field Security** — Checked, forces an administrator to define specific view or edit rights for users and groups on the [Characteristic Object - Rights Tab](#), and forces other EDM Library programs to adhere to those rights when viewing or editing characteristic data. For example, enabling Force Field Security and then specifying that user lindsey is the only account with view rights on the **Rights** tab means that only lindsey can view the characteristic data in the EDM Library Cockpit program, and also is the only user that can extract the characteristic data from the database with a loader program.

Enabling Force Field Security but not defining any rights on the **Rights** tab prevents any users or groups who use the EDM Library Cockpit interface or other EDM Library programs from having access rights to the characteristic data.

The default is not to enable Force Field Security, which still permits other EDM Library programs view access to the characteristic data even though some users might not have the rights to view the data in EDM Library Cockpit.

Related Topics

[Characteristic Object - Top Tab](#)

[Characteristic Object - Table Tab](#)

[Characteristic Object - Status Tab](#)

[Characteristic Object - Rights Tab](#)

[Characteristic Object - Other Tab](#)

[Characteristic Object - History Tab](#)

[Reference Characteristics](#)

[Creating a Standard Characteristic](#)

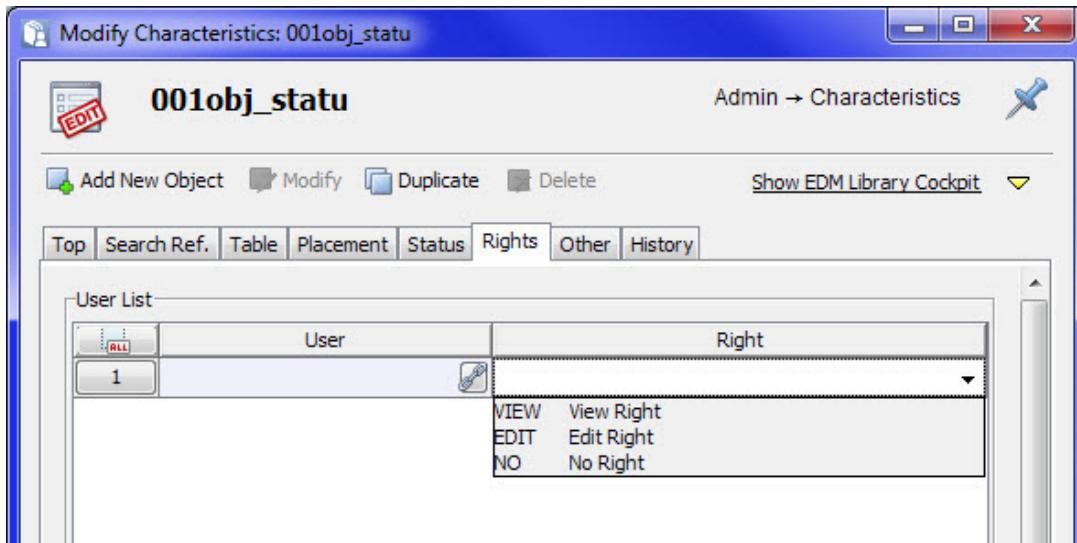
Characteristic Object - Rights Tab

To access: Refer to “[Opening a Characteristic Information Window](#)” on page 146.

The **Rights** tab of a Characteristic information window manages access permissions of individual users to the characteristic.

Description

Figure 95. Rights Tab of a Characteristic



Rows in the User List table define the view or edit rights of user or user groups to the characteristic. If the User List is empty, the characteristic inherits the class rights of the user role assigned to the user, and any rights defined in the parent catalog. You cannot grant greater rights at the characteristic level than permitted at the class or catalog group level. For example, if the catalog group containing the characteristic specifies that lindseyt has view rights only, you cannot grant lindseyt edit rights to a characteristic in that catalog group. Once you define edit rights for one user on the **Rights** tab, you must explicitly define edit rights for any other users, or those users cannot edit the characteristic.

To make the rights on the **Rights** tab effective, you must uncheck the Edit Allowed flag on the [Characteristic Object - Status Tab](#). If Edit Allowed is checked, the characteristic ignores user-dependent access rights from the **Rights** tab and remains editable. In addition, checking the Force Field Security flag on the **Status** tab adds security by requiring that you then specify specific user or group permissions on the **Rights** tab to permit other EDM Library programs to view the characteristic data.

For list frame characteristics, View, Edit or No rights can also be specified for the list columns. Specifying a View right for a list frame characteristic prevents adding rows or editing column values in the list. Specifying a No right for a list frame characteristic makes the list invisible for the corresponding user.

To make a single characteristic visible for a special user group, uncheck **Display Input Mask** on the **Status** tab and set the Edit or View right for the user group.

Characteristics

Each row in the User List contains the following two columns:

- **User** — The name of a user or user group. Using the reference button in the User column displays a **User** tab in the search window with which to search for user names or user groups.
- **Right** — One of three user permissions:

- **View Right** — The user or group cannot edit the characteristic value. The characteristic shows up as a display field, making it only possible to view the characteristic information.
- **Edit Right** — The user or group has full edit permission and can edit the characteristic value.
- **No Right** — The search window or information window does not include the characteristic for that user.

Related Topics

[Characteristic Object - Top Tab](#)

[Characteristic Object - Search Ref Tab](#)

[Characteristic Object - Table Tab](#)

[Characteristic Object - Placement Tab](#)

[Characteristic Object - Status Tab](#)

[Characteristic Object - Other Tab](#)

[Characteristic Object - History Tab](#)

[Creating a Standard Characteristic](#)

[Setting User-Based or Group-Based Characteristic Editing Permissions](#)

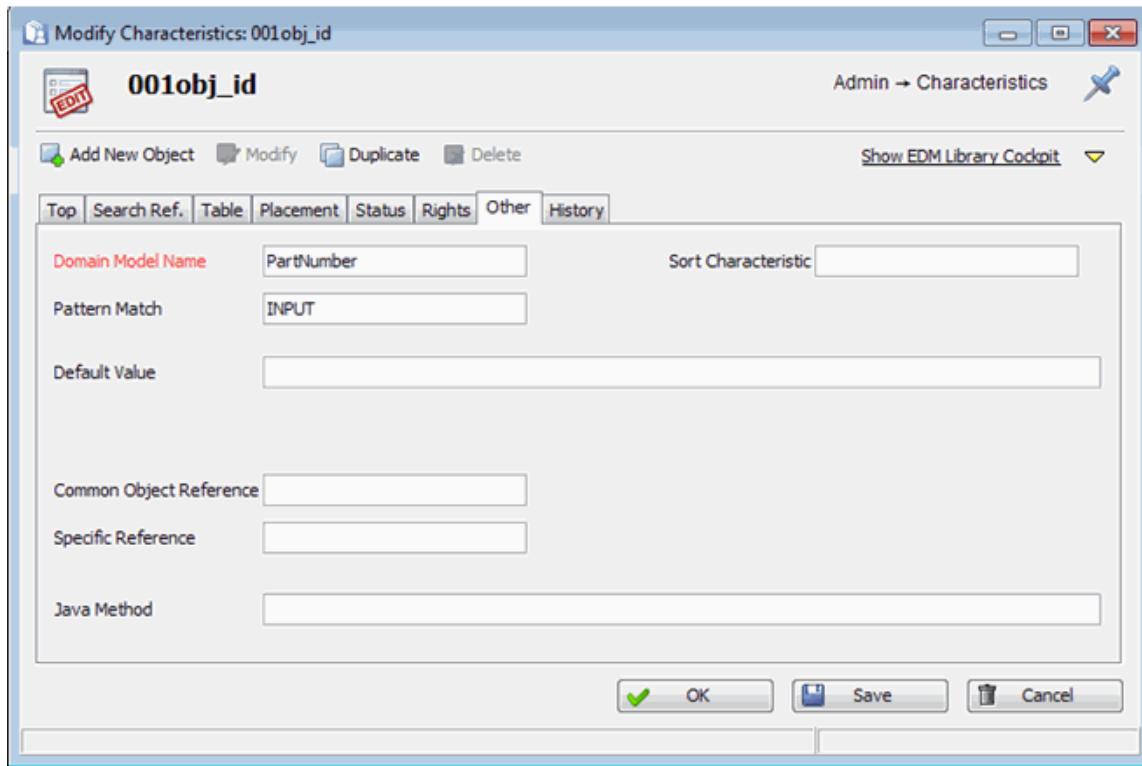
Characteristic Object - Other Tab

To access: Refer to “[Opening a Characteristic Information Window](#)” on page 146.

The **Other** tab of a Characteristic information window contains additional data about the characteristic.

Description

Figure 96. Other Tab of a Characteristic



Characteristics

- **Domain Model Name** — A required value used by an API client program to identify the characteristic when attempting to determine a path through the data model (refer to “[PathQuery Dialog Box - Path Builder Tab](#)” on page 359). Unlike the value of Domain Model Name on class objects and catalog objects, the value of Domain Model Name on a characteristic does not have to be unique across the entire data model, but only within a given object class.

To check if a Domain Model Name value is unique, use the **domain_model_checker** command located in the <SDD_HOME>/dms/bin directory. The **domain_model_checker** command checks for characteristics without a domain name and characteristics with the same domain name in the same object class and returns a message if either condition occurs.



CAUTION:

Do not change the value of Domain Model Name in existing characteristics that are part of the default data model because it can break functionality. Changing the Domain Model Name of custom characteristics could also break any custom API program that accesses data using a path with a Domain Model Name.

- **Pattern Match** — Contains the Input pattern of the characteristic and is used for input verification. Pattern Match describes the input type of the characteristic. The selected input key must exist in the Units object class (for characteristics with a value type of Double) or Input

Pattern Check object class (for characteristics with a value type of CHAR). For example, for units the value of Pattern Match might be A, V, Farad, or Henry. For an input pattern check for the date, the value might be Date.

- **Sort Characteristic** — Applies only to characteristics with an option list. Use Sort Characteristic to override the default sort algorithm that sorts characteristic option values in the search results list in alphabetic order. Type the value of a reference characteristic in the Sort Characteristic field that has the same option list values as the source characteristic, but with different values in the Sort column of its option list.



Note:

The Sort Characteristic field is best used when creating a new object class with new characteristics and before you have loaded data.

As an example of how to use Sort Characteristic, the 001obj_statu (Status) characteristic has an option list with the Sort values in [Figure 97](#) defined in its **Top** tab.

Figure 97. Sort Definitions for 001obj_statu

Option List						
	Catalog Group	Option	Text	Sort	Ref. Characteristic	Reference Value
1		A	Approved/Released	3		
2		D	In Development	2		
3		R	Restricted	4		
4		U	Under Construction	1		
5		X	Obsolete	5		

To specify the sort order of data values inside the search results list of class number 1 (Components), you can assign a Sort Order value of 1 to the 001obj_statu characteristic (**Placement** tab). The search results list of components now sorts alphabetically by the Status value when you do a search (all components with Status value A, then all components with Status value D, and so on). If this is an adequate sort order, there is no need to use the Sort Characteristic field.

Otherwise, copy the 001obj_statu characteristic (after you have assigned a Sort Order value of 1 on the **Placement** tab) to create another characteristic named 001statusort with the same attributes and option list values, but with different Sort values in the **Top** tab ([Figure 98](#)) and with only the Input Characteristic bit active in the **Status** tab.

Figure 98. Sort Definitions for 001statusort

Option List						
	Catalog Group	Option	Text	Sort	Ref. Characteristic	Reference Value
1		A	Approved/Released	1		
2		D	In Development	3		
3		R	Restricted	2		
4		U	Under Construction	4		
5		X	Obsolete	5		

Typing 001statusort in the Sort Characteristic input field of the **Other** tab of the 001obj_statu characteristic and refreshing the database now results in the sort order in the search results list defined by the 001statusort characteristic.

- **Default Value** — Specifies a default value to assign when a user creates a new object. The following are acceptable entries:

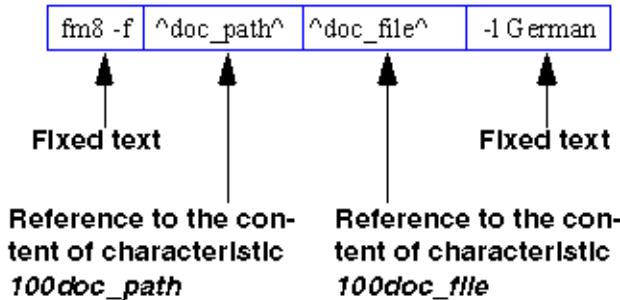
- Any characters or numbers as a standard default value.
- An individual date format for characteristics of the DATE type, overriding the date format setting specified in the Preferences dialog box using **Edit > Preferences**.
- A program call for an action characteristic.

The program call string can reference other characteristic names that hand over the contents of the specified characteristics to the application called by the action button.

To reference another characteristic, use the format ^<characteristic name>^. Omit the prefixed class number in the characteristic name. Execute a program call by typing the program name in the format progrname, without caret symbols.

The example in [Figure 99](#) calls a FrameMaker document specified by the characteristics 100doc_path and 100doc_file:

Figure 99. Example of a Program Call



EDM Library Cockpit always executes commands in asynchronous mode, which means that the application is never blocked and always continues to run in parallel to the command call.

Note that an entry in the Java Method field on the **Other** tab can override a program call specified in the Default Value field.

- The name of another characteristic, which automatically takes over the value from that characteristic. To insert references to other characteristics inside the same object class, use the format ^characteristic_name^ (without the class number prefix). Note that the Take Over Value status bit must be set to take over the value of another characteristic.
- Equations can also be used for characteristics that are controlled by an arithmetic rule. For more information about arithmetic functions, refer to [“Creating a Characteristic With a Value That is an Arithmetic Function”](#) on page 212.



Note:

Do not define the values of the Common Object Reference, Specific Reference, and Frozen Reference characteristics without first understanding how these characteristics are used when implementing major/minor versioning.

- **Common Object Reference and Specific Reference** — Used when implementing Major-/Minor versioning, as follows:
 - A common object reference points from a characteristic of the owner object class to any characteristic of another target object class.
 - A specific object reference always points from a characteristic of the owner object class to the obj_id (that is, the key) characteristic of the target object class, thereby permitting EDM Library to identify the object to open.



Note:

There are four characteristics in the data model that use these fields and that have default values. Do not change these values unless you are creating a customized EDM Library configuration with the assistance of a Siemens representative. It is very important to handle the contents of these three characteristics carefully and only permit your Siemens representative, or someone working with a Siemens representative, to modify their values.

- **Java Method** — Used for system-internal function calls, such as a vfunction call associated with an action button that opens a datasheet.

An entry in the Java Method field overrides the entry in the Default Value field. The software looks first for a value inside the Java Method field and, if found, executes the system-internal function defined by that value. If the Java Method field is empty, EDM Library Cockpit looks for a function call inside the Default Value field and executes that function.

The following line shows an example for a function call in the Java Method field:

```
DFO:com.mentor.dms.ddm.MasterBOM#updateBOMFromDx
```

The calls specified in the Java Method field always run synchronously to the EDM Library Cockpit program, which means that the user interface is blocked until the client method has finished its operation. However, it is also possible to control this behavior through the Java client program. The Java method can immediately return to EDM Library and execute its operation in an own thread so that the EDM Library user interface will run in parallel to the Java method.

Related Topics

[Characteristic Object - Top Tab](#)

[Characteristic Object - Search Ref Tab](#)

[Characteristic Object - Table Tab](#)

[Characteristic Object - Placement Tab](#)

[Characteristic Object - Status Tab](#)

[Characteristic Object - Rights Tab](#)

[Characteristic Object - History Tab](#)

[data_model_checker](#)

[domain_model_checker](#)

[Creating a Standard Characteristic](#)

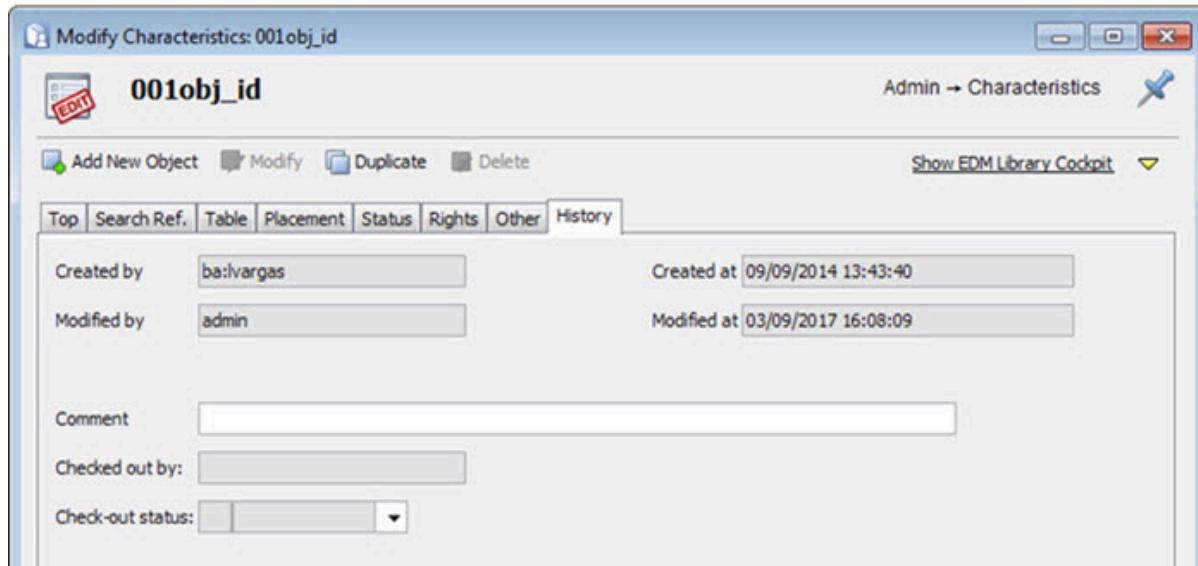
Characteristic Object - History Tab

To access: Refer to “[Opening a Characteristic Information Window](#)” on page 146.

The **History** tab of a Characteristic information window contains system-maintained information about the history of the characteristic. The only field in which you can type information is the Comment characteristic.

Description

Figure 100. History Tab of a Characteristic



Characteristics

- **Created by** — The user who created the characteristic.
- **Modified by** — The user who last modified the characteristic. The system automatically updates this field when a user edits an object.
- **Created at** — The date of creation.
- **Modified at** — The last date that the object was modified.
- **Comment** — Space to type a comment relating to the history.

- **Checked out by** — The username of the person that has the object checked out. Checking out an object prevents others from editing that object.
- **Check-out status** — A flag that, when set to “1” indicates the object is checked out and cannot be edited by a user other than the one identified by the Checked out by characteristic.

Related Topics

[Characteristic Object - Top Tab](#)

[Characteristic Object - Search Ref Tab](#)

[Characteristic Object - Table Tab](#)

[Characteristic Object - Placement Tab](#)

[Characteristic Object - Status Tab](#)

[Characteristic Object - Rights Tab](#)

[Characteristic Object - Other Tab](#)

[Creating a Standard Characteristic](#)

Creating a Standard Characteristic

In the characteristic definition, a standard characteristic has the Characteristic Type value set to Standard. A standard characteristic in the user interface often enables the user to type in a value or choose one from an option list. Most characteristics you see on the tabs of an information window are standard characteristics.

The best way to show how to create a standard characteristic is by an example that you can then adapt to your specific data needs. The example in this procedure explains how to create a new class characteristic inside the **Top** tab of the Components object class that holds a text string. The new characteristic is then in every **Top** tab of all catalog groups in the Components object class.

The procedure makes only the changes in a Characteristic information window necessary to quickly create the characteristic. You might want to change a parameter not described in the procedure to make the characteristic behave in a certain way (for example, to restrict the users that can edit the characteristic, to specify an input pattern that typed data must follow, or to pre-populate the characteristic with a default value). Refer to Characteristic object tab descriptions listed in Related Topics for more information about the fields not covered in this procedure.



Note:

Only a qualified administrator who is familiar with the EDM data model, the user community, and how EDM Library is used within his or her company should create characteristics.

In lieu of this procedure, you can also use the Create Characteristic wizard to create standard characteristics in most Comp module object classes. The wizard limits editable parameters to the minimum required to create the characteristic. For more information, refer to “Adding a Static or Dynamic Characteristic” in the *Xpedition EDM Library Guide for Component Engineers*.

Prerequisites

- You invoked the Xpedition EDM Library Cockpit application and connected to an EDM Server using an account with administrative privileges that permits creating new characteristics.
- You have obtained information about the class where you want to assign the characteristic by searching the Object Classes class as described in [“Classes, Class Number, and Class Tables”](#) on page 104. You need the following information:
 - The class number. For example, the class number of the Component class is 1. Add the class number on the **Top** tab of the new characteristic definition.
 - The class table name. For example, the class table name for the Component class is te_sachnr. Add the class table name on the **Table** tab of the new characteristic definition.

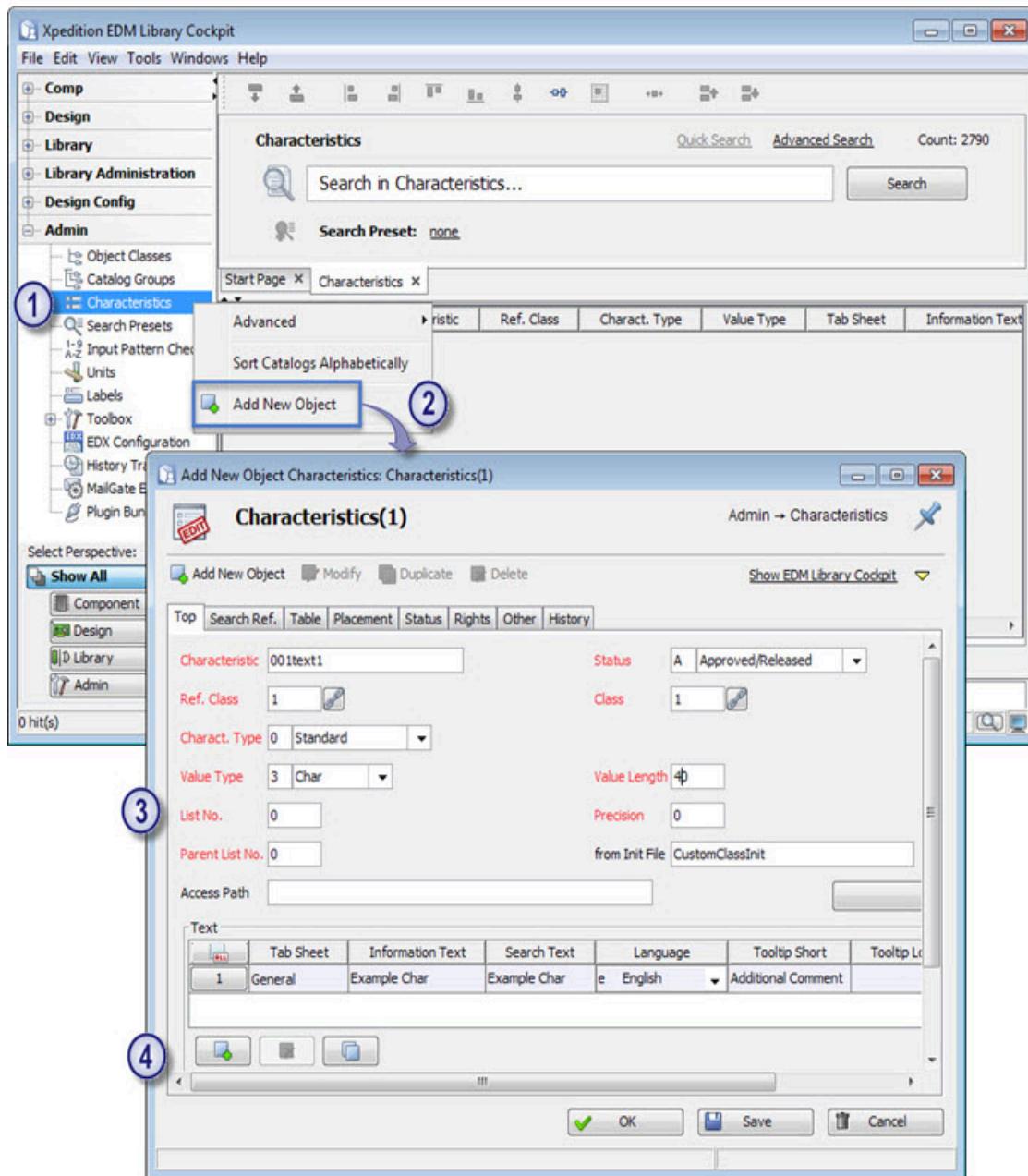
Procedure

1. In the classification pane, open the Admin module and select **Characteristics**.
2. Right-click and then choose **Add New Object** from the classification hierarchy popup menu or the search results popup menu.

Alternately, perform a search, select an existing standard characteristic from the search results list, right-click, and choose the **Duplicate** popup menu item (refer to [“Opening a Characteristic Information Window by Searching the Characteristics Object Class”](#) on page 150).

By copying an existing characteristic, the tabs already contain data. You then only need to modify the data that is different.

Figure 101. Creating a Characteristic



3. Enter data or verify values on the various tabs of the characteristic definition, as follows:

a. **Top** tab ([Figure 102](#))

- Type a name in the Characteristic field (for example, 001text1). The prefix 001 indicates that the characteristic belongs to object class 1 (Components).



Note:

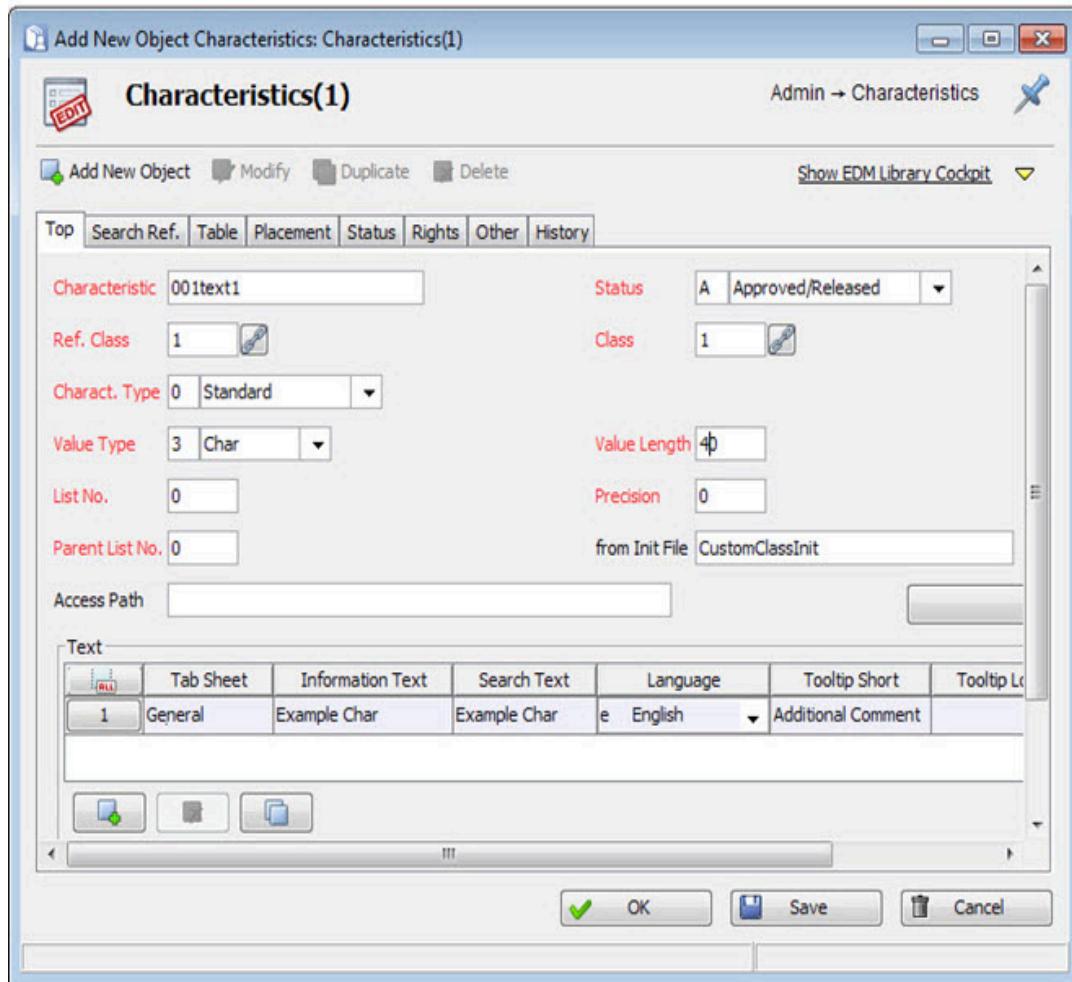
A characteristic name must not begin with the three characters 'smw', as these characters have special meaning to the EDM Library software.

- Type a class number in the Ref. Class and Class field. To create a characteristic to attach to the Components class, type 1.
- In the Characteristic Type field, choose a value of Standard.
- In the Value Type field, specify if the characteristic value holds a number or a string. For a characteristic that holds textual data, choose Char.
- The entries in List-No. and Parent List No. are only applicable to list characteristics (refer to ["Creating a List \(List Frame Characteristics and List Columns\)"](#) on page 226). Because this example does not create a list, type 0 in both fields.
- Status makes the characteristic visible or invisible. A value of U does not display the characteristic in EDM Library Cockpit or other EDM Library applications. Because you want to display the characteristic in this example, choose A (Approved/Released).
- Value Length specifies the number of legal characters permitted in the characteristic data value. In this example, type 40.
- If the Value Type is Double, you can use the Precision field to specify the number of digits permitted to the right of the decimal point. For other value types, the value of Precision must be 0 (the default).

Because this example does not create a characteristic with a value type of Double, type 0 in the Precision field.

- In the Text list, type the name of the tab to hold the new characteristic (for example, General). The Information Text and Search Text are the characteristic labels that display in the Search and Information window (in [Figure 102](#) the value is "Example Char"). The Tooltip Short displays at the mouse cursor when moving the mouse over the characteristic label. To show the tooltip feature, type "Additional Comment" for the short tooltip, and "Please enter an additional comment" for the long tooltip.
- The Option List contains enumerated values defined for a standard characteristic. For a Text Frame characteristic, leave this table blank.

Figure 102. Completed Top Tab (Standard Characteristic Example)



b. Table tab (Figure 103)

- In the Table Name field, type the name of the table that holds the characteristics. For this example, type te_sachnr.



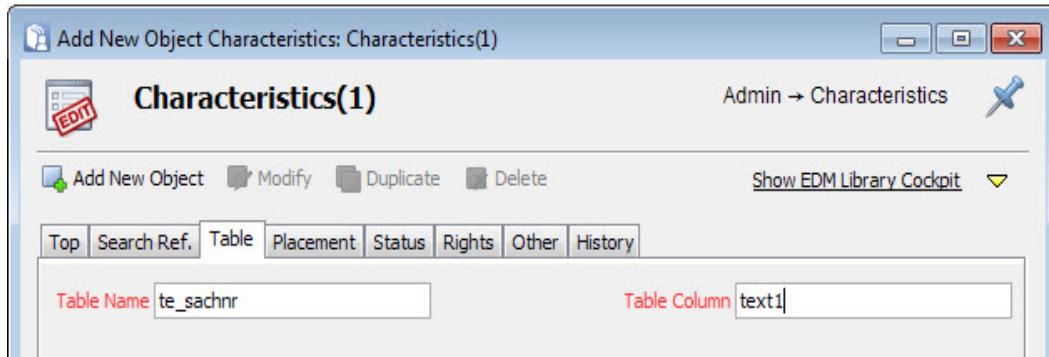
Note:

A table column name cannot be a reserved keyword in SQL (for example, SELECT, UPDATE, and so on). For a list of reserved words in SQL, refer to your database documentation.

The combination of the Table Name and Table Column should not exceed 30 characters for a searchable characteristic. This restriction is necessary because, when searching, Oracle forms the index names by concatenating <table_name>_<column_name> and only permits a maximum of 30 characters per index. If the resulting Oracle index exceeds 30 characters, Oracle can issue an ORA-00972 error.

- In the Table Column field, type the name of a column within the table you want to hold the characteristic values. In this example, text1.

Figure 103. Completed Table Tab (Standard Characteristic Example)



c. **Placement** tab ([Figure 104](#))

The **Placement** tab defines the position and size of the characteristic. Specifying X=0 and Y=0 automatically places the characteristic. A list characteristic is an exception to this rule.

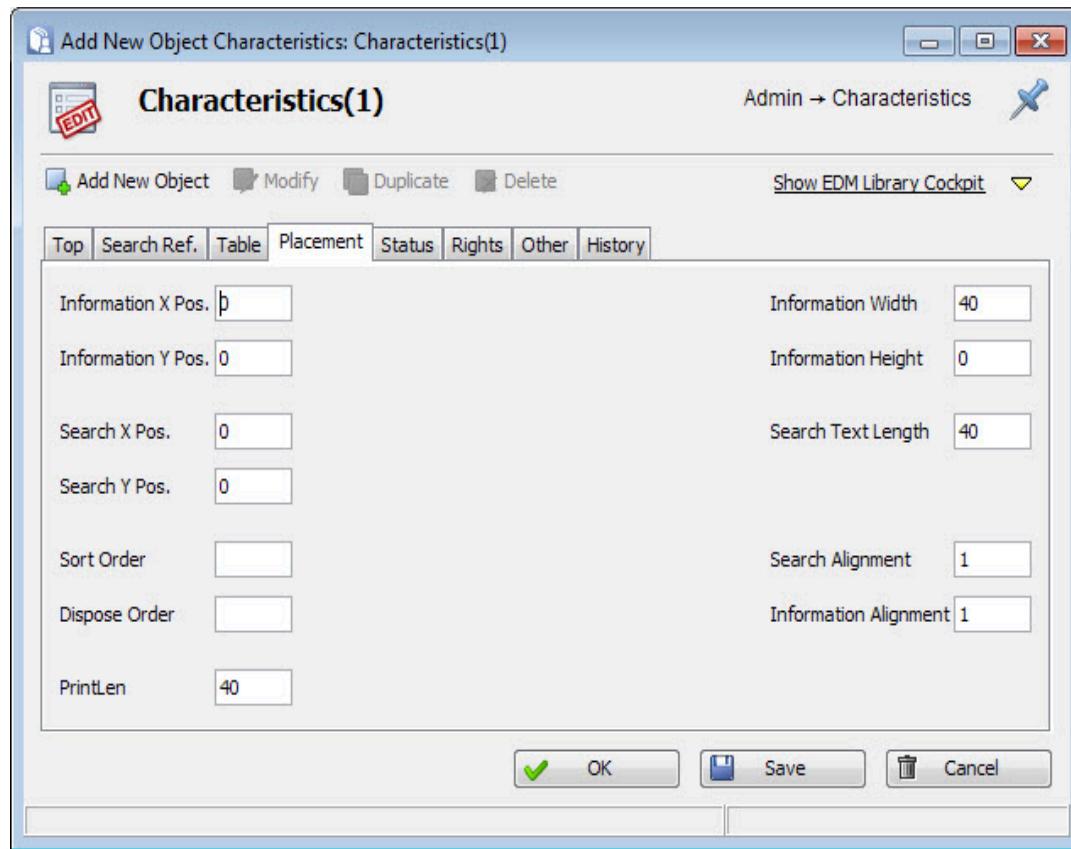


Note:

A characteristic might not be visible if the placement values cause an overlap with another characteristic.

Use the values shown in [Figure 104](#) for the **Placement** tab to position the example characteristic.

Figure 104. Placement Tab (Standard Characteristic Example)



d. **Status** tab (Figure 105)

Correct entries in the **Status** tab are extremely important for the proper display and function. In this example, check only the following status bits (Figure 105):

- Edit
- Search Characteristic
- Input Characteristic
- Display Input Mask
- Display Search Mask

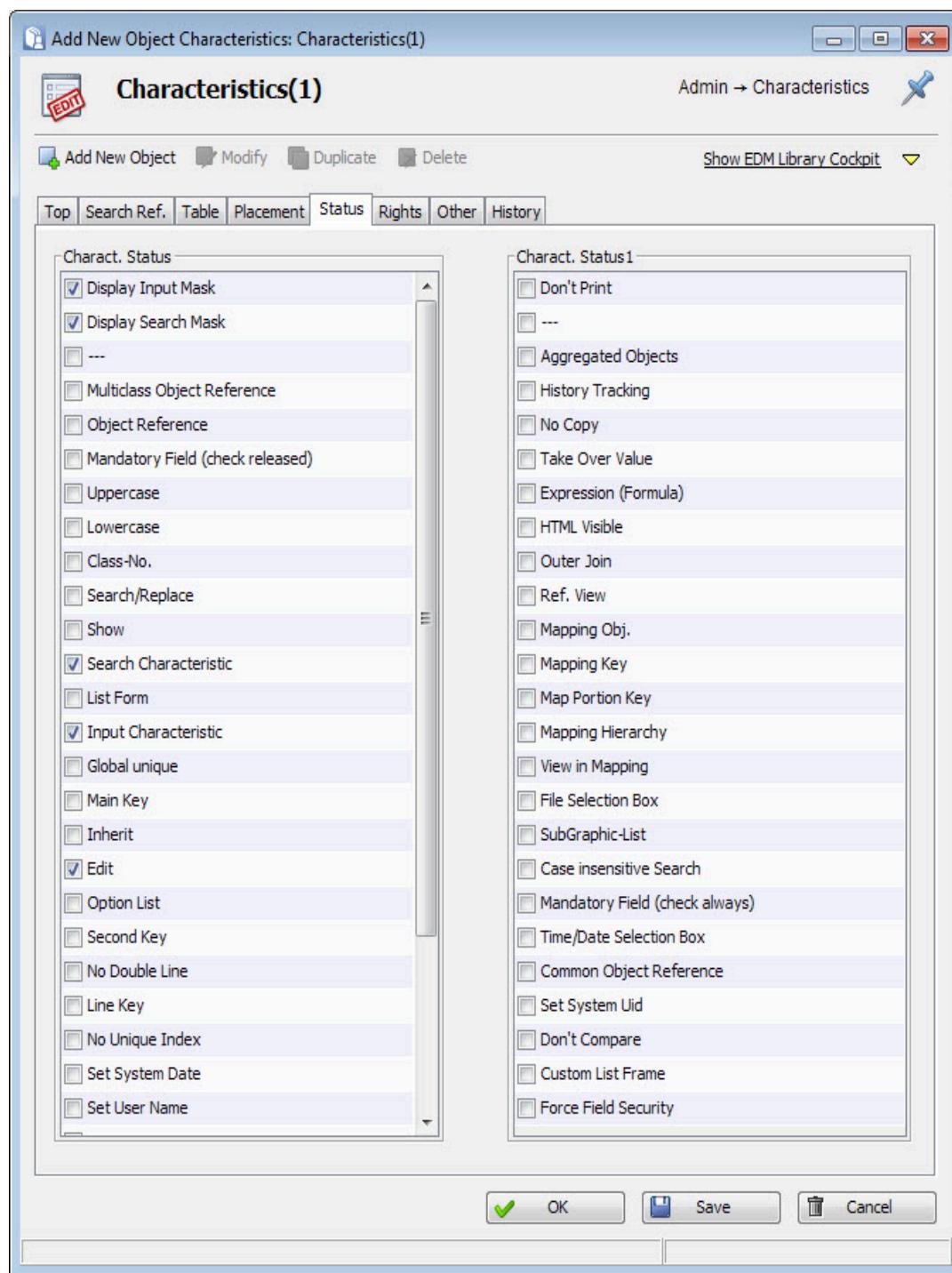


Note:

List characteristics require different default settings.

These are also the default settings.

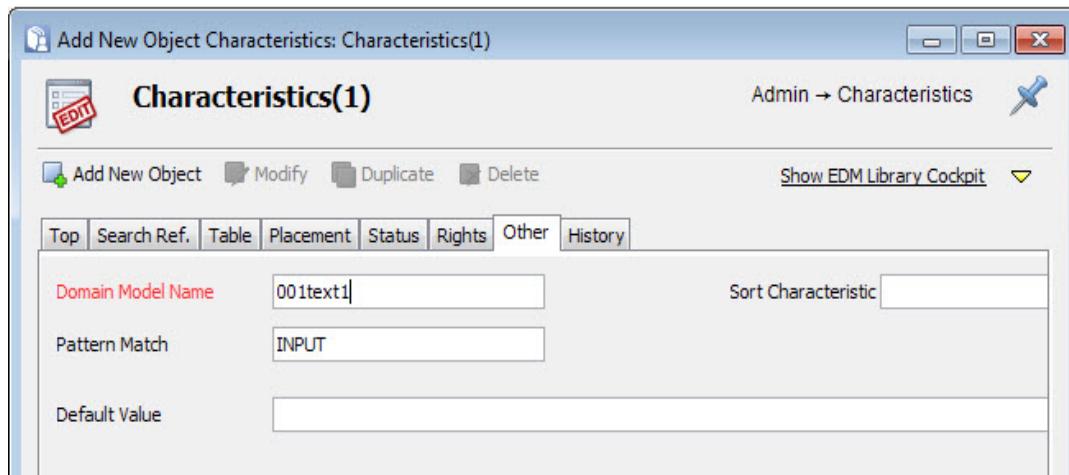
Figure 105. Status Tab Default Settings for a Characteristic



e. Other tab (Figure 106)

Assign a unique Domain Model Name value to the new characteristic. Use the name of the characteristic (for example, 001text1) to ensure a unique value.

Figure 106. Other Tab (Standard Characteristic Example)



4. Click the **OK** button to add the new characteristic to the database and to close the information window. The **Save** button creates the new characteristic without closing the information window, enabling you to create several characteristics in succession.



Note:

After creating a new characteristic or changing the placement parameters of an existing characteristic, you must reload the data model cache to see the effect in EDM Library Cockpit.

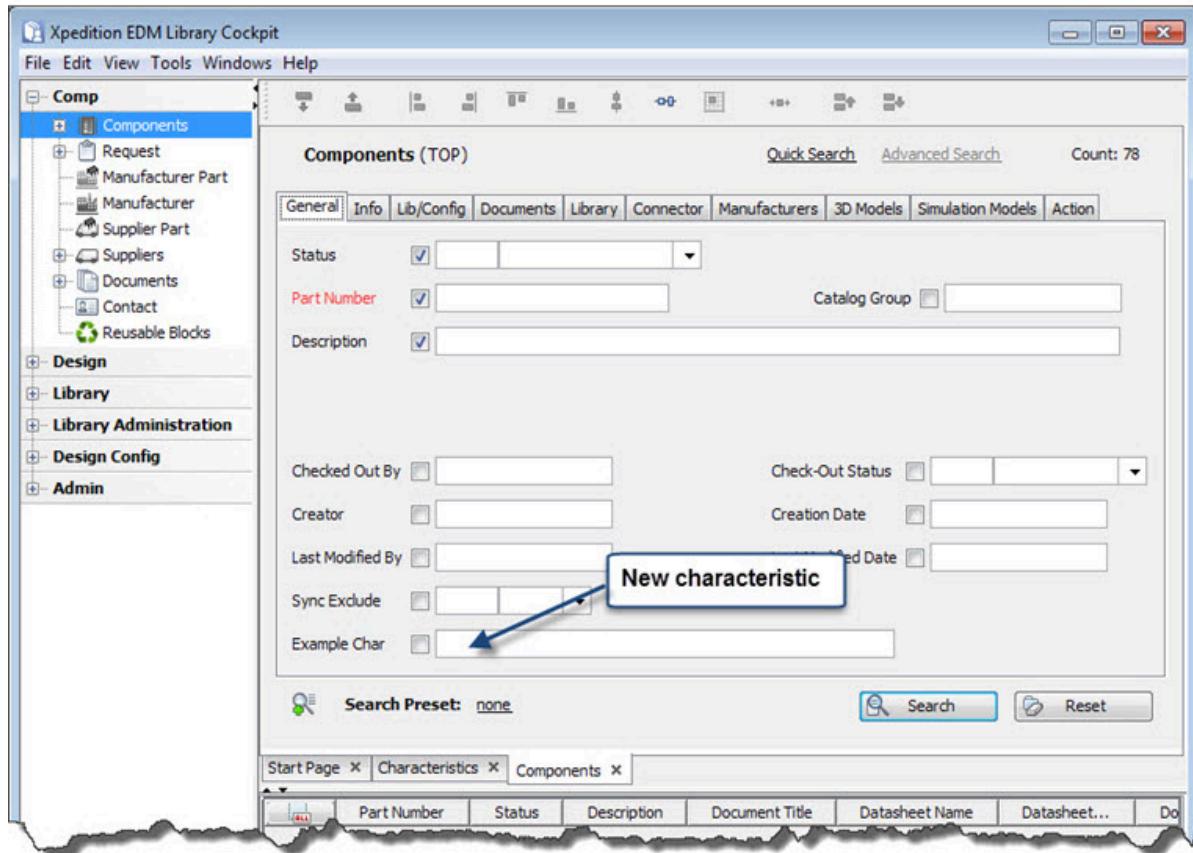
5. Choose **File > Refresh Data Model** or **Tools > Administration > Reload Data Model** to update the database cache with your changes.

Results

The standard characteristic definition is now stored in the database and displays on the component search criteria pane and component information window.

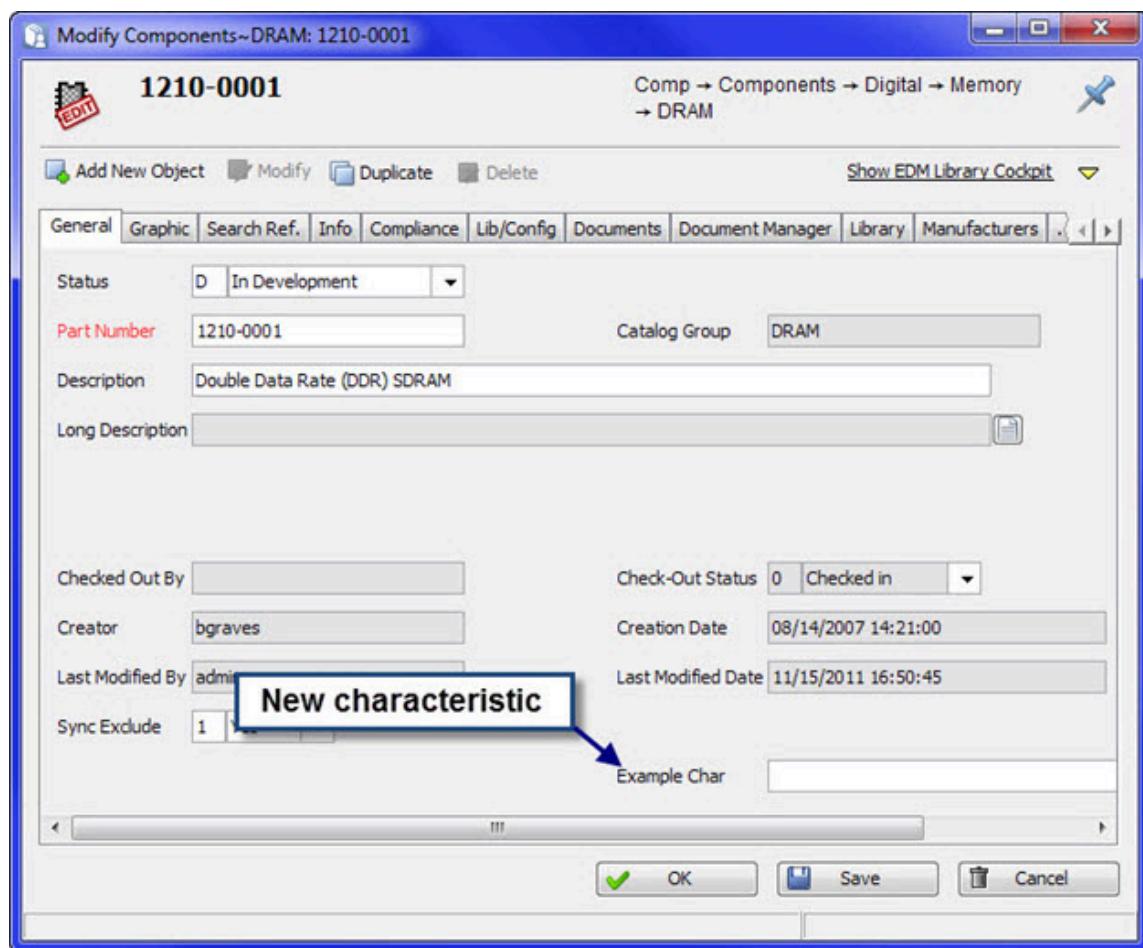
Opening the component search window by choosing Comp > Components in the classification pane now displays the new characteristic (Figure 107).

Figure 107. New Search Characteristic in the Search Window



Opening the component information window shows the new characteristic in the **General** tab [Figure 108](#).

Figure 108. New Characteristic in the Information Window



Related Topics

[Classes, Class Number, and Class Tables](#)

[Class Characteristics and Dynamic Characteristics](#)

[Standard Characteristic Type](#)

[Characteristic Object - Top Tab](#)

[Characteristic Object - Table Tab](#)

[Characteristic Object - Placement Tab](#)

[Characteristic Object - Status Tab](#)

[Characteristic Object - Rights Tab](#)

[Characteristic Object - Other Tab](#)

[Creating a Characteristic to View Data From a Foreign Class](#)

[Creating a Characteristic With a Value That is an Arithmetic Function](#)

[Creating an Action Button Characteristic](#)

[Creating a List \(List Frame Characteristics and List Columns\)](#)

Creating a Characteristic to View Data From a Foreign Class

You can create a search characteristic in one object class that searches for a characteristic value from a different (foreign) object class.



Note:

Only a qualified administrator who is familiar with the data model, the user community, and how EDM Library is used within his or her company should create characteristics.

The example in this task creates a new search characteristic in the Components class to search for the supplier type of the component. The Supplier object class already has a Supplier Type (092suptype) characteristic holding data values. Because the EDM data model has a built-in reference between Component objects and Supplier objects, you can create a characteristic in the Components class to view the data from the foreign Supplier object class. As shown in the procedure, creating the new Component class characteristic requires invoking the path selector from the definition of the new characteristic and using it to navigate the path through the database to the referenced (foreign) characteristic containing the source data.

In the Components advanced search pane, the Supplier Type characteristic will be added on the **Source** tab beneath the Supplier Name field. Upon completion, a user will be able to view and search for the Supplier Type in the components advanced search pane (in addition to the Supplier ID, Supplier Name and Supplier Part Number).

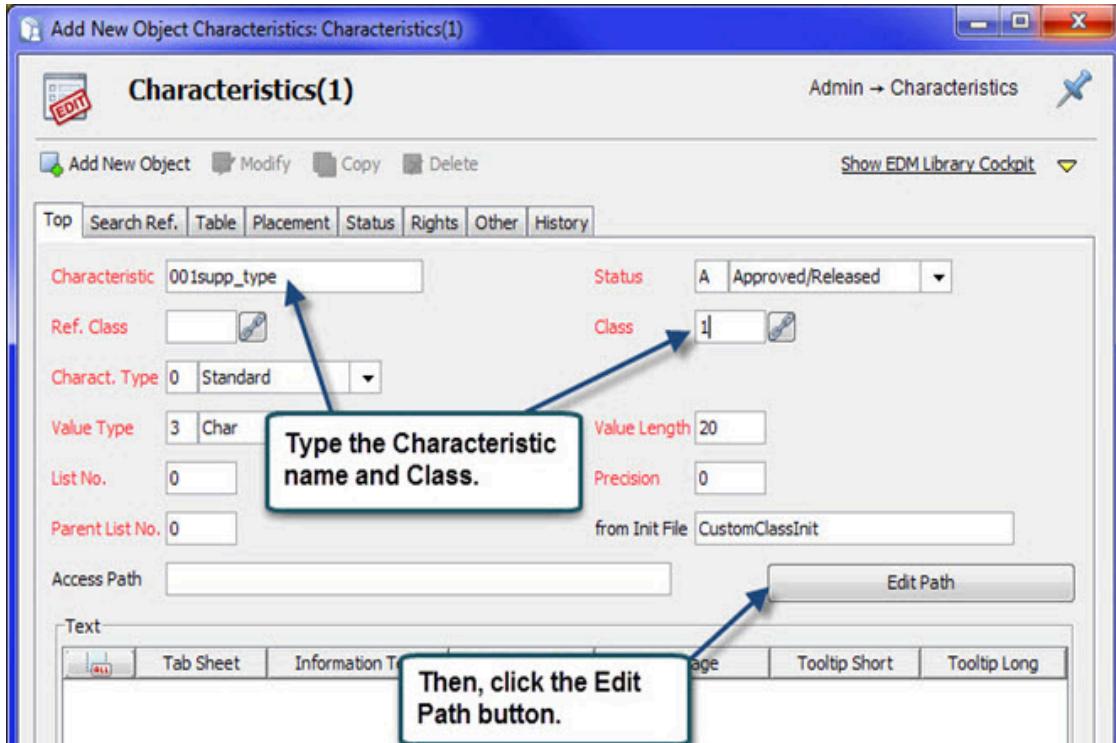
Prerequisites

- You invoked the Xpedition EDM Library Cockpit application and connected to an EDM Server using an account with administrative privileges that permits creating new characteristics.

Procedure

1. In the classification hierarchy pane, open the Admin module and select **Characteristics** to display the Characteristic search window.
2. Right-click and choose **Add New Object** from the search results popup menu.
3. In the Add Characteristic information window ([Figure 109](#)), type a unique Characteristic name and the Class number of the target class (in this example, 1 for the Components class) in the **Top** tab.

Figure 109. Top Tab for the Example Supplier Type Characteristic in the Components Class

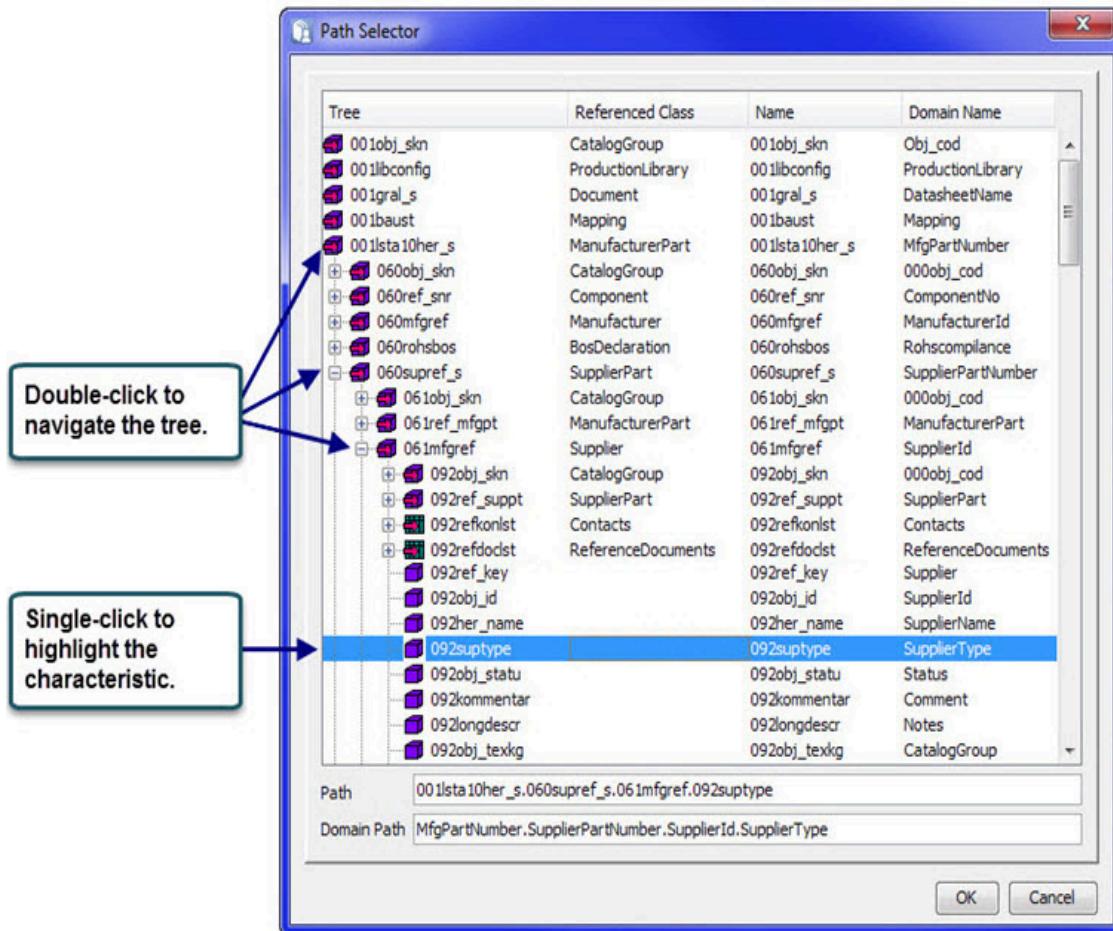


4. Click the **Edit Path** button.

The Path Selector dialog box shows a tree with the paths to all possible reference characteristics accessible through the data model from the source object class (in this example, components). Icons with a red arrow on top of a purple box mark tree branches. A purple box icon marks characteristics available for reference.

5. Double-click the tree branches to specify the search path and then select the target characteristic (Figure 110) to fill in the Path and Domain Path fields of the Path Selector dialog box.

Figure 110. Path Selector Dialog Box Showing the Database Path to the Foreign Supplier Type Characteristic (092suptype)



6. Click OK.

The Path Selector dialog box closes and you are back in the Add Characteristic information window ([Figure 111](#)). The EDM Library software automatically enters values to the target characteristic in the Access Path and the Ref.Class fields on the **Top** tab ([Figure 111](#)) and the Table Name and Table Column of the **Table** tab ([Figure 112](#)).

Figure 111. Auto-Generated Values in the Top Tab of the Example Component Supplier Type Characteristic

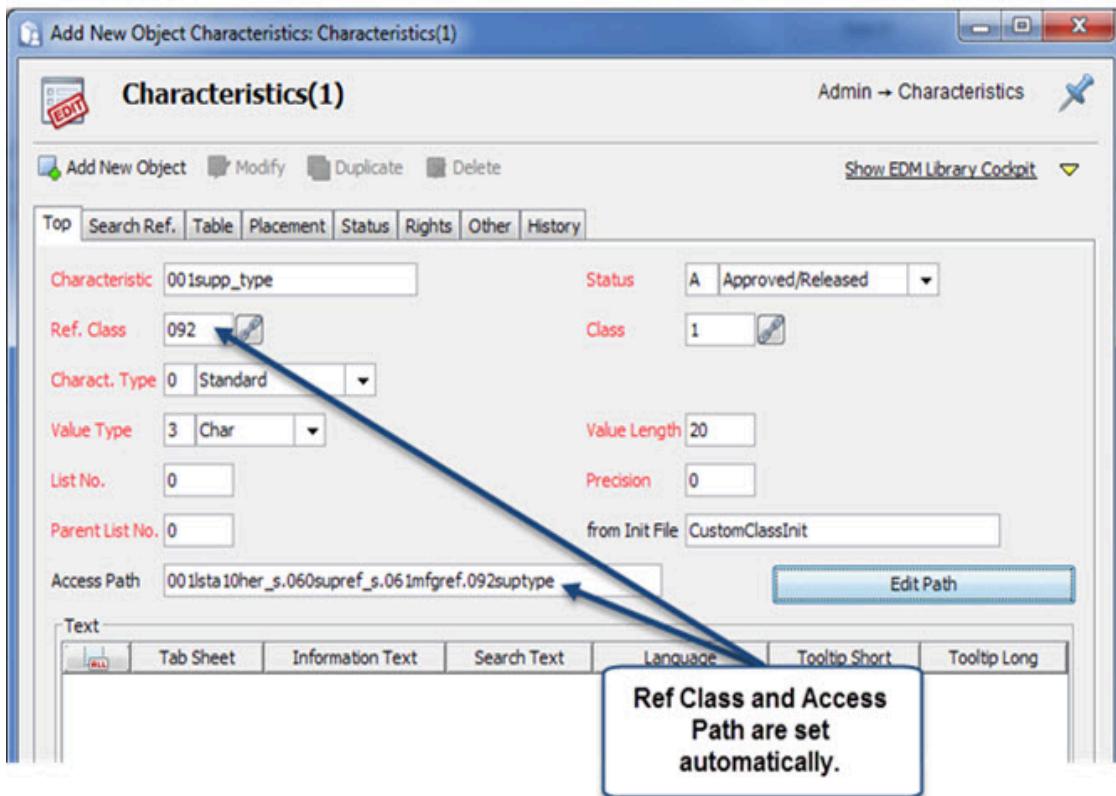
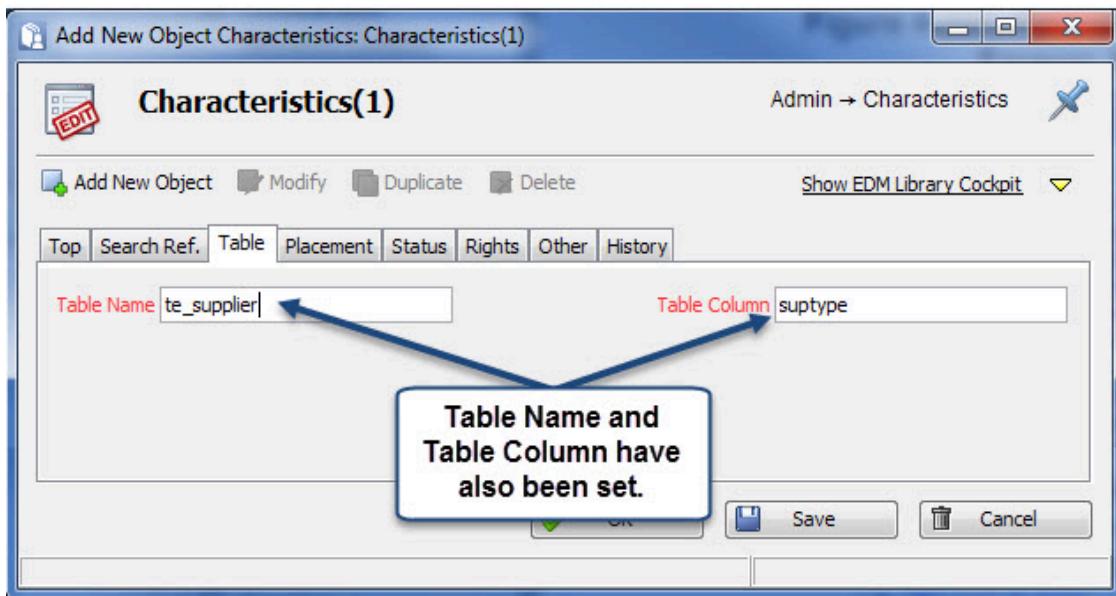
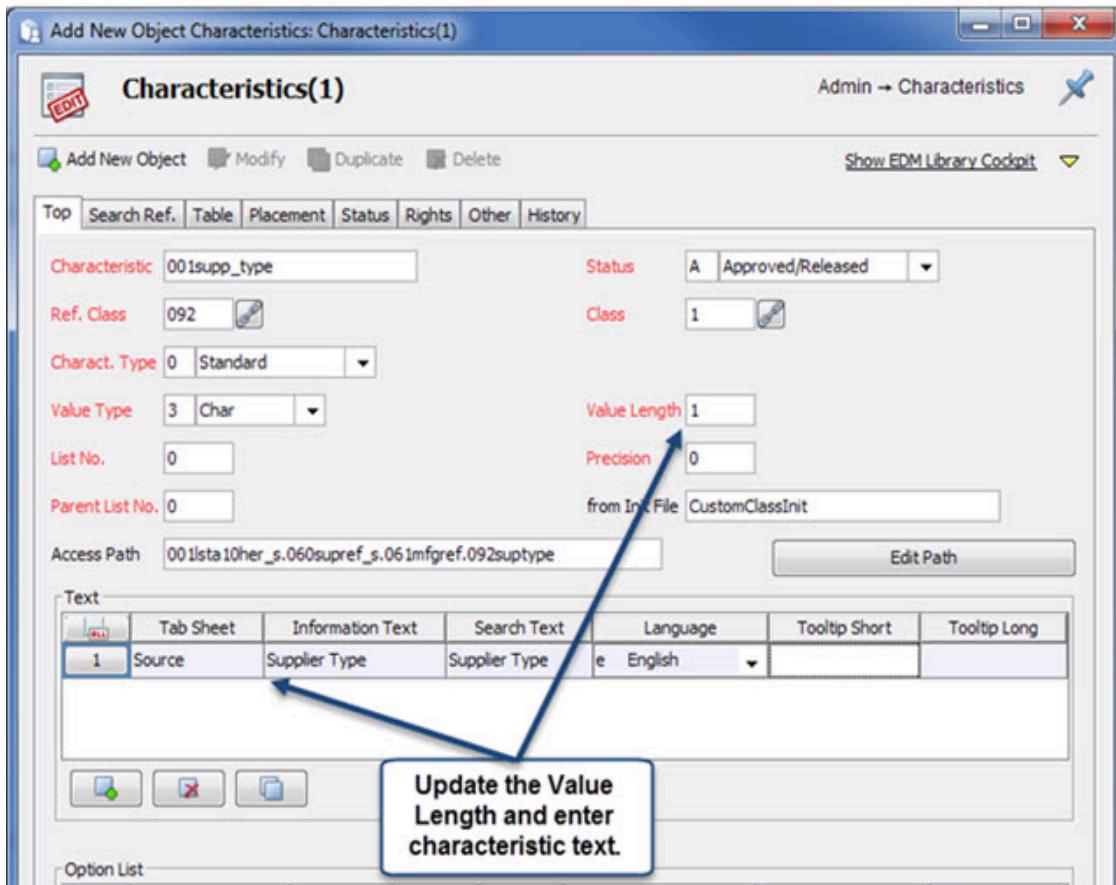


Figure 112. Auto-Generated Values in the Table Tab of the Example Component Supplier Type Characteristic



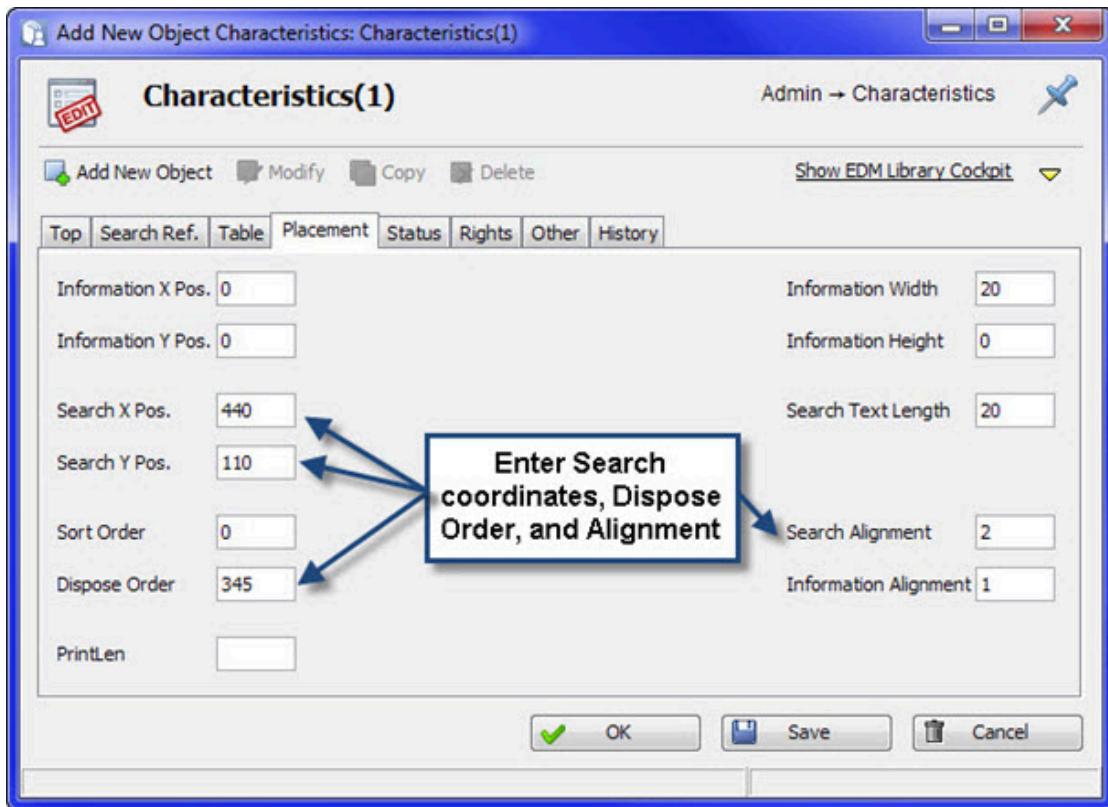
7. Click the **Top** tab, update the value length according to the value length of the target characteristic, and complete the Text list as shown in [Figure 113](#).

Figure 113. Final Changes on the Top Tab of the Example Component Supplier Type Characteristic



8. Click the **Placement** tab and type the values of the search coordinates, the Dispose Order, and the Search Alignment ([Figure 114](#)).

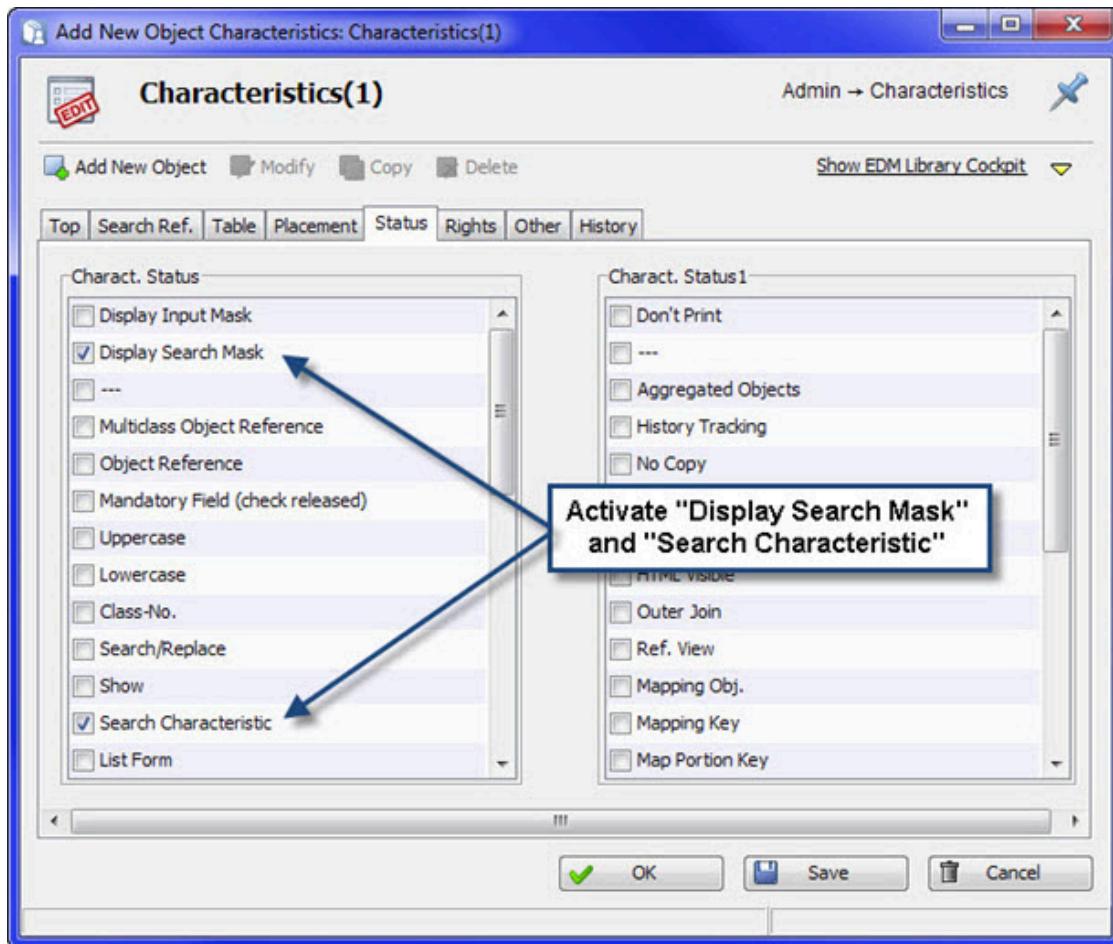
Figure 114. Placement Tab of the Example Component Supplier Type Characteristic



9. Click the **Status** tab and uncheck all status settings except “Display Search Mask” and “Search Characteristic” (Figure 115).

Checking only those two status flags adds the new characteristic to the component advanced search criteria pane, but not to the component information window.

Figure 115. Status Tab of the Example Component Supplier Type Characteristic



10. Click the **Other** tab and type a value for Domain Model Name that is the same as the characteristic name (for example, 001supp_type) to ensure a unique value.
11. Click **OK**.



Note:

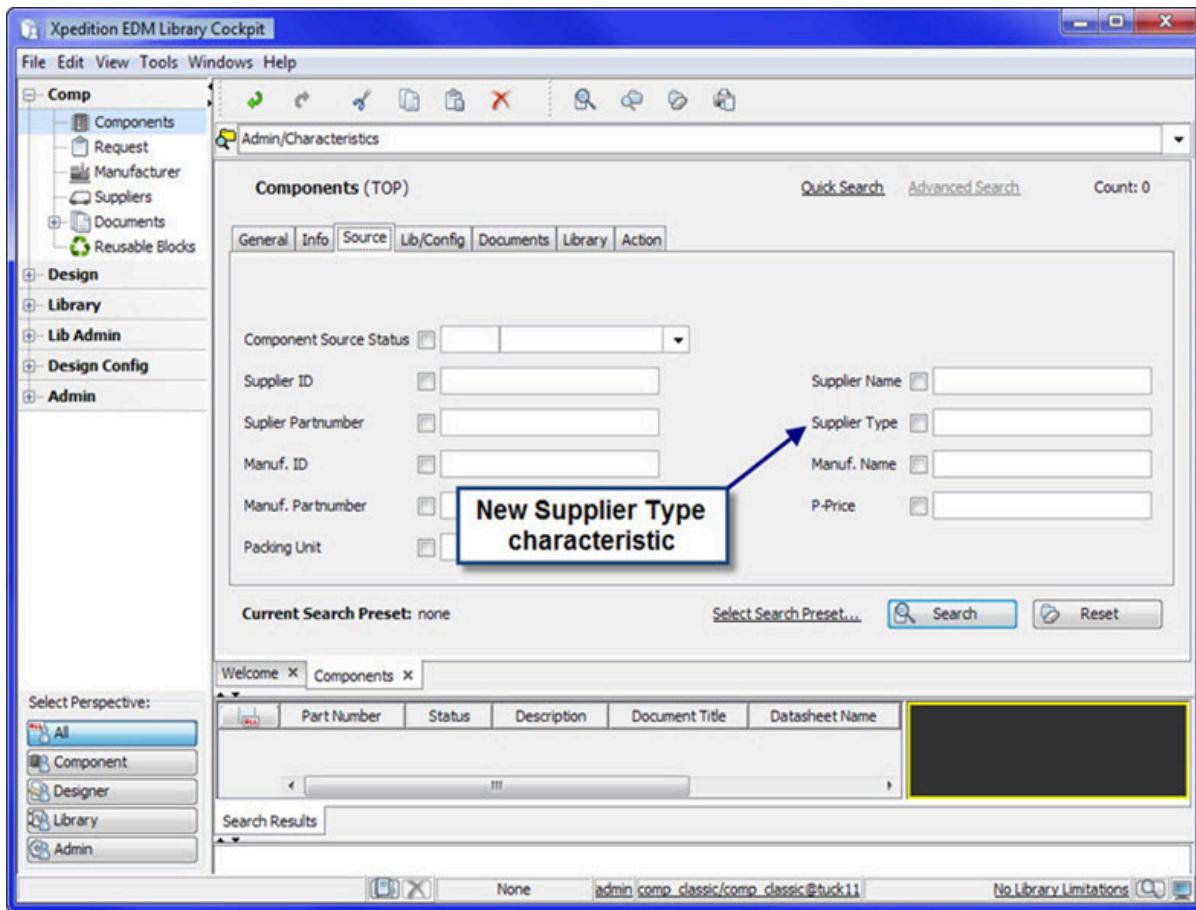
After creating a new characteristic or changing the placement parameters of an existing characteristic, you must reload the data model cache to see the effect.

12. Choose **File > Refresh Data Model** or **Tools > Administration > Reload Data Model** to update the database cache with your changes.

Results

The characteristic definition is now stored in the database and displays on the component advanced search criteria pane (Figure 116).

Figure 116. New Supplier Type Characteristic in Search Window



Related Topics

[Classes, Class Number, and Class Tables](#)

[Characteristic Object - Top Tab](#)

[Characteristic Object - Table Tab](#)

[Characteristic Object - Placement Tab](#)

[Characteristic Object - Status Tab](#)

[Characteristic Object - Rights Tab](#)

[Characteristic Object - Other Tab](#)

[Creating a Standard Characteristic](#)

[Creating a Characteristic With a Value That is an Arithmetic Function](#)

[Creating an Action Button Characteristic](#)

[Creating a List \(List Frame Characteristics and List Columns\)](#)

[PathQuery Dialog Box - Migration Tab](#)

[PathQuery Dialog Box - Path Builder Tab](#)

Creating a Characteristic With a Value That is an Arithmetic Function

You can create a standard characteristic where the characteristic value can consist of the answer to an arithmetic function performed on the values of other characteristics.

Arithmetic functions can use the following operations:

- Addition (+)
- Subtraction (-)
- Multiplication (*)
- Division (/)

EDM Library permits parentheses (with nesting) in an arithmetic function. An arithmetic function as a data value overwrites former data values.

Restrictions and Limitations

- You can only use class characteristics with arithmetic functions. Using a dynamic characteristic in an arithmetic function returns a null value as the result.

Prerequisites

- You invoked the Xpedition EDM Library Cockpit application and connected to an EDM Server using an account with administrative privileges that permits creating new characteristics.

Procedure

1. Create at least two new characteristics (refer to “[Creating a Standard Characteristic](#)” on page 193) or identify at least two existing characteristics with values that can act as input into the formula. Because you are performing an arithmetic operation on the values, the characteristic data type must be an integer or double, and not be a string.

For example, you might create two characteristics named `<class_no>value1` and `<class_no>value2`.

2. Create a third characteristic that performs an arithmetic function on the values of the first two characteristics:
 - a. Copy one of the two characteristics.
 - b. Type a new characteristic name in the **Top** tab (for example, `<class_no>result`), as well as new search and information texts.
 - c. In the **Other** tab, enter the arithmetic function in the Default Value input field. For example:

```
<class_no>value1 + <class_no>value2
```

or

```
<class_no>value1 * <class_no>value2
```

- d. In the **Table** tab, type a new name for the Table Column in which to store the resulting value.
-



CAUTION:

You cannot use any word that is reserved in SQL as a table or column name. For a list of reserved words in SQL, refer to your database documentation.

- e. Type placement coordinates for the new characteristic in the **Placement** tab.
 - f. Check the Expression (Formula) bit in the **Status** tab.
 - g. Type a unique value for the Domain Model Name on the **Other** tab.
 - h. Click **OK** to save the characteristic.
-



Note:

After creating a new characteristic or changing the placement parameters of an existing characteristic, you must reload the data model cache to see the effect in EDM Library Cockpit.

- i. Choose **File > Refresh Data Model** or **Tools > Administration > Reload Data Model** to update the database cache with your changes.
-

Results

EDM Library calculates the value when you save an object whose value is part of the calculation.



Note:

If the input fields of the characteristics already had data before you created the Result characteristic with the arithmetic operation, you must open each object in Modify mode and save it to trigger the result calculation.

Examples

Arithmetic operations in lists offer additional possibilities. The following examples use the following assumptions:

- Two lists, list1 and list2, exist with the following column characteristics:
 - List1: col1list1, col2list1, col3list1 and col4list1.
 - List2: col1list2, col2list2, col3list2 and col4list2.

Table 15. Arithmetic Operations in Lists - List1

Column1	Column2	Column3	Column4
1	0.2	-2	10
2	0.4	-4	2
3	0.6	-3	4
4	0.8	0	6
5	1	2	8

Table 16. Arithmetic Operations in Lists - List2

Column1	Column2	Column3	Column4
0.2	0.4	0.6	0.8
-1	-0.5	-2	-0.25
1	2	3	4
-3	-2	-1	-4
5	4	3	2

- There are two characteristics named charact1 with a value of 10, and charact2 with a value of 5.

Example 1. Calculate the row value of col4 by adding the values of col1 and col2 (in the same row) to the sum of col3 of List1.

$$\text{col4(list1)} = \text{col1} + \text{col2} + \sum \text{col3}$$

The following is the syntax in the Default Value characteristic on the **Other** tab:

```
list1::col1list1 + list1::col2list1 + SUM[list1::col3list1]
```

col4 now contains the following values:

```
-5.8 = (1+0.2+(-2-4-3+0+2))
-4.6 = (2+0.4+(-2-4-3+0+2))
-3.4 = (3+0.6+(-2-4-3+0+2))
-2.2 = (4+0.8+(-2-4-3+0+2))
-1 = (5+1+(-2-4-3+0+2))
```

Example 2. Calculate the value of charact1 by adding the sum of col1, list1 to the product of col1, list2 and then add the value of charact2.

$$\text{charact1} = \sum \text{col1(list1)} + \prod \text{col1(list2)} + \text{charact2}$$

The following is the syntax in the Default Value characteristic on the **Other** tab:

```
SUM[list1::col1list1] + PRODUCT[list2::col1list2] + charact2
```

charact1 is then $23 = ((1+2+3+4+5) + (0.2*(-1)*1*(-3)*5) + 5)$

Example 3. First, calculate the product of col1, list1. Then, subtract the product of charact1 and the corresponding row of col4, list 2. Calculate the final sum and store the result in charact2.

$$\text{charact2} = \sum [\prod [\text{col1(list1)}] - \text{charact1} * \text{col4(list2)}]$$

The following is the syntax in the Default Value characteristic on the **Other** tab:

```
SUM[PRODUCT[list1::col1list1] - charact1 * list2::col4list2]
```

The value is calculated as: $574.5 = (([1*2*3*4*5]-10*0.8)+([1*2*3*4*5]-10*(-0.25))+([1*2*3*4*5]-10*4)+([1*2*3*4*5]-10*(-4))+([1*2*3*4*5]-10*2))$

Example 4. First, calculate the total sum of list1. Then subtract the product of charact1 and the quotient of col1 and col4 of list2 and store the result in col2 of list2.

Two equivalent expressions can be used to make the calculation. The difference is in the summation of list1.

- The first expression adds the sum of all columns.
- The second expression adds the sum of all rows.

$$\begin{aligned}\text{col2(list2)} &= \sum \text{col1(list1)} + \sum \text{col2(list1)} + \sum \text{col3(list1)} + \\ &\quad \sum \text{col4(list1)} - \text{charact1} * (\text{col1(list2)} / \text{col4(list2)}) \\ \text{col2(list2)} &= \sum [\text{col1(list1)} + \text{col2(list2)} + \text{col3(list3)} + \\ &\quad \text{col4(list1)}] - \text{charact1} * (\text{col1(list2)} / \text{col4(list2)})\end{aligned}$$

This yields the following equivalent equations:

```
SUM[list1::col1list1] + SUM[list1::col2list1] + SUM[list1::col3list1]
+ SUM[list1:: col4list1] - charact1 * (list2::col1list2 / list2::col4list2)
```

```
SUM[list1::col1list1 + list1::col2list1 + list1::col3list1 +
list1::col4list1] -
- charact1 * (list2::col1list2 / list2::col4list2)
```

The result of the calculation in row 1 of list2 is as follows:

$38.5 = (1+2+3+4+5) + (0.2+0.4+0.6+0.8+1) + (-2+(-4)+(-3)+0+2) + (10+2+4+6+8) - 10*(0.2/0.8)$.

The values of the other rows are calculated in the same way: 1, 38.5, 33.5, 16.

Related Topics

[Classes, Class Number, and Class Tables](#)

[Characteristic Object - Top Tab](#)

[Characteristic Object - Table Tab](#)

[Characteristic Object - Placement Tab](#)

[Characteristic Object - Status Tab](#)

[Characteristic Object - Rights Tab](#)

[Characteristic Object - Other Tab](#)

[Creating a Standard Characteristic](#)

[Creating a Characteristic to View Data From a Foreign Class](#)

[Creating an Action Button Characteristic](#)

[Creating a List \(List Frame Characteristics and List Columns\)](#)

Creating an Object Reference Characteristic

An object reference characteristic describes a relationship between two object classes. An object reference characteristic also creates a reference button next to the input field that enables a user to enter a value by using “Send To” functionality.



Tip

When creating a new object reference characteristic, it is often useful to view an existing object reference characteristic definition. Alternately, you can copy an existing characteristic and then only change the parameters necessary to create the new characteristic.

For more information and a reference characteristic example, refer to “[Object Reference](#)” on page 138.

Prerequisites

- You invoked the Xpedition EDM Library Cockpit application and connected to an EDM Server using an account with administrative privileges that permits creating new characteristics.
- You have obtained information about the class where you want to assign the characteristic by searching the Object Classes class as described in “[Classes, Class Number, and Class Tables](#)” on page 104. You need the following information:

- The class number. For example, the class number of the Component class is 1. Add the class number on the **Top** tab of the new characteristic definition.
- The class table name. For example, the class table name for the Component class is te_sachnr. Add the class table name on the **Table** tab of the new characteristic definition.

Procedure

1. In the classification pane, open the Admin module and select **Characteristics** to display the Characteristic search window.
2. Right-click, and then choose **Add New Object** from the classification hierarchy popup menu or search results popup menu.

Alternately, you can perform a search, select an existing characteristic from the search results list, right-click, and choose the **Duplicate** popup menu item.
3. On the **Top** tab:
 - Type the name of the new characteristic in the Characteristic field.
 - Type the number of the class to which the characteristic belongs in the Class field.
 - Type the name of the class to reference in the Ref. Class field.
 - In the Value Length field, make sure that the value length of the reference characteristic is the same as the value length of the obj_id of the referenced object class.
 - In the Text list, type the name of the tab to hold the new characteristic (for example, General). The Information Text and Search Text are the characteristic labels to display in the Search and Information window.
4. On the **Table** tab, type the name of the class table in the Table Name field and the name of a column to hold values in Table Column field.
5. On the **Status** tab, check the Object Reference status bit to indicate that the characteristic is a reference to the key of another object class.

Also, check the Class-No. status bit if the obj_id of the referenced class has the Class-No. flag checked.
6. On the **Other** tab, type a unique value for Domain Model Name.
7. Click the **OK** button to add the new characteristic to the database and to close the information window.

Related Topics

[Reference Characteristics](#)

[Characteristic Object - Top Tab](#)

[Characteristic Object - Table Tab](#)

[Characteristic Object - Status Tab](#)

[Characteristic Object - Other Tab](#)

Creating an Action Button Characteristic

In the characteristic definition, an action button characteristic has the Characteristic Type value set to Action Button. An action button characteristic displays as a button in the EDM Library Cockpit user interface that, when clicked, launches an external program.



Note:

Only a qualified administrator who is familiar with the EDM data model, the user community, and how EDM Library is used within his or her company should create characteristics.

The best way to show how to create a action button characteristic is by an example that you can then adapt to your specific data needs. The example in this procedure creates a new action button characteristic to display a 'Hello World' message when clicked. The button is also available in the search window to activate from within the search results list.

Prerequisites

- You invoked the Xpedition EDM Library Cockpit application and connected to an EDM Server using an account with administrative privileges that permits creating new characteristics.
- You have obtained information about the class where you want to assign the characteristic by searching the Object Classes class as described in ["Classes, Class Number, and Class Tables"](#) on page 104. You need the following information:
 - The class number. For example, the class number of the Component class is 1. Add the class number on the **Top** tab of the new characteristic definition.
 - The class table name. For example, the class table name for the Component class is te_sachnr. Add the class table name on the **Table** tab of the new characteristic definition.

Procedure

1. In the classification hierarchy pane, open the Admin module and select **Characteristics** to display the Characteristic search window.
2. Right-click and choose **Add New Object** from the classification hierarchy popup menu or the search results popup menu to add a new characteristic.

Alternately, you can perform a search, select an existing action button characteristic from the selection list, right-click and choose the **Duplicate** popup menu item.

By copying an existing action button characteristic definition, the tabs already contain data. You then only need to modify the characteristic name, the table/column definition, domain model name, and the placement of the characteristic.

3. Enter data or verify values on the various tabs of the characteristic definition, as follows:

a. **Top Tab** ([Figure 117](#))

- Type a name in the Characteristic field (for example, 001hello). The prefix 001 indicates that the action characteristic belongs to object class 1 (Components).



Note:

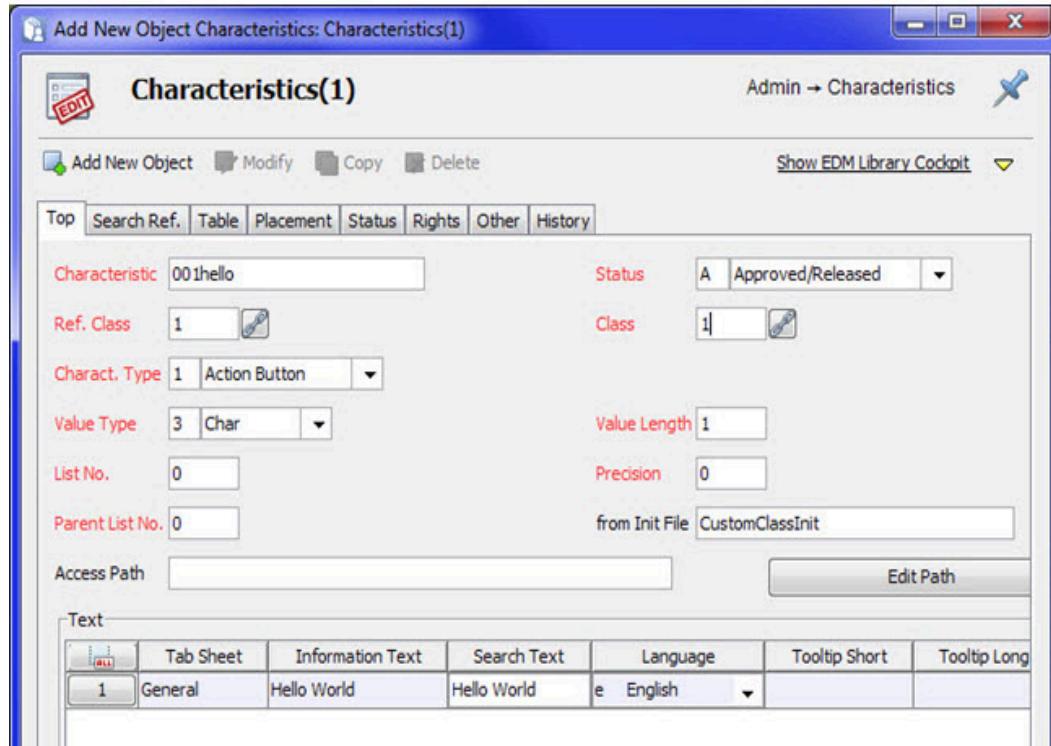
A characteristic name must not begin with the three characters “smw,” as these characters have special meaning to the EDM Library software.

- Type a class number in the Ref. Class and Class field. To create a characteristic and attach it to the Components class, enter 1.
- Set the Characteristic Type value to “Action Button”.
- Set the Value Type field to CHAR (because the data content of an Action characteristic is a program call).
- In the Value Length field specify the length of the input field. Because the actual table column of the button in the database will not contain data, action buttons typically have a value length of 1.

Instead of containing data, an action button usually executes a program call specified in the **Other** tab.

- Type the name to display on the action button in the information window and search results list in the Information Text and Search Text fields.

Figure 117. Top Tab of an Example Action Button Characteristic



b. **Table** tab (Figure 118)

- In the Table Name field, type the name of the class table that holds the characteristics (in this example, “te_sachnr”).
- In the Table Column field, type the name of a column to create within the class table (in this example, “actionhello”).

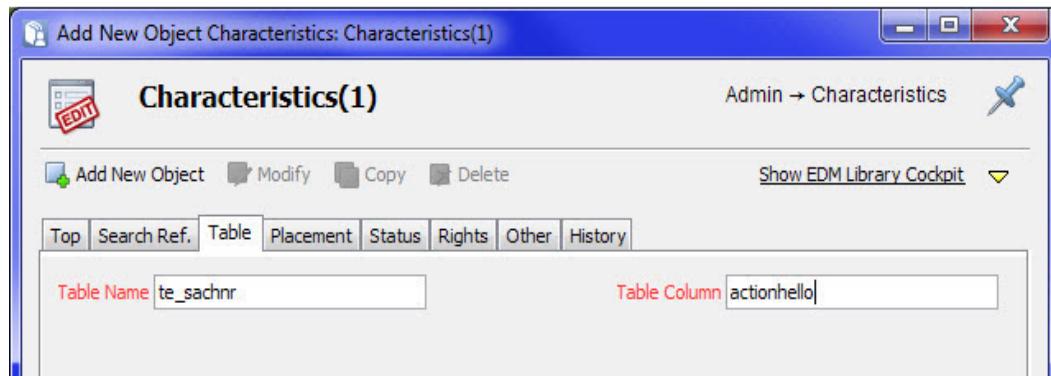


Note:

A table column name cannot be a reserved keyword in SQL (for example, SELECT, UPDATE, and so on). For a list of reserved words in SQL, refer to your database documentation.

The combination of the Table Name and Table Column should not exceed 30 characters for a searchable characteristic. This restriction is necessary because, when searching, the EDM Library software concatenates both fields together to create an Oracle index. If the resulting Oracle index exceeds 30 characters, Oracle can issue an ORA-00972 error.

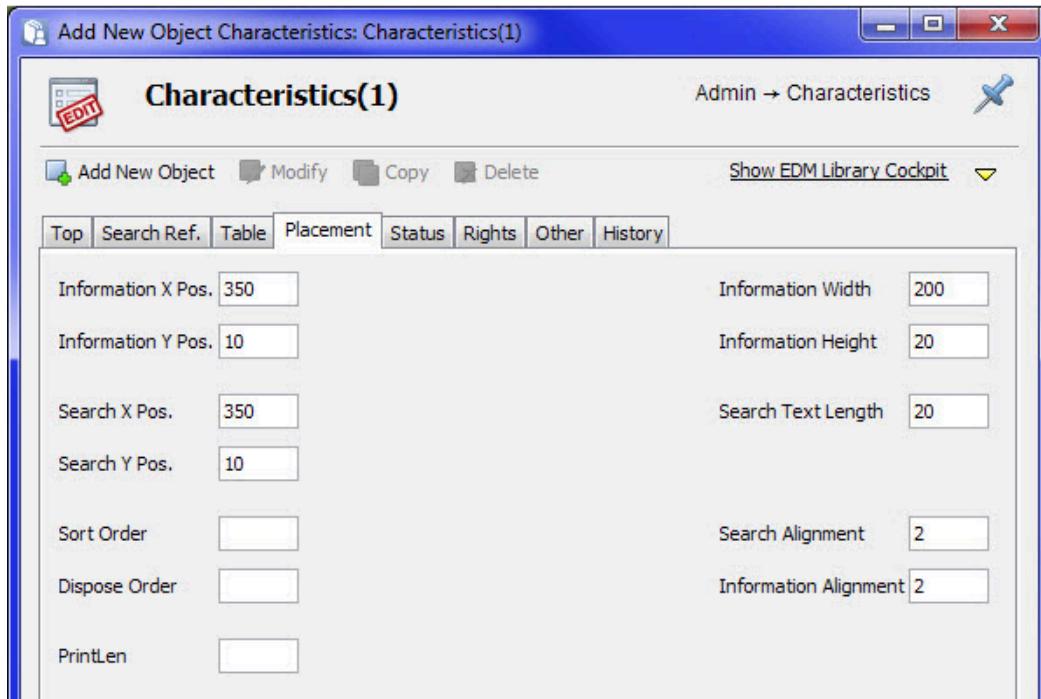
Figure 118. Table Tab of an Example Action Button Characteristic



c. **Placement tab (Figure 119)**

Type the position coordinates of the action button on the **Placement** tab in the Information X Pos and Information Y Pos fields. Use the Information Width and Information Height fields to make the action button a certain size. All position and size information is in pixels.

Figure 119. Placement Tab of an Example Action Button Characteristic



d. **Status** tab

Check the following status bits on the **Status** tab:

- Display Input Mask displays the action characteristic inside the information window.
- Edit permits all users to click the button. If Edit is not checked, use the **Rights** tab to specify only those users or groups that can click the button.
- Display Search Mask, Search Characteristic, and Show to display the button inside the search window. Displaying an action button in a search window automatically creates an **Action** tab, as shown in [Figure 121](#) on page 224.

e. **Other** Tab ([Figure 120](#))

- Assign a unique Domain Model Name value to the new characteristic. Use the name of the characteristic (for example, 001actionhello) to ensure a unique value.
- Type the program call in the Default Value field.



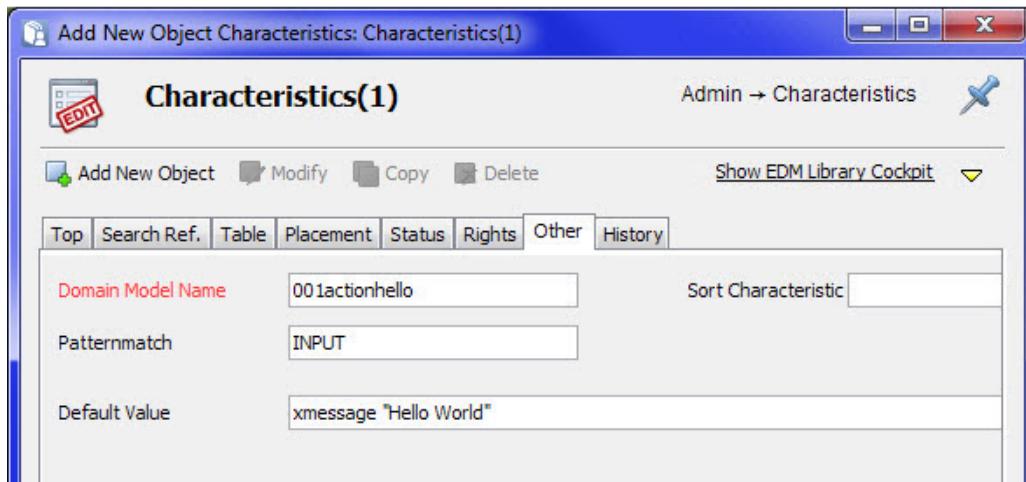
Note:

Instead of using the Default Value field, you can also use the Java Method field to call an internal function. If an entry is present in Java Method, it overrides the entry in the Default Value input field.

In this example, type the following string in the Default Value field to display the “Hello World” message:

```
xmessage "Hello World"
```

Figure 120. Other Tab of an Example Action Characteristic



4. Click either of the following buttons:

- **OK** to add the new characteristic to the database and to close the information window.
- **Save** to create the new characteristic without closing the information window, which enables you to create several characteristics in succession.



Note:

After creating a new characteristic or changing the placement parameters of an existing characteristic, you must reload the data model cache to see the effect in EDM Library Cockpit.

5. Choose **File > Refresh Data Model** or **Tools > Administration > Reload Data Model** to update the database cache with your changes.

Results

The action button characteristic now displays in the search window, search results list, and information window of the Components object class.

In the advanced search pane, activating the checkbox on the **Action** tab displays the button in the search results list (Figure 121). Any characteristics containing the data values used by that button (for example, Datasheet Name in the case of the **View Datasheet** button) are also automatically shown in the search results list. Clicking the button executes the line you typed in the Default Value field of the **Other** tab when defining the characteristic, resulting in the “Hello, World” message.

Figure 121. New “Hello World” Action Characteristic in a Search Window

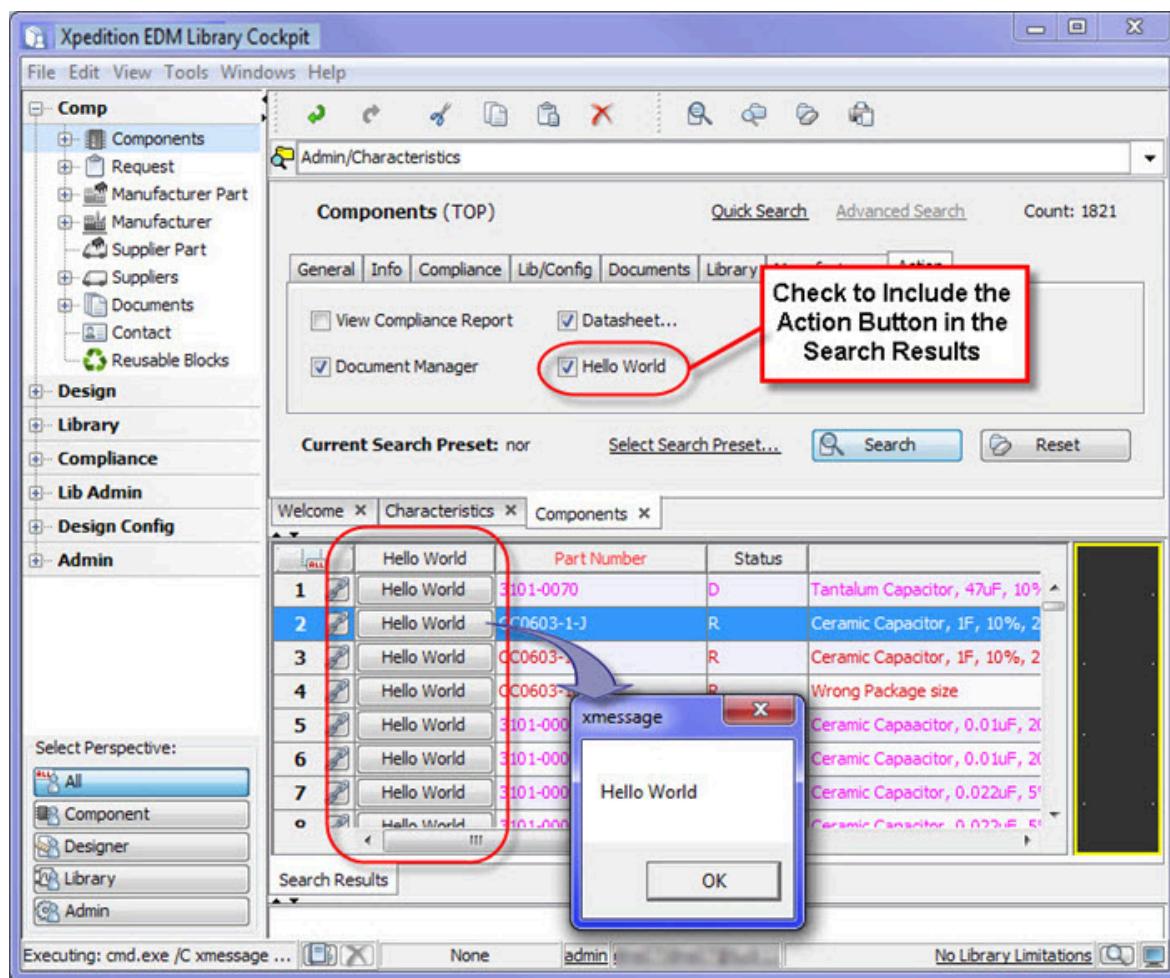
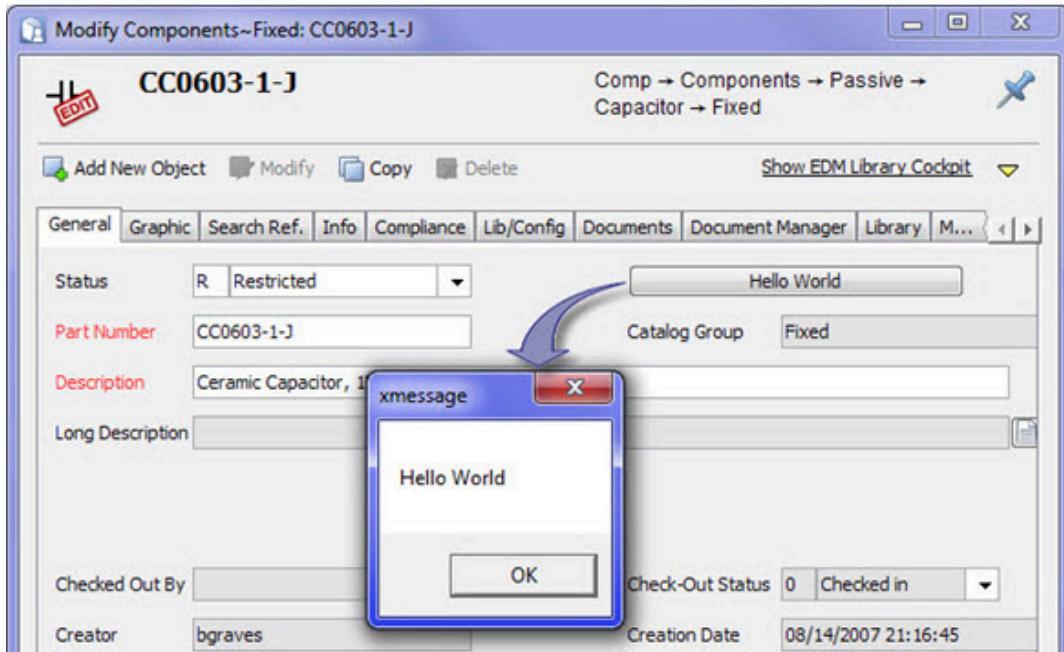


Figure 122 shows the new Hello World action button in the Component information window.

Figure 122. New “Hello World” Action Characteristic in an Information Window



Action buttons can also exist as column characteristics in a list. [“Creating an Action Button Characteristic”](#) on page 218 shows an example procedure to create a list characteristic.

Related Topics

[Classes, Class Number, and Class Tables](#)

[Action Button Characteristic Type](#)

[Characteristic Object - Top Tab](#)

[Characteristic Object - Table Tab](#)

[Characteristic Object - Placement Tab](#)

[Characteristic Object - Status Tab](#)

[Characteristic Object - Rights Tab](#)

[Characteristic Object - Other Tab](#)

[Creating a Standard Characteristic](#)

[Creating a Characteristic to View Data From a Foreign Class](#)

[Creating a Characteristic With a Value That is an Arithmetic Function](#)

[Creating a List \(List Frame Characteristics and List Columns\)](#)

Creating a List (List Frame Characteristics and List Columns)

Use the list frame characteristic type to create a list with one or more columns and an unlimited number of lines. Unlike other characteristics, list frame characteristics that describe a list to display in EDM Library Cockpit reside in their own table within the database, rather than as columns in the class table.

Use one characteristic to create the list frame, and additional characteristic definitions to create the columns within the list frame (refer to “[List Frame Characteristic Type](#)” on page 116) for a description of list elements. In addition to a simple list, lists can be nested within other lists.



Note:

Only a qualified administrator who is familiar with the EDM data model, the user community, and how EDM Library is used within his or her company should create characteristics.

To display a list in EDM Library Cockpit, you must create one list frame characteristic, a line key column characteristic, and additional column characteristics.

The best way to show how to create a list is by an example that you can then adapt to your specific data needs. The procedures in this section create the characteristics that form an example list with several columns in the **General** tab of the Components object class.

[Creating the List Frame Characteristic](#)

[Creating the Line Key List Column](#)

[Creating Additional Columns in a List](#)

[Creating Hierarchical Lists \(List Frames Within List Frames\)](#)

Creating the List Frame Characteristic

The list frame characteristic is the first characteristic you must create to complete a list.

The example values used in this procedure create a list frame that is the first step to create an example list with several columns in the **General** tab of the Components object class. To finish the list, you then create a line key characteristic as described in “[Creating the Line Key List Column](#)” on page 232, and additional column characteristics as described in “[Creating Additional Columns in a List](#)” on page 237.

Prerequisites

- You invoked the Xpedition EDM Library Cockpit application and connected to an EDM Server using an account with administrative privileges that permits creating new characteristics.
- Before creating the list frame and list column characteristics, obtain the class number of the class where you want to assign the characteristics by searching the Object Classes class as described in “[Classes, Class Number, and Class Tables](#)” on page 104. For example, the class number of the Component class is 1. In the procedures, you must enter the class number on the **Top** tab of the new characteristic definition.

List frame characteristics reside in their own database table. Therefore, you do not need to know the class table name to create a list frame characteristic and list columns.

Procedure

1. In the classification hierarchy pane, open the Admin module and select **Characteristics** to display the Characteristic search window.
2. Right-click and choose **Add New Object** from the classification hierarchy popup menu or the search results popup menu to add a new characteristic.

Alternately, you can perform a search, select an existing list frame characteristic from the selection list, right-click and choose the **Duplicate** popup menu item.

By copying an existing list frame characteristic definition, the tabs already contain data.

3. Enter data or verify values on the various tabs of the characteristic definition, as follows:

- a. **Top** tab ([Figure 123](#)):

- Type a name in the Characteristic field (for example, 001list1). The prefix 001 indicates that the characteristic belongs to object class 1 (Components).



CAUTION:

A characteristic name must not begin with the three characters “smw,” as these characters have special meaning to the EDM Library software.

- Type a class number in the Ref. Class and Class field. To attach the characteristic to the Components class, type 1.
- In the Characteristic Type field, choose List Frame.
- In the Value Type field, choose Char.
- In the List No. field, type an integer larger than zero. To avoid conflict with lists already in the default data model or that might be created in a future update, Siemens reserves values between 1000 and 1999 for custom list frames.

The list number is necessary for identification of the list and to be able to assign column characteristics. Enter a list number that does not already exist for this object class (for example, 1001).

To find list numbers already used by other characteristic definitions, perform an advanced search on the Characteristic object class to return a list of characteristics with List No values greater than 0. Then, choose a number between 1000 and 1999 not already used for your custom list frame.

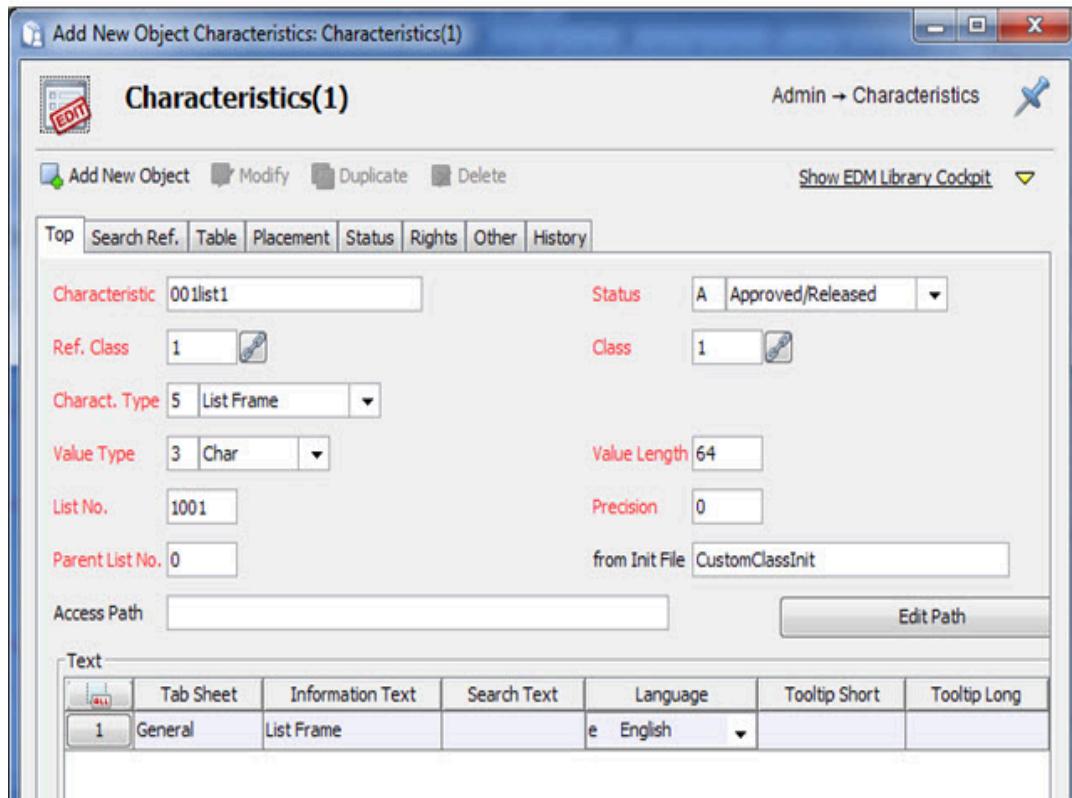
- In the Parent List No. field, leave the default value of 0, as this example does not create a hierarchical list structure.
- In the Value Length field, type 64.

The Value Length of the list frame characteristic must be the same as the Value Length of the obj_id main key characteristic of class 1.

- In the Precision field, leave the default value of 0.

- In the Text list for this example, type “General” as the tab location of the new characteristic and “List Frame” as the label for the list in the Information Text column.
- Leave Option list empty.

Figure 123. Example Top Tab of a List Frame Characteristic



b. **Table** tab:

- Because each list frame has a separate table in the database, enter a name for the new list table in the Table Name field (for example, tl_testlist).
- Enter obj_id in the Table Column input field to denote the first list column of the new list table as the main key. Subsequent list characteristics are additional columns in the list table.



Note:

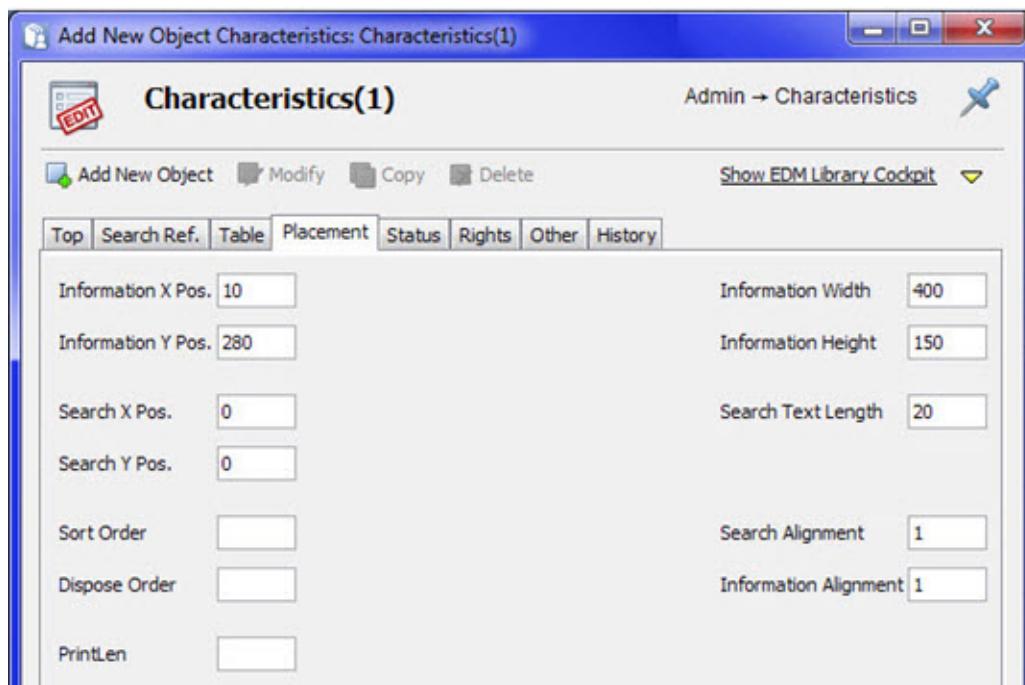
A table column name cannot be a reserved keyword in SQL (for example, SELECT, UPDATE, and so on). For a list of reserved words in SQL, refer to your database documentation.

c. **Placement** tab ([Figure 124](#)):

The list frame characteristic must have a position to prevent having the list overlap with other characteristics. Enter the coordinates of the upper left corner of the list frame in the information window in the Information X Pos and Information Y Pos fields.

Information Width and Information Height properties determine the size of the list frame. Specify a specific list frame size in pixels or '-1' for information width and/or height to automatically adapt the size of the list frame to the size of the information window.

Figure 124. Example Placement Tab for a List Frame Characteristic



d. **Status tab** (Figure 125):

Correct entries in the **Status** tab are extremely important for the proper display and function of the database. In this example, set only the following status bits to reflect a default status for a list frame characteristic:

- Display Input Mask
- Class No.

Only check Class No if the Class-No. flag is also checked for the <class number>obj_id characteristic of the class that is hosting the list frame (in this example, the Class-No. bit of the 001obj_id characteristic). Otherwise, the EDM Library software cannot find the correct reference and cannot make list entries.

- Input Characteristic
- Main Key
- Edit

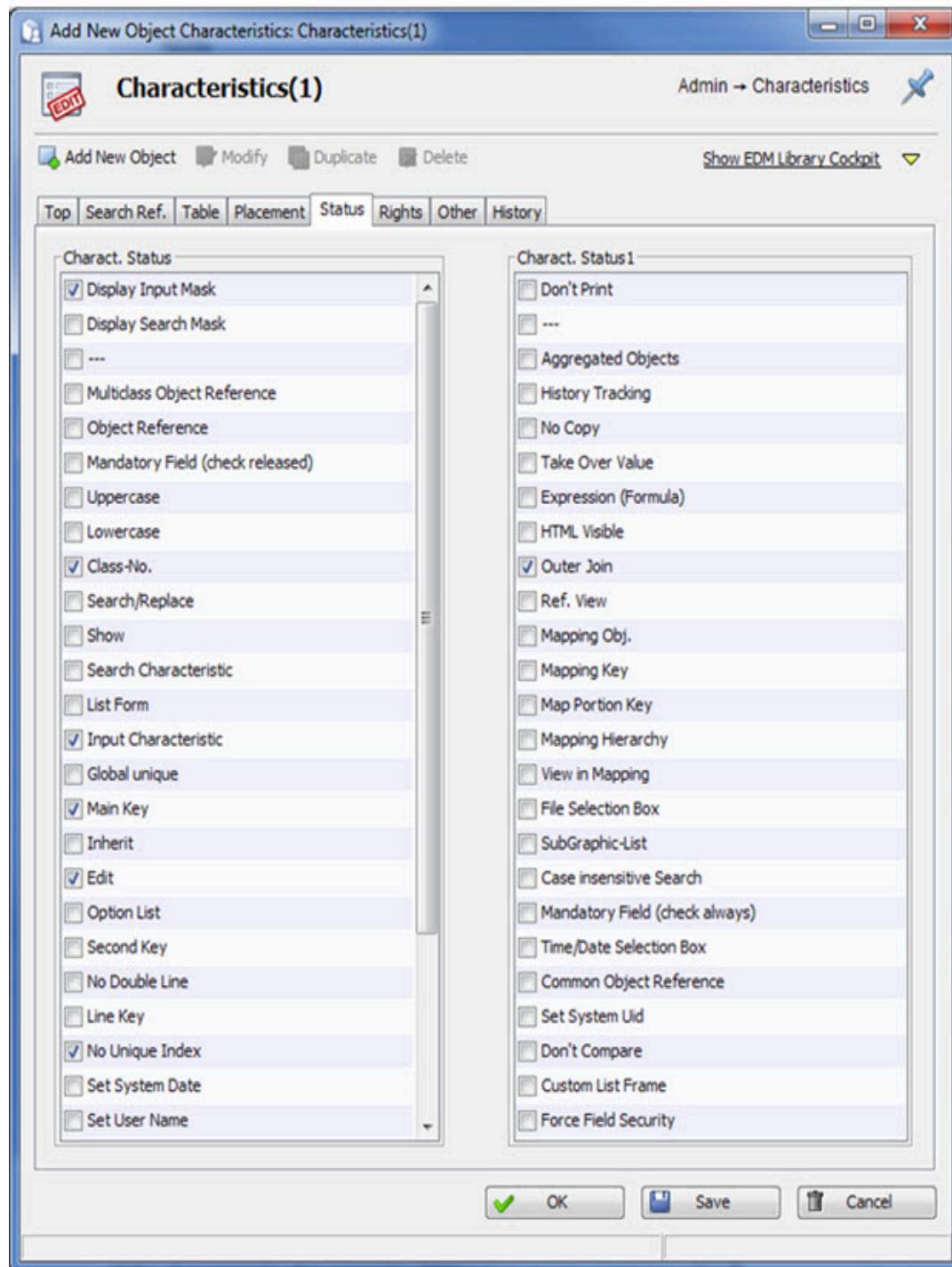
- No Unique Index
 - Outer Join
-



Note:

Do not set the Show bit on the **Status** tab of a list frame characteristic definition.

Figure 125. Default Status for List Frame Characteristics



e. **Other** tab:

Type a unique Domain Model Name value for the new characteristic. Use the name of the characteristic (for example, 001list1) to ensure a unique value.

4. Click **OK** to save and close the list frame characteristic definition.

Results

The list frame Characteristic object now exists in the database.

Having created the list frame characteristic, a line key characteristic for the list is required. Continue to [“Creating the Line Key List Column” on page 232](#) for the procedure to create the line key characteristic.

Related Topics

[List Frame Characteristic Type](#)

[Creating Additional Columns in a List](#)

[Creating Hierarchical Lists \(List Frames Within List Frames\)](#)

[Characteristic Object - Top Tab](#)

[Characteristic Object - Table Tab](#)

[Characteristic Object - Placement Tab](#)

[Characteristic Object - Status Tab](#)

[Characteristic Object - Rights Tab](#)

[Characteristic Object - Other Tab](#)

Creating the Line Key List Column

Once a list frame characteristic exists, create a characteristic that defines the line key list column. Each list frame requires one list characteristic defining the column that acts as the line key. Every entry in the line key column must have a unique value that serves as the row identifier.

You can use one of the following methods to ensure the line key column values are unique:

- Manually specify the value (1, 2, 3, 4 ...).
- Create a unique object reference to another class.
- Concatenate several column characteristic values.

This procedure uses the first method. To avoid typing all the data in all the tabs again, the procedure has you copy the list frame characteristic created in [“Creating the List Frame Characteristic” on page 226](#) and makes only necessary changes to the characteristic definition.



Note:

The example values used in this line key list characteristic creation procedure provide the second step to create an example list with several columns in the **General** tab of the Components object class.

Prerequisites

- You invoked the Xpedition EDM Library Cockpit application and connected to an EDM Server using an account with administrative privileges that permits creating new characteristics.
- You created a list frame characteristic, as described in “[Creating the List Frame Characteristic](#)” on page 226.

Procedure

1. In the classification hierarchy pane, open the Admin module and select **Characteristics** to display the Characteristic search window.
2. Make sure the search window is in advanced search mode, and then type the following search criteria:
 - **Charact.Type** — Choose List Frame.
 - **List No.** — Type the List No value (in this example, 1001).
3. Click **Search**.

The list frame characteristic displays in the search results list (in this example, 001list1).

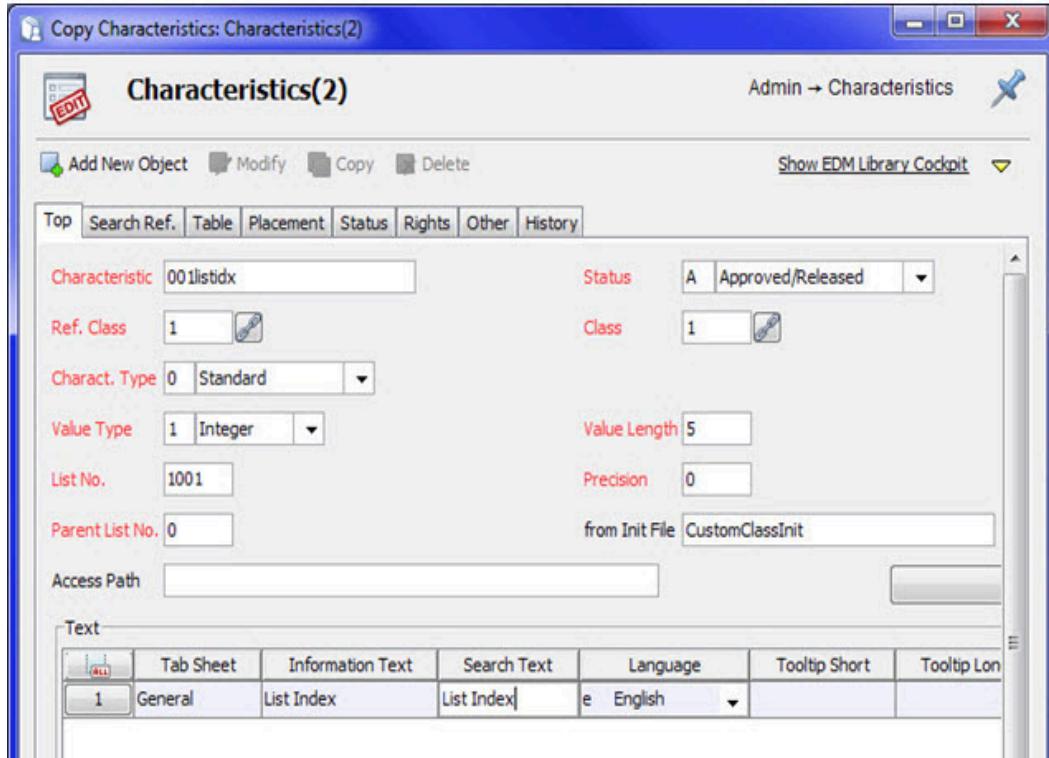
4. Select the line, right-click, and then choose **Duplicate**.

An information window displays for the list frame characteristic. Modify the settings in the tab of the list characteristic so that it becomes a line key column characteristic, as follows:

- a. **Top** tab ([Figure 126](#)):
 - Type a name in the Characteristic field. In this example, 001listidx.
 - In the Characteristic Type field, choose Standard.
 - Choose Value Type and Value Length values appropriate for the identification information that the characteristic will contain. In this example, the Value Type is Integer and the Value Length is 5.

- In the Text area, type a value for the Information Text and Search Text (for example, List Index).

Figure 126. Example Top Tab of a Line Key Characteristic



b. **Table tab:**

- Because the column belongs to the list characteristic, the Table Name is the same as in the list frame characteristic (in this example, tl_testlist).
- In the Table Column field, type a new column name (for example, listidx).



CAUTION:

A table column name cannot be a reserved keyword in SQL (for example, SELECT, UPDATE, and so on). For a list of reserved words in SQL, refer to your database documentation.

c. **Placement tab:**

The entry in the Dispose Order field defines the position of the column in the list. Columns display in ascending order in the table from left to right. This example uses 1 for the Dispose Order of the first index column, and then 2, 3, 4, and so on for the remaining list columns.

Entries in the Information X/Y Pos, Information Width, and Information Height fields have no effect because the columns vary in height according to the number of list entries, and the software automatically adapts the width to either the length of the list heading or the length of the longest entry.

d. **Status tab:**

In this example, check only the following status bits:

- Display Input Mask
- Input Characteristic
- Edit
- Line Key
- Unique Value (List)

To enable a user to search column values from the advanced search criteria pane, check the Display Search Mask and Search Characteristic status bits and use the **Placement** tab to adjust the search position of the characteristic.

e. **Other tab:**

Type a unique Domain Model Name value. Use the name of the characteristic (for example, 001listidx) to ensure a unique value.

5. Click the **OK** button to save and close Characteristic information window.



Note:

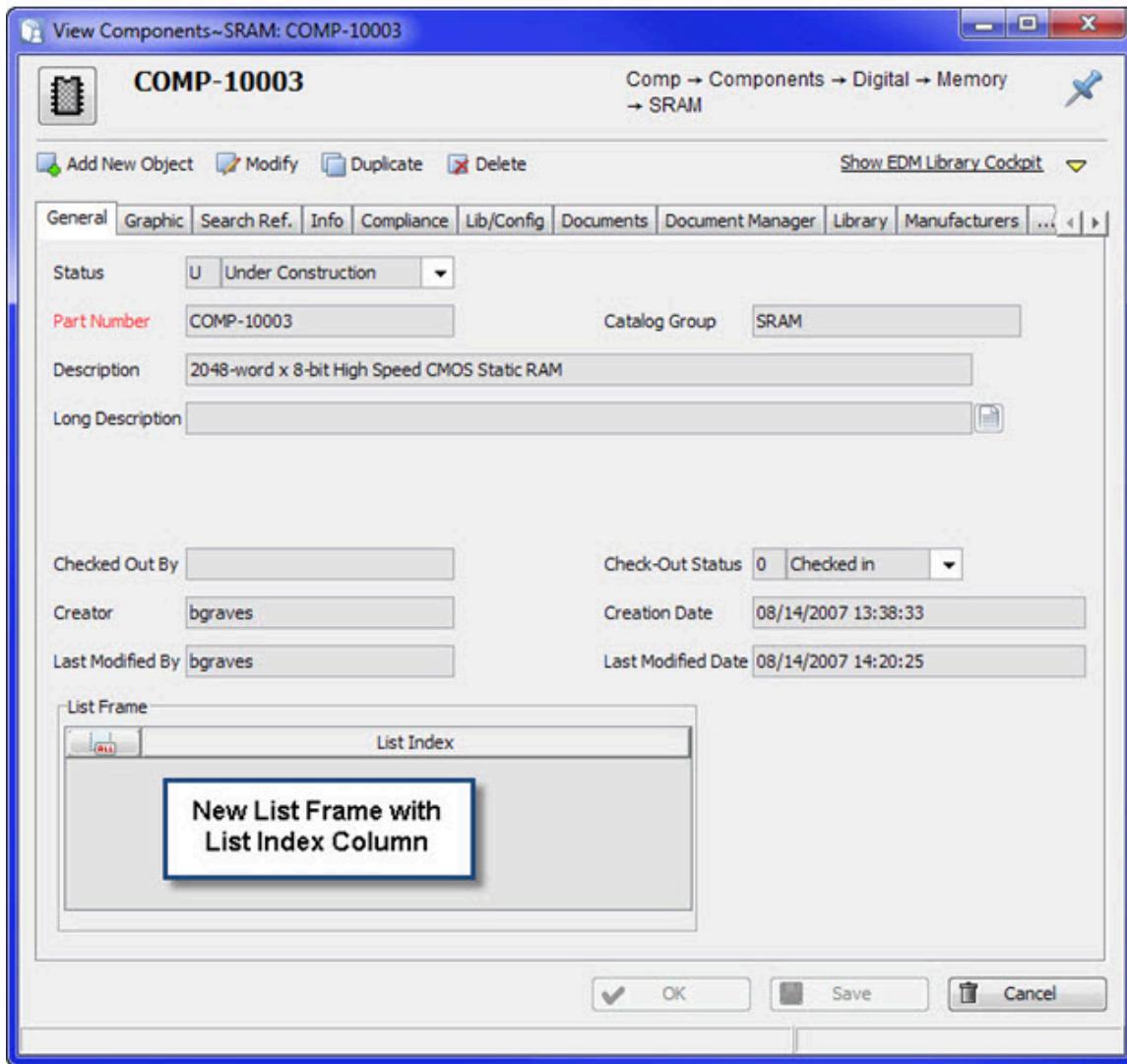
After creating a new characteristic or changing the placement parameters of an existing characteristic, you must reload the data model cache to see the effect in EDM Library Cockpit.

6. Select **File > Refresh Data Model** or **Tools > Administration > Reload Data Model** to reload the data model cache.

Results

You can now select **Comp > Components**, search for existing components and open a Components information window. The component information window should look similar to [Figure 127](#).

Figure 127. Component Information Window With the New List Frame



Related Topics

- [List Frame Characteristic Type](#)
- [Creating the List Frame Characteristic](#)
- [Creating Additional Columns in a List](#)
- [Creating Hierarchical Lists \(List Frames Within List Frames\)](#)
- [Characteristic Object - Top Tab](#)
- [Characteristic Object - Table Tab](#)
- [Characteristic Object - Placement Tab](#)
- [Characteristic Object - Status Tab](#)

[Characteristic Object - Rights Tab](#)

[Characteristic Object - Other Tab](#)

Creating Additional Columns in a List

Once a list frame characteristic and line key characteristic exist, you can create one or more characteristic definitions to define additional list columns.



Note:

The example values used in this characteristic creation procedure provide the third step to create an example list with several columns in the **General** tab of the Components object class.

Prerequisites

- You invoked the Xpedition EDM Library Cockpit application and connected to an EDM Server using an account with administrative privileges that permits creating new characteristics.
- You created a list frame characteristic as described in “[Creating the List Frame Characteristic](#)” on page 226.
- You created a line key characteristic as described in “[Creating the Line Key List Column](#)” on page 232.

Procedure

1. In the classification hierarchy pane, open the Admin module and select **Characteristics** to display the Characteristic search window.
-



Note:

To add an additional column to the list, this procedure copies the line index column characteristic instead of using **Add New Object**. Because the tabs already contain data, you need only replace values where necessary.

2. Search for the characteristic defining the index column of the list (in this example, the 001listidx characteristic you created in the “[Creating the Line Key List Column](#)” task).
3. Select the list frame characteristic row in the search results list, right-click, and choose **Duplicate**.
4. Modify the settings in the tabs of the list index characteristic definition so that it becomes a standard column characteristic, as follows:
 - a. **Top** tab:
 - Type the name in the Characteristic field. In this example, 001col1.
 - In the Characteristic Type field, choose Standard.

- Choose Value Type and Value Length values appropriate for the information that the characteristic will contain. In this example, the Value Type is Char and the Value Length is 20.
- In the Text area, type a value for the Information Text and Search Text (for example, Column 1).

b. **Table** tab:

- Because the column belongs to the list characteristic, the TableName is the same as the list characteristic (in this example, tl_testlist).
- In the Table Column field, type a new column name (for example, col1).



CAUTION:

A table column name cannot be a reserved keyword in SQL (for example, SELECT, UPDATE, and so on). For a list of reserved words in SQL, refer to your database documentation.

c. **Placement** Tab:

In the Dispose Order field, type an entry to specify the position of the column in the list. Columns display in ascending order in the list table from left to right. In this example, because the Dispose Order of the list index is 1, use 2 for the Dispose Order of the column characteristic. Use higher dispose order numbers for additional column characteristics.

Entries in the Information X/Y Pos, Information Width, and Information Height fields have no effect because the columns vary in height according to the number of list entries, and the software automatically adapts the width to either the length of the list heading or the length of the longest entry.

d. **Status** tab:

In this example, check only the following status bits:

- Display Input Mask
- Input Characteristic
- Edit

To enable a user to search column values from the advanced search criteria pane, check the Display Search Mask and Search Characteristic status bits and use the **Placement** tab to adjust the search position of the characteristic.

e. **Other** tab:

Type a unique Domain Model Name value. Use the name of the characteristic (for example, 001col1) to ensure a unique value.

5. Click the **OK** button to save and to close the Characteristic information window.



Note:

After creating a new characteristic or changing the placement parameters of an existing characteristic, you must reload the data model cache to see the effect in EDM Library Cockpit.

6. Select **File > Refresh Data Model** or **Tools > Administration > Reload Data Model** to reload the data model cache.

Results

You can now select **Comp > Components**, search for existing components and open a Components information window. After adding two example lines to the list, the component information window looks similar to [Figure 128](#).

Figure 128. A New List Frame With Two Columns

The screenshot shows the 'Modify Components~SRAM: COMP-10003' dialog box. At the top, there's a breadcrumb trail: 'Comp → Components → Digital → Memory → SRAM'. Below the title bar are buttons for 'Add New Object', 'Modify', 'Duplicate', and 'Delete', along with a 'Show EDM Library Cockpit' link. The main area contains tabs for General, Graphic, Search Ref., Info, Compliance, Lib/Config, Documents, Document Manager, Library, Manufacturers, and more. Under the General tab, there are fields for Part Number (COMP-10003), Catalog Group (SRAM), Description (2048-word x 8-bit High Speed CMOS Static RAM), and Long Description. Below these are fields for Checked Out By, Check-Out Status (0, Checked in), Creator (bgraves), Creation Date (08/14/2007 13:38:33), Last Modified By (admin), and Last Modified Date (11/16/2011 17:02:34). A 'List Frame' section contains a table with two columns: 'List Index' and 'Column 1'. The table has two rows: Row 1 with index 1 and value 'Value 1'; Row 2 with index 2 and value 'Value 2'. Below the table are 'New List Frame with Two Columns and Data Values' buttons for adding and deleting rows. At the bottom are 'OK', 'Save', and 'Cancel' buttons.

Related Topics

- [List Frame Characteristic Type](#)
- [Creating the List Frame Characteristic](#)
- [Creating the Line Key List Column](#)
- [Creating Hierarchical Lists \(List Frames Within List Frames\)](#)
- [Characteristic Object - Top Tab](#)
- [Characteristic Object - Table Tab](#)
- [Characteristic Object - Placement Tab](#)
- [Characteristic Object - Status Tab](#)
- [Characteristic Object - Rights Tab](#)
- [Characteristic Object - Other Tab](#)

Creating Hierarchical Lists (List Frames Within List Frames)

You can create hierarchical list frames by defining list frames within list frames.

Prerequisites

- You invoked the Xpedition EDM Library Cockpit application and connected to an EDM Server using an account with administrative privileges that permits creating new characteristics.

Procedure

1. Create the parent (topmost) list frame characteristic as described in “[Creating the List Frame Characteristic](#)” on page 226. Make sure to check the List Form status bit on the **Status** tab. If this bit is not checked, then the sub-list is not visible in the list frame.
2. Define the parent line key characteristic to be auto-concatenated (that is, check the Take Over Value bit) with the `<class>obj_id` characteristic. Alternately, check the Set System Uid bit to force a unique identifier for each row of each object.
3. Define the child list frame characteristic the same as the parent list frame characteristic, but with the following differences:
 - a. Make the value for Parent List No on the **Top** tab equal to the value of List No on the parent list frame characteristic.
 - b. Make the value for Value Length on the **Top** tab equal to the Value Length of the Line Key characteristic of the parent list.

- c. Make sure that the Main Key status bit on the **Status** tab is not checked.
 - d. Make sure that Class-No. status bit on the **Status** tab has the same setting as the Class-No. status bit on the parent line key characteristic.
4. For all other sub-list characteristics, make sure that the value of Parent List No. is set to 0 and that the value of List No. is the same (these are the same rules as for other lists).

Related Topics

- [List Frame Characteristic Type](#)
- [Creating the List Frame Characteristic](#)
- [Creating the Line Key List Column](#)
- [Creating Hierarchical Lists \(List Frames Within List Frames\)](#)
- [Characteristic Object - Top Tab](#)
- [Characteristic Object - Table Tab](#)
- [Characteristic Object - Placement Tab](#)
- [Characteristic Object - Status Tab](#)
- [Characteristic Object - Rights Tab](#)
- [Characteristic Object - Other Tab](#)

Setting User-Based or Group-Based Characteristic Editing Permissions

As an administrator, you can override permission settings for an individual characteristic to permit only a specific user or group to edit the characteristic.



Note:

You cannot grant greater rights at the characteristic level than permitted at the class or catalog group level.

The characteristic to change can be of type standard, text frame, list frame, or action button.

Restrictions and Limitations

- The user role assigned to a user group must have Modify or Edit All rights to the class containing the characteristic. A user cannot modify a characteristic without Modify or Edit All class rights, regardless of the settings on the **Rights** tab of a Characteristic object.
- Settings on the **Rights** tab of the catalog group definition cannot prevent the user from editing characteristics in that catalog group.

Prerequisites

- You invoked the Xpedition EDM Library Cockpit application using an account with administrative privileges that permits modifying a characteristic definition.

Procedure

1. Identify the characteristic to change, and then open the information window for that characteristic in Modify mode.
2. Bring the **Rights** tab forward and click  to add a row to the User List for each user or user group you want to have edit access. Specify the name of the user or user group in the User column and choose **Edit** in the Right column.



Note:

If the **Rights** tab is empty, users inherit class rights from the user group to which they belong. Adding an Edit entry denies edit rights to all other users or groups unless you specifically list them on the **Rights** tab.

You can either type a value in the User column or use “Send To” functionality as follows:

- a. Click the link icon (). In the EDM Library Cockpit session window, a **User** tab displays in the search criteria pane and the cursor changes to a magic wand.
 - b. Use the tabbed search pane to return a list of users or user groups. In the search results, individual users have a Type value of “1”, while user groups have a Type value of “2”.
 - c. Select a row.
 - d. Right-click and choose the **Send to** popup menu item to return the value back to the User column of the characteristic definition.
3. Bring the **Status** tab forward and uncheck the Edit status bit.
 4. Click **OK** to save the characteristic definition and to close the information window.

Results

Only the user(s) or members of the user group specified on the **Rights** tab can edit the characteristic value.

Examples

Users brad and catj are both members of the default Component Engineers group, which grants rights to edit characteristic values in the Components object class. You want to change the rights of the Commercial Identifier characteristic (001comm_id) so that only brad can enter a value for this characteristic, but not catj or any other member of the Component Engineers group.

Figure 129 shows the **Rights** and **Status** tabs of the modified 001comm_id characteristic definition.

Figure 129. Characteristic Definition Edit Settings

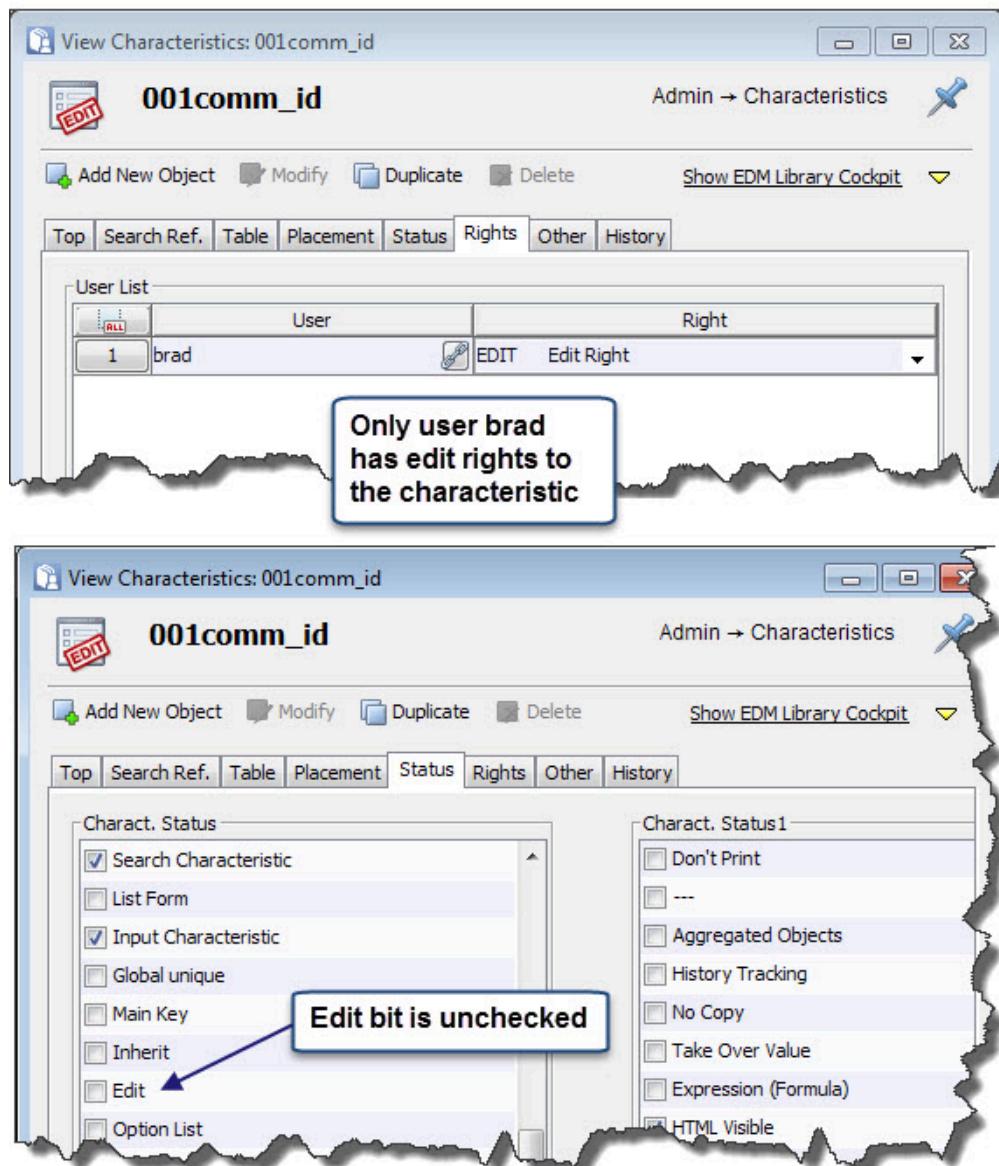
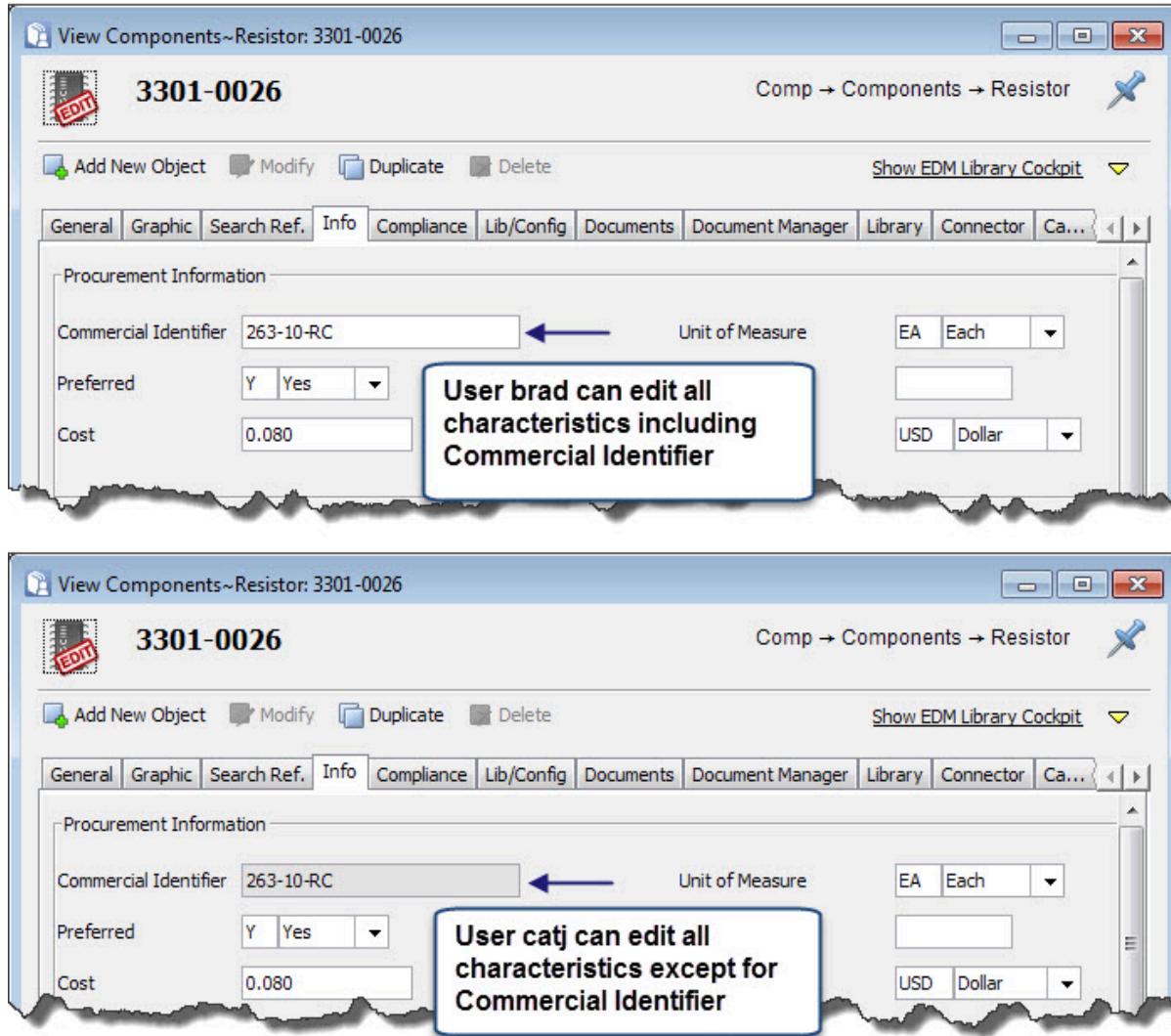


Figure 130 shows the result when users catj and brad attempt to edit characteristic values on the **Info** tab of a Component object.

Figure 130. Characteristic Rights Differences Between Users



Related Topics

[Access Permission Levels](#)

[Showing User Group Permissions](#)

[Opening a Characteristic Information Window](#)

[Characteristic Object - Rights Tab](#)

[Characteristic Object - Status Tab](#)

[Catalog Group Object - Rights Tab](#)

Search Preset Administration

The Xpedition EDM Library Cockpit application enables a user to save search criteria settings and recall them to repeat the search later. Saved search criteria resides in the Search Preset object class. The application can save any number of different search preset settings.

Of the Admin classes, Search Preset objects are the only items that are usually managed by a design engineer or a librarian instead of an administrator. To add new search presets or edit existing search presets, a user must be using an appropriate license role or otherwise been granted edit permission for the Search Presets object class. Search presets are saved for static characteristics, dynamic characteristics, and the currently activated catalog group. Search presets can be saved for individual users or for the entire user community.

Working in the Search Presets object class lets you view more information about different search presets, modify search criteria defined by a preset, or delete one or more saved presets from the database. Also, if you cannot remember exactly what a search preset does based on its name, searching the Search Preset objects can return more information (such as the comments associated with a preset).

To locate all saved search presets (not just the presets for a specific object type), use one of the following methods:

- In the object classification pane, open the Admin module and choose **Search Presets** to display a search window for the Search Preset object class. Execute a search to return a list of Search Preset objects.
- Choose the **Tools > Search Presets > Manage search presets** pulldown menu item to return a list of all Search Preset objects.

With a list of returned Search Preset objects in the search result area, you can now use the popup menu to perform the same actions you would on other objects.

Related Topics

[Saving Search Criteria \[Xpedition EDM Library Overview\]](#)

Input Pattern Checking

In the classification pane, **Admin > Input Pattern Checks** provides access to objects that specify a specific character pattern. A static or dynamic characteristic definition can reference an Input Pattern Check object, so that any input into the characteristic must match the data values and patterns specified by the Input Pattern Check object. If the input into the characteristic does not match the pattern, an error message occurs.

The data model provides several default input patterns, which you can use as examples when creating additional custom input patterns unique to your organization (for example, an input pattern that follows your company part numbering convention). When creating an Input Pattern Check object, an administrator can use regular input pattern syntax or a Java regular expression to specify the input pattern.

[How EDM Library Applies Pattern Checks](#)

[Regular Input Pattern Syntax](#)

[Java Regular Expression Syntax](#)

[Opening an Input Pattern Check Information Window](#)

[Input Pattern Check Object - Top Tab](#)

How EDM Library Applies Pattern Checks

When creating a new characteristic, an administrator uses the **Other** tab of a Characteristic information window to specify the input pattern (if any) to apply to the characteristic.

If the characteristic definition references an Input Pattern Check object, the EDM Library software checks the characteristic value when the user or external program saves an object. If the object contains a value that violates the input pattern check rules, the EDM Library software displays an error message and prevents saving the object to the database.

For example, the Input Pattern Check object used by a characteristic holding telephone number information has the following syntax to specify an acceptable seven digit telephone number pattern:

<0-9><0-9><0-9><0-9><0-9><0-9><0-9>

Notice that the pattern contains no spaces or special characters between numbers.

Using this input pattern accepts a telephone number of the form “3255733” as input. But, typing the telephone number into the characteristic as “325-5733” displays an error message when you try and save the object because the pattern does not permit hyphens. Notice also that the input pattern is only seven characters long, so typing an eighth character would similarly violate the pattern.

Because input pattern checks are also valid through the EDM Library API, a pattern check can prevent an external application (such as a loader program or schematic capture application) from transferring data to the database and using that data as a characteristic value if the value would violate the conditions of the pattern check.

Related Topics

[Characteristic Object - Other Tab](#)

[Regular Input Pattern Syntax](#)

[Java Regular Expression Syntax](#)

[Input Pattern Check Object - Top Tab](#)

Regular Input Pattern Syntax

Regular input patterns (entered into the Pattern field of an Input Pattern Check object) must adhere to certain syntax rules.

- The pattern string specifies the anticipated entry. For example:

xyz accepts only xyz

- A comma separates alternatives. For example:

abc,xyz accepts abc or xyz.



Note:

There must not be a space character after the comma.

- An asterisk (*) accepts any number of characters. For example:

A* accepts A, AA, AAAA, and so on.

- The question mark (?) accepts any printable character. For example:

?* accepts all subsequently printable characters.

- A hyphen (-) enables simpler character classes. For example:

<0-9> accepts a number from 0 to 9. <A-C> accepts either A, B, or C.

- The tilde (~) excludes individual characters from the character class. For example:

<0-9~6~7> accepts all digits from 0 to 9 except 6 and 7.



Note:

An input pattern can also specify the operators - * ~ < > () , and \ to include or exclude. Mark these characters with a backslash (\) so that the system can recognize that they are not operators. To specify a backslash (\) in an input pattern, type a double backslash (\\\).

You can combine syntax characters with each other. The following examples show various types of input patterns:

- Date (for example, 12.06.2017)

A date is made up of three groups of numbers, separated by a period. The first digit of the first pair (representing the month) is either 0 or 1, while the second digit of the first pair is 0, 1, or 2. The first digit of the second pair (representing the day) is between 0 and 3. The third group of numbers (representing the year), must begin with "20" with all other digits accepting any value (0 to 9). The syntax is as follows:

<0-1><0-2>.<0-3><0-9>.20<0-9><0-9>

- Interface Pin (for example, IN1)

The names of interface pins can contain any character except for the space character, to ensure compatibility with all other programs (IN 1 is not valid). The name can be any length. The syntax is as follows:

<?~>*

Java Regular Expression Syntax

Java regular expressions follow the definitions of the Java programming language and provide the ability to create more complex input patterns than can be described with the regular pattern input syntax. Type Java regular expressions in both the Look at Reg Exp and Reg Expressions fields of an Input Pattern Check object.

The following shows a Java regular expression example that restricts a characteristic value to a string of exactly five Greek characters.

- Value in Look at Reg Exp: [p{InGreek}]{0,5}

EDM Library applies this value to restrict input to a maximum of five Greek characters when entering data into an input field.

- Value in Reg Expressions: [p{InGreek}]{5}

EDM Library applies this value after the user enters data and tries to save the object. The check only permits exactly five Greek characters. EDM Library does not save the object in the database if the input string contains fewer or more than five Greek characters.

Opening an Input Pattern Check Information Window

Return a list of Input Pattern Check objects to the search results list, and then open an information window on a selected object to view the contents. An Input Pattern Check object only contains a **Top** tab and a **History** tab.

Prerequisites

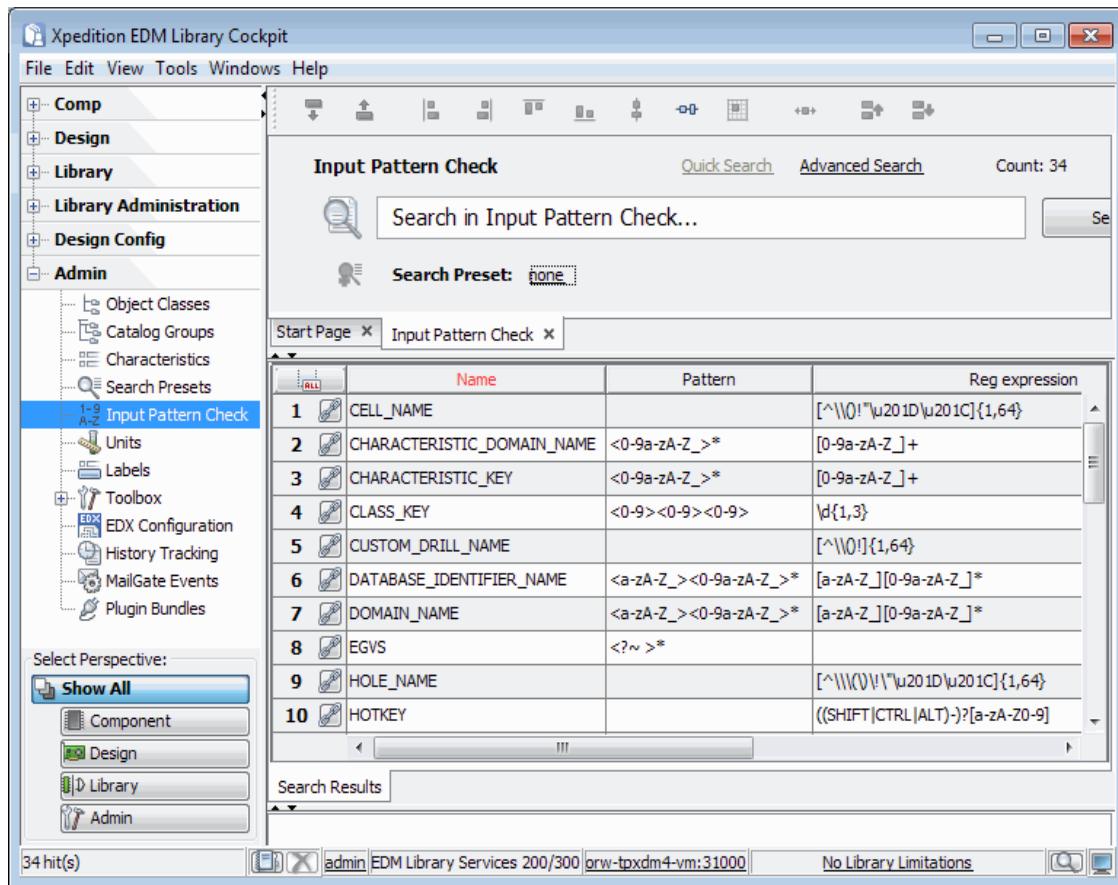
- You invoked the Xpedition EDM Library Cockpit application and connected to an EDM Server.
- Your user account has administrator privileges.

Procedure

1. In the classification hierarchy pane, open the Admin module and select **Input Pattern Check** ([Figure 131](#)). Alternately, type Admin/Input Pattern Check in the location bar.

The search criteria pane can be in either quick search mode or advanced search mode, depending on your preference settings. A link at the top of the search criteria pane lets you switch between modes.

Figure 131. Input Pattern Check Search Window



Tip

For general information about searching in Xpedition EDM Library Cockpit and how to formulate a search query, refer to “Searching and Replacing” in the *EDM Library Overview*

2. Enter a search query into the quick search field in the quick search pane or search queries into one or more characteristic input fields in the advanced search criteria pane. When using the advanced search pane, place a check mark next to the characteristics on the tabs that you want as columns in the search results list.

In advanced search mode, you can enter search criteria in multiple fields on multiple tabs before executing a search. Leave the search criteria blank to return all Input Pattern Check objects to the search results pane.

3. Click the **Search** button.
4. Select a row from the search results list. Then, with the cursor still in the search results area, right-click and choose an editing function.

Alternately, click the reference button to the left of a row to open an information window in view mode without first having to select the row or use the popup menu.

An information window can either be floating or docked. Your EDM Library Cockpit preferences determine the default.

Related Topics

[How EDM Library Applies Pattern Checks](#)

[Regular Input Pattern Syntax](#)

[Java Regular Expression Syntax](#)

[Input Pattern Check Object - Top Tab](#)

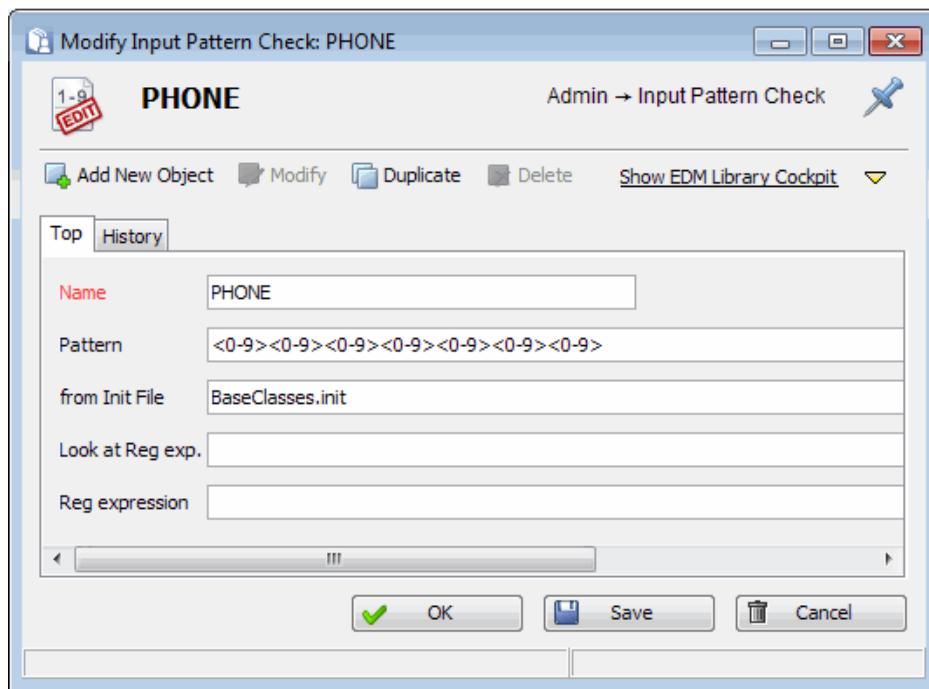
Input Pattern Check Object - Top Tab

To access: Refer to “[Opening an Input Pattern Check Information Window](#)” on page 248

The **Top** tab of an Input Pattern Check object defines an input pattern.

Description

Figure 132. Top Tab of the Input Pattern Check Information Window



When entering information on the **Top** tab, choose whether to use a regular input pattern or Java regular expression:

- Type the syntax in the Pattern field of the **Top** tab and leave the Look at Reg Exp and Reg Expression fields empty.
- Type values in the Look at Regular Expression field and the Reg Expression field (both fields must have values).

Values in the Look at Regular Expression field and the Reg Expression field override the value in the Pattern field. That is, once the Look at Regular Expression field and the Reg Expression field contain data

values, EDM Library programs always use the Java regular expressions before the regular input pattern when possible.

Characteristics

- **Name** — A required field that specifies the unique identifier for the input pattern. A letter or number code is acceptable, depending on the configuration of the characteristic.
- **Pattern** — Specifies the regular pattern syntax. In [Figure 132](#), the Pattern field indicates that when a characteristic applies the PHONE input pattern, a user must enter telephone numbers as seven numeric digits, and each digit must be between zero and nine (inclusive).
- **from Init File** — If loading an initialization file creates the input pattern object, the system writes the name of the initialization file in this field. When creating a new input pattern object, entering a value enables the **batchadmin** tool to automatically write the input pattern definition to that initialization file.
- **Look at Reg Exp**— Contains the Java regular expression to evaluate as a user types a character sequence into an input field.
- **Reg Expression** — Contains the Java regular expression to evaluate when a user or external program saves an object to the database.

Related Topics

[How EDM Library Applies Pattern Checks](#)

[Regular Input Pattern Syntax](#)

[Java Regular Expression Syntax](#)

[Opening an Input Pattern Check Information Window](#)

Unit Administration

In the classification pane, **Admin > Units** provides access to objects that define the unit definition to apply and display for a certain numeric characteristic value. The default data model contains several predefined unit objects for common electrical and measurement units such as watts, farads, ohms, amperes, volts, and so on. Each unit object defines the thresholds for both large and small values; for example, when a value is appropriate to express as a watt (W), as a kilowatt (KW), or as a milliwatt (mW).

[Opening a Units Information Window](#)

[Units Object - Top Tab](#)

Opening a Units Information Window

Return a list of Units objects to the search results list, and then open an information window on a selected object to view the contents. A Units object only contains a **Top** tab and a **History** tab.

Prerequisites

- You invoked the Xpedition EDM Library Cockpit application and connected to an EDM Server.
- Your user account has administrator privileges.

Procedure

1. In the classification hierarchy pane, open the Admin module and select **Units**. Alternately, type Admin/Units in the location bar.

The search criteria pane is in either quick search mode or advanced search mode, depending on your preference settings. A link at the top of the search criteria pane lets you switch between modes.



Tip

For general information about searching in Xpedition EDM Library Cockpit and how to formulate a search query, refer to “Searching and Replacing” in the *EDM Library Overview*.

2. Enter a search query into the quick search field in the quick search pane or search queries into one or more characteristic input fields in the advanced search criteria pane. When using the advanced search pane, place a check mark next to the characteristics on the tabs that you want as columns in the search results list.

In advanced search mode, you can enter search criteria in multiple fields on multiple tabs before executing a search. Leave the search criteria blank to return all Units objects to the search result pane.

3. Click the **Search** button.
4. Select a row from the search results list. Then, with the cursor still in the search results area, right-click and choose an editing function.

Alternately, click the reference button to the left of a row to open an information window in view mode without first having to select the row or use the popup menu.

An information window can either be floating or docked. Your EDM Library Cockpit preferences determine the default.

Related Topics

[Units Object - Top Tab](#)

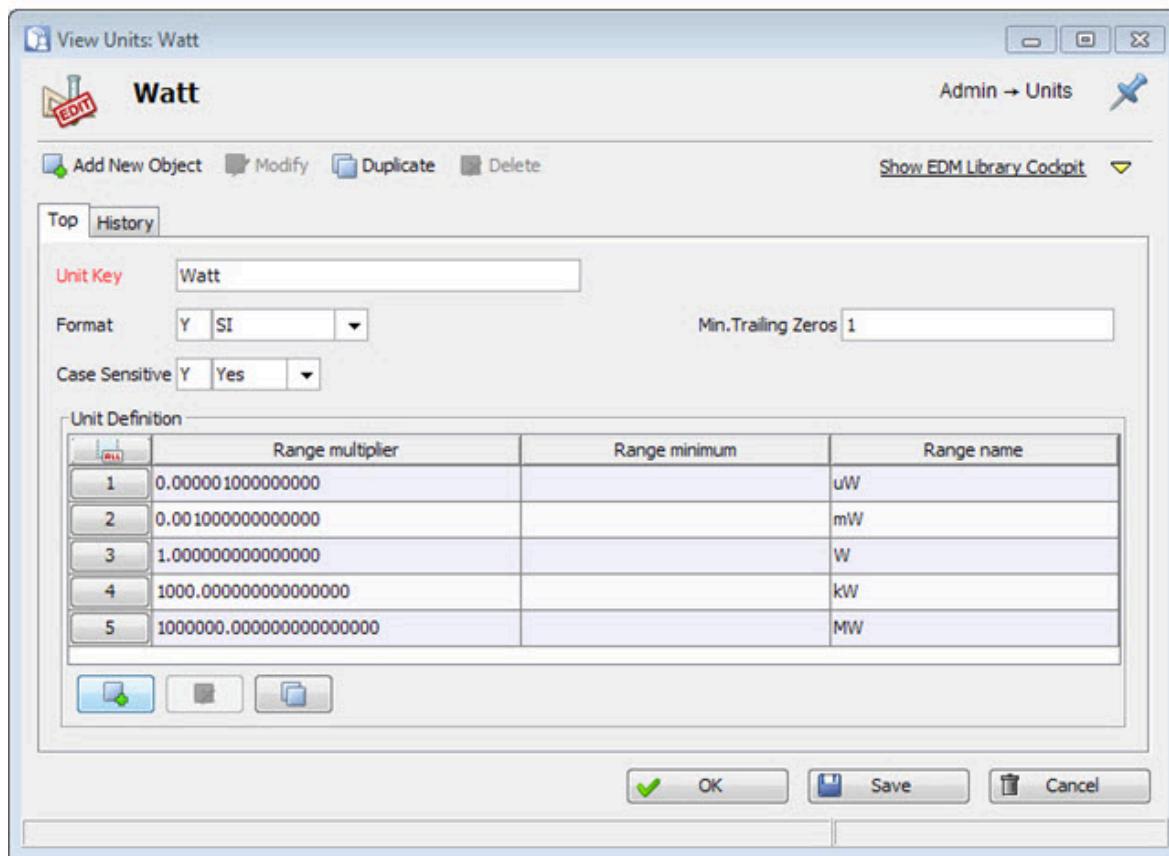
Units Object - Top Tab

To access: Refer to “[Opening a Units Information Window](#)” on page 252

The **Top** tab of an Units object defines acceptable values for the unit.

Description

Figure 133. Top Tab of the Unit Information Window



Characteristics

- **Unit Key** — A required string value that uniquely identifies the unit object (for example, “W” for “Watts”).
- **Format** — Defines the display format of the data values assigned to this unit definition and contains the following possible values are:
 - Y (SI)** — The value is a rational decimal number (for example, resistance 1.7k)
 - N (European)** — The value uses the European notation (for example, resistance 1K7)
 - E (Scientific)** — The value is displayed in exponential format (for example, resistance 1.7E +001k)
- **Min. Trailing Zeros** — A value that applies to the SI format and scientific E format only. Min. Trailing Zeros defines the display precision (that is, the number of digits to the right of the decimal point) in a data value. For example, a value of 3 results in a display value of 1.700k (SI format) or 1.700E+001k (Scientific format).
- **Case Sensitive** — A Y (Yes) or N (No) value that determines how to match the case of the range name during a search. For example, assume there is a defined unit with a range named MW but, while searching, a user provides a search value of 10 mw. With Case Sensitive set to Y, EDM Library does not report a match. However, if Case Sensitive is set to N, then a match shows in the search results list.

The Unit Definition list contains entries that define the numeric thresholds and multipliers to apply when determining the appropriate unit values to use. The Unit Definition list has the following columns:

- **Range Multiplier** — Specifies the value multiplier and lower limit to use when applying the Range Name. For example, [Figure 133](#) on page 253 shows the multiplier values to use when determining whether to show a unit of uW (microwatts), mW (milliwatts), W (watts), KW (kilowatts), or MW (megawatts) behind the numeric value.
The column must always contain a value. For any base unit value, the range multiplier value is always 1.
- **Range Minimum** — Lets an administrator specify a minimum threshold value to use when assigning a value to the Range Name. If there is a value in the Range Minimum column, EDM Library uses that value and ignores any value in the Range Multiplier column. The Range Minimum column thus enables display in preferred units when the value is less than one (or example, displaying the 0.00000001 farad value of a capacitor as 0.01uF instead of 10000pF, as in [Figure 134](#)).

Figure 134. Using the Range Minimum Column

| Unit Definition | | | |
|-----------------|-------------------|-------------------|------------|
| | Range multiplier | Range minimum | Range name |
| 1 | 0.000000000001000 | | pF |
| 2 | 0.000001000000000 | 0.000000100000000 | uF |
| 3 | 1.000000000000000 | | F |

In the default data model, none of the unit definitions have a defined range minimum. If your company prefers to always display values in certain units, you must edit the appropriate unit objects to add the range minimums.

- **Range Name** — Specifies the nomenclature to use for the unit (for example, V for Volt, kV for Kilovolt, and so on). Note that if the value of Format is set to N (denoting European notation), then the Range Name column can only contain a single letter such as “V” or “K” corresponding to each range limit.

Related Topics

[Opening a Units Information Window](#)

Label Customization

Labels objects in the database store the text that displays on various windows in Xpedition EDM Library Cockpit, text that displays on the buttons in the search and information windows, error message text associated with operations, and keyboard shortcuts to pulldown menu items. Changing existing name values in Labels objects enable an administrator or developer to customize or localize the EDM Library Cockpit application.



Tip

To discard customizations and restore label text to the factory settings, use the **batchadmin** command with the -labels switch (refer to “[batchadmin Syntax](#)” on page 364).

[Opening a Labels Information Window](#)

[Labels Object - Top Tab](#)

[Defining Hot Keys for Menu Items](#)

Opening a Labels Information Window

Return a list of Labels objects to the search results list, then open an information window on a selected object to view the content.

Prerequisites

- You invoked Xpedition EDM Library Cockpit and connected to an EDM Server.
- Your account has administrator privileges.

Procedure

1. In the classification hierarchy pane, open the Admin module and select **Labels**. Alternately, type Admin/Labels in the location bar.

The search criteria pane is in either quick search mode or advanced search mode, depending on your preference settings. A link at the top of the search criteria pane lets you switch between modes.



Tip

For general information about searching in Xpedition EDM Library Cockpit and how to formulate a search query, refer to “[Searching and Replacing](#)” in the *EDM Library Overview*.

2. Type a search query into the search criteria pane. Leave all search criteria fields empty to return a list of all Labels objects.

Advanced search mode enables you to enter search criteria into one or more input fields on multiple tabs and place a check mark next to the characteristics that you want as columns in the search results list.

3. Click **Search**.

4. Select a row from the search results list. Then, with the cursor still in the search results area, right-click, and then choose an editing function.

Alternately, click the reference button to the left of a row to open an information window in view mode without first having to select the row or use the popup menu.

An information window can either be floating or docked. Your EDM Library Cockpit preferences determine the default.

Related Topics

[Labels Object - Top Tab](#)

[Defining Hot Keys for Menu Items](#)

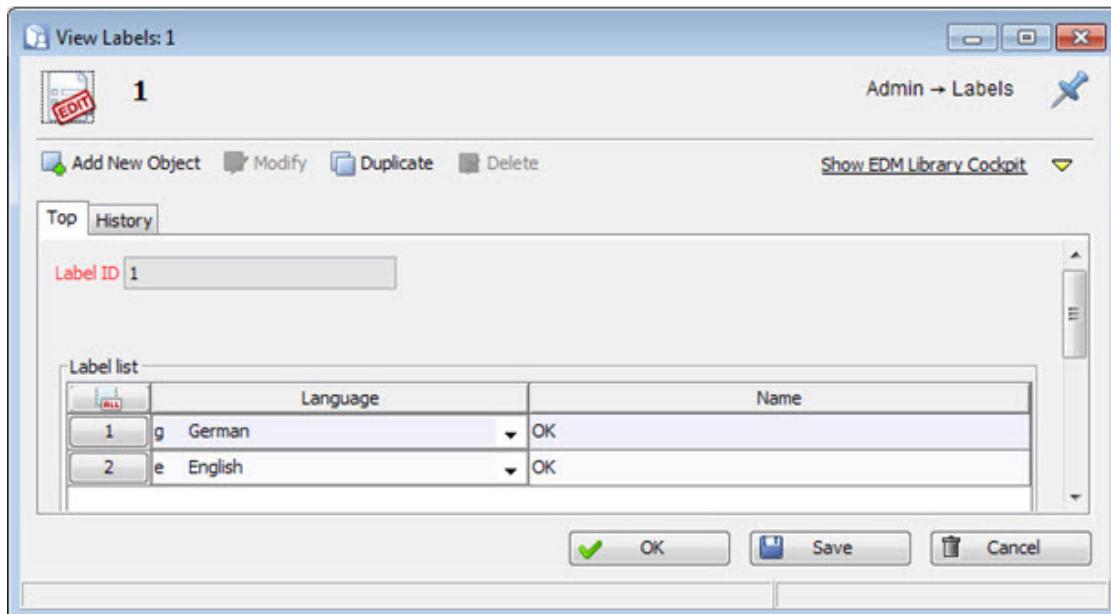
Labels Object - Top Tab

To access: Refer to “[Opening a Labels Information Window](#)” on page 256

The **Top** tab of a Labels object defines the language and text for the label.

Description

Figure 135. Top Tab of the Labels Information Window



Characteristics



Note:

VX.2 and later releases only support English language labels and ignore non-English rows.

- **Label ID** — A required characteristic that uniquely identifies the label. EDM Library accepts a letter or number code as a label ID, depending on the configuration of the characteristic.
- **Language** — The language code. All Labels objects have text to support the default language code of “e” for “English.”
- **Name** — The label text that displays in the EDM Library Cockpit application.

Related Topics

[Opening a Labels Information Window](#)

[Defining Hot Keys for Menu Items](#)

Defining Hot Keys for Menu Items

You can redefine keyboard shortcuts (hot keys) for some menu items in the **File**, **Edit**, **Windows**, and **Help** pulldown menus, provided those menu items already have associated Labels objects. You can also use the Labels object class to define keyboard shortcuts for custom menu items.

Procedure

1. Open the Labels object associated with the text of the menu item.
2. Type the name of the menu item in the Name column of the Label List box, followed by a caret (^) character, and then the keys that make up the shortcut. The syntax must have the following pattern:

`<menu_item_name>^(C|S|CS)+([A...Z][[1...9]|[F1...F12])`

If used, the “C” (Control key), “S” (Shift key), or “CS” (Control-Shift key combination) follow the caret (^). A letter, number, or function key can then follows the “C”, “S”, or “CS” designators. EDM Library does not permit key combinations of C+C, C+X, or C+V because they would conflict with the predefined copy, cut, and paste keyboard shortcuts in Windows.

3. Choose **Tools > Administration > Reload Data Model** to update the database cache with your changes.

Examples

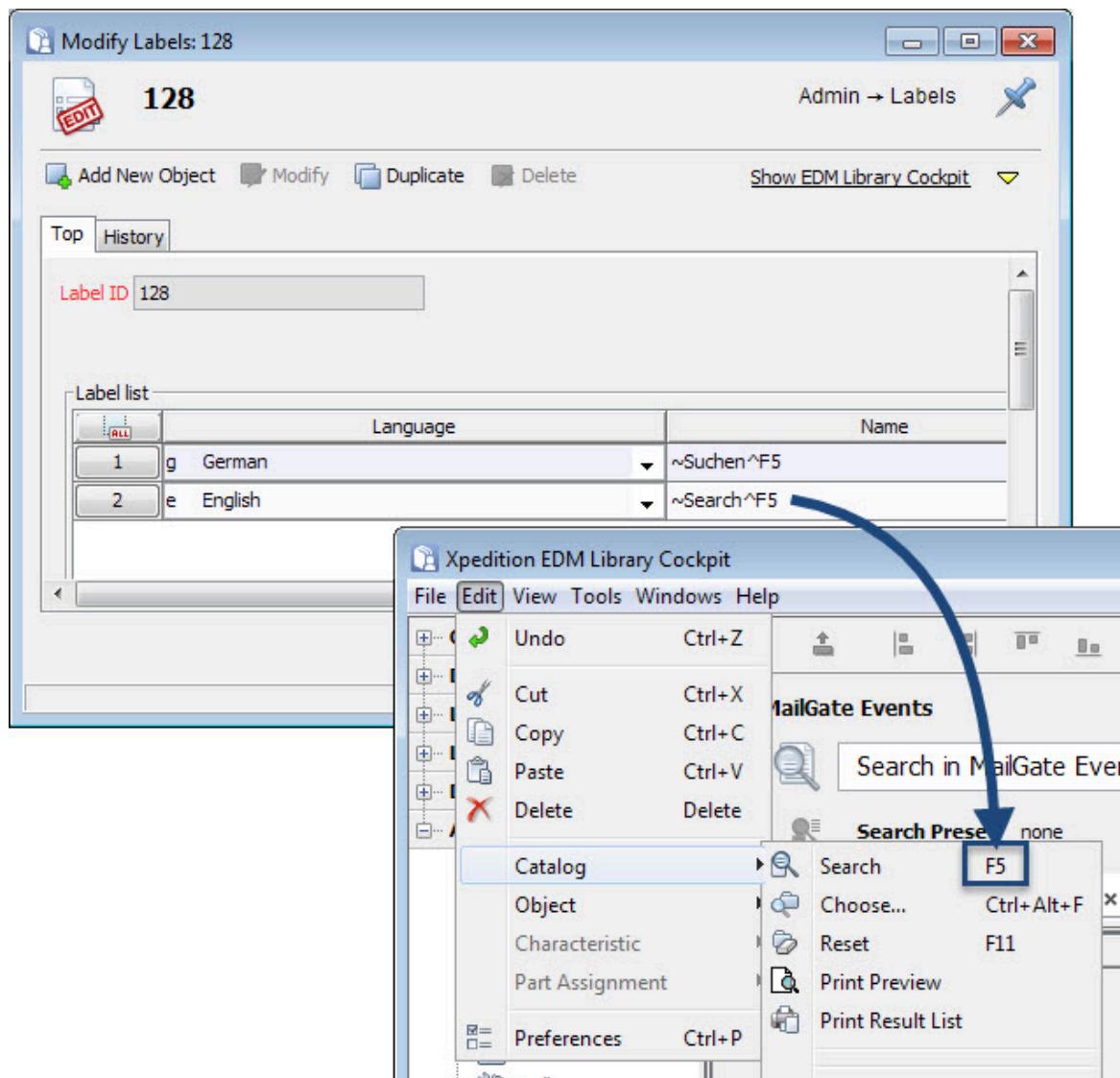


Note:

In the older interface, the tilde (~) instructed the software to underline the character that followed in the menu item name. The current interface ignores the tilde and does not render an underline.

Figure 136 shows an example of the keyboard shortcut definition for the **Edit > Search** pulldown menu item. In Figure 136 the default Label object defines the F5 key as a keyboard shortcut for the menu item. Optionally, you could change the value (for example, to use the F4 key instead of F5), and then re-initialize the database cache to use a different keyboard shortcut.

Figure 136. Keyboard Shortcut for the Edit > Search Menu Item



Related Topics

[Opening a Labels Information Window](#)

[Labels Object - Top Tab](#)

The Toolbox

Admin > Toolbox provides access to objects stored in the Toolbox object class. Toolbox objects specify the rules a client application, macro, or interface should follow when establishing a database connection or when transferring characteristic data into or out of a database. For example, default Toolbox objects specify how design tools use the IPC interface to communicate with schematic capture applications and transfer data when performing instantiation, assigning, and BOM extraction operations. Similarly, custom toolboxes created by a developer specify how EDM Library software communicates and transfers data when using custom plug-in applications.

Some Toolbox objects also specify access or transfer rules for certain modules of functionality, such as identifying the characteristics to use in quick search.

Each Toolbox object specifies the rules for one program or functional area. Not all programs use all tabs on a Toolbox object, and some programs require that a Toolbox object have a special tab to specify information just for that program.

[Default Toolboxes and Hierarchy](#)

[Information on Default Toolbox Tabs](#)

[Identification of a Toolbox for Program Usage](#)

[Opening a Toolbox Information Window](#)

[Toolbox Object - Top Tab](#)

[Toolbox Object - IPC Tab](#)

[Toolbox Object - Menu Tab](#)

[Toolbox Object - MetaDataMap Tab](#)

[Toolbox Object - History Tab](#)

[Part Replacement Toolboxes](#)

[Databook Toolboxes](#)

[Quick Search Toolbox](#)

[BOM Extraction Toolbox](#)

[Update Component Info Toolbox](#)

[Compliance Management Toolboxes](#)

[Component Synchronization Toolbox](#)

[Content Provider Toolboxes](#)

[Assignment Rows](#)

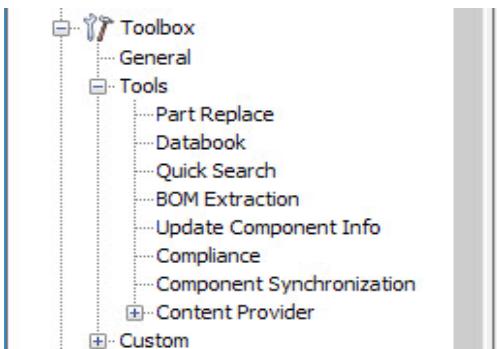
Default Toolboxes and Hierarchy

The default toolbox hierarchy has General, Tools, and Custom subcatalogs. The Tools subcatalog is further divided according to the functionality supported by the toolbox. The Custom catalog does not contain objects in a default database because it is reserved for Toolbox objects a developer creates to use with custom applications.

**Note:**

The Compliance branch is only present in the Tools subcatalog if the database is configured with the Compliance module. The Content Provider branch is only present in the Tools subcatalog if the database is configured with EDM Supplychain functionality.

Figure 137. Toolbox Hierarchy



General Toolboxes

Tools Toolboxes

General Toolboxes

The General catalog in the toolbox hierarchy is the storage location for general purpose toolboxes created by initializing a database. Toolboxes in the General catalog contain only the minimum default set of tabs, and do not contain any special tabs to augment basic toolbox functionality.

The following default toolboxes reside in the General catalog:

- **mentordx_ipc** and **mentordx_ipc_2007**: — Contains information needed to establish an IPC connection with the Xpedition Designer schematic capture application and a listing of the characteristic values to transfer during instantiation, part assignment, or part replacement. There are separate toolboxes for pre-2007 and 2007 and later versions of Xpedition Designer due to changes in the way Xpedition Designer stores schematic data on the file system.
- **mentordx_updc** — Defines the component characteristic values to transfer from the database to the part (PDB) when creating a production library cache.
- **SERVERCONFIGURATION**: — This toolbox is no longer used by EEVX.2 and later applications.

Tools Toolboxes

The Tools catalog in the toolbox hierarchy contains toolboxes unique to special Siemens plug-in functionality. Toolboxes that reside in the Tools catalog usually contain one or more dynamic tabs not found in the general toolboxes and that are specific to the application or functionality that the toolbox supports.

The Tools catalog group can contain the following subcatalogs (with each subcatalog containing one or more Toolbox objects):

- **Part Replace** — Adds the dynamic **Part Replacement** and **Replacement Restrictions** tabs used by part replacement functionality (refer to “[Part Replacement Toolboxes](#)” on page 279).
- **Databook** — Adds the dynamic **Databook** tab that provides for Databook configuration when using the EDM Library Connector application (refer to “[Databook Toolboxes](#)” on page 285).
- **Quick Search** — Adds the dynamic **Quick Search** tab used when performing a quick search in Xpedition EDM Library Cockpit (refer to “[Creating a Quick Search Toolbox](#)” on page 303).
- **BOM Extraction** — Enables the addition or ignoring of non-existing EDM Library components in a BOM (refer to “[BOM Extraction Toolbox](#)” on page 305).
- **Update Component Info** — Enables the transferring of units into a BOM (refer to “[Update Component Info Toolbox](#)” on page 309).
- **Compliance** — When the Compliance Manager module is installed, the `ComplianceManagerConfig:` toolbox describes the object classes on which to perform a compliance calculation and the Java class to use to process the data for that calculation (refer to “Administrative Tasks” in the *Xpedition EDM Library Compliance Management Module Guide*).
- **Component Synchronization** — Specifies the separator characteristic and a list of excluded characteristics when synchronizing data between parent and child components (refer to “Synchronizing Parent and Child Components” in the *Xpedition EDM Library Guide for Component Engineers*).
- **Content Provider** - When EDM Supplychain functionality is enabled, specifies the parameters and proxy host settings to interface with an external content provider (refer to “[Content Provider Toolboxes](#)” on page 314).

Default toolboxes residing in the Tools catalog include the DXReplace: toolbox to support part replacement functionality in Xpedition Designer and the DXDBDefault: toolbox to support connection to Databook through EDM Library Connector. Each of these toolboxes can be copied and customized, but must remain in the Tools catalog.

Information on Default Toolbox Tabs

All Toolbox objects contain a set of default tabs (named **Top**, **IPC**, **MetaDataTable**, **Menu**, and **History**) that define information available to an application, macro, or interface program. Every Toolbox object always requires identifying information on the **Top** tab. By default, the system also maintains history information on the **History** tab. But, how an application or process uses the toolbox determines which other tabs require information.

The default tabs provide the following types of information:

- Name of the IPC client or the name of an external program that uses the toolbox
- Library specification that applies to the toolbox
- Object classes and characteristics to transfer from the database to the client or from the client to database
- Method to convert characteristic and property names when transferring the information between a database and the client
- Conversion direction (from the database to a CAE tool or from a CAE tool to the database)
- Default values of the characteristics in the CAE application to use if no data values are available in the source application
- Arguments to transfer to the client(s)
- IPC function (for example, Instantiate or Cross Highlight)
- Syntax for the transfer (for example, whether or not the order of arguments to transfer is fixed or whether the arguments can be parsed)
- Caller that is the source for an IPC transfer (for example, 'Hit List', 'Object Macro' or 'Object List')

Related Topics

[Default Toolboxes and Hierarchy](#)

[Identification of a Toolbox for Program Usage](#)

[Toolbox Object - Top Tab](#)

[Toolbox Object - IPC Tab](#)

[Toolbox Object - MetaDataMap Tab](#)

[Toolbox Object - Menu Tab](#)

[Toolbox Object - History Tab](#)

Identification of a Toolbox for Program Usage

Every Toolbox object contains three characteristics on the **Top** tab named Call name, Call name key, and Lib Spec. The administrator or developer who creates a toolbox must always enter a Call name value. In some cases (depending on the program using the toolbox), the administrator or developer might also enter a Lib Spec value to restrict the toolbox for use with only that library specification. Using the values for Call name and Lib Spec, the system automatically creates a value for Call name key.

If there is more than one toolbox with the same Call name, the Call name key characteristic uniquely identifies the toolbox according to the related Library Specification object. When receiving a request for

toolbox access by a particular client, the EDM Library software searches for the appropriate Toolbox object according to the following algorithm:

- If EDM Library software finds a toolbox that has a Call name key in the following format, it uses that toolbox:

<Call name>:<Library Specification>

For example, `mentordx_ipc_2007:SLEN`.

- If there is no toolbox containing a library specification value as part of the Call name key, the EDM Library software searches for a toolbox without a library specification value in the Call name key, but that still has the trailing colon:

<Callname>:

For example, `mentordx_ipc_2007`:

- If there is no toolbox with either a trailing library specification value or a trailing colon, the EDM Library software uses a toolbox with a Call name key that has the same value as the call name characteristic. For example, `mentordx_ipc_2007`.

With toolboxes used with IPC communication, the EDM Library software automatically reloads the current toolbox when a user changes the active production library setting as follows:

- The EDM Library software searches for a toolbox corresponding to the library specification referenced by the new production library setting (for example, `mentordx_ipc_2007:SLEN`).
- If there is no toolbox corresponding to a library configuration, it uses the default toolbox (for example, `mentordx_ipc_2007`).



Note:

When using the assignment functionality, the EDM Library software only supports one production library setting per session. While in an assignment window, do not change the production library setting, as it could invalidate your results.

Related Topics

[Default Toolboxes and Hierarchy](#)

[Information on Default Toolbox Tabs](#)

Opening a Toolbox Information Window

Return a list of Toolbox objects to the search results list, and then open an information window on a selected object to view the contents. A Toolbox object contains the default **Top**, **IPC**, **Menu**, **MetaDataMap**, and **History** tabs, plus possible additional dynamic tabs if the toolbox resides in the Tools catalog.

Prerequisites

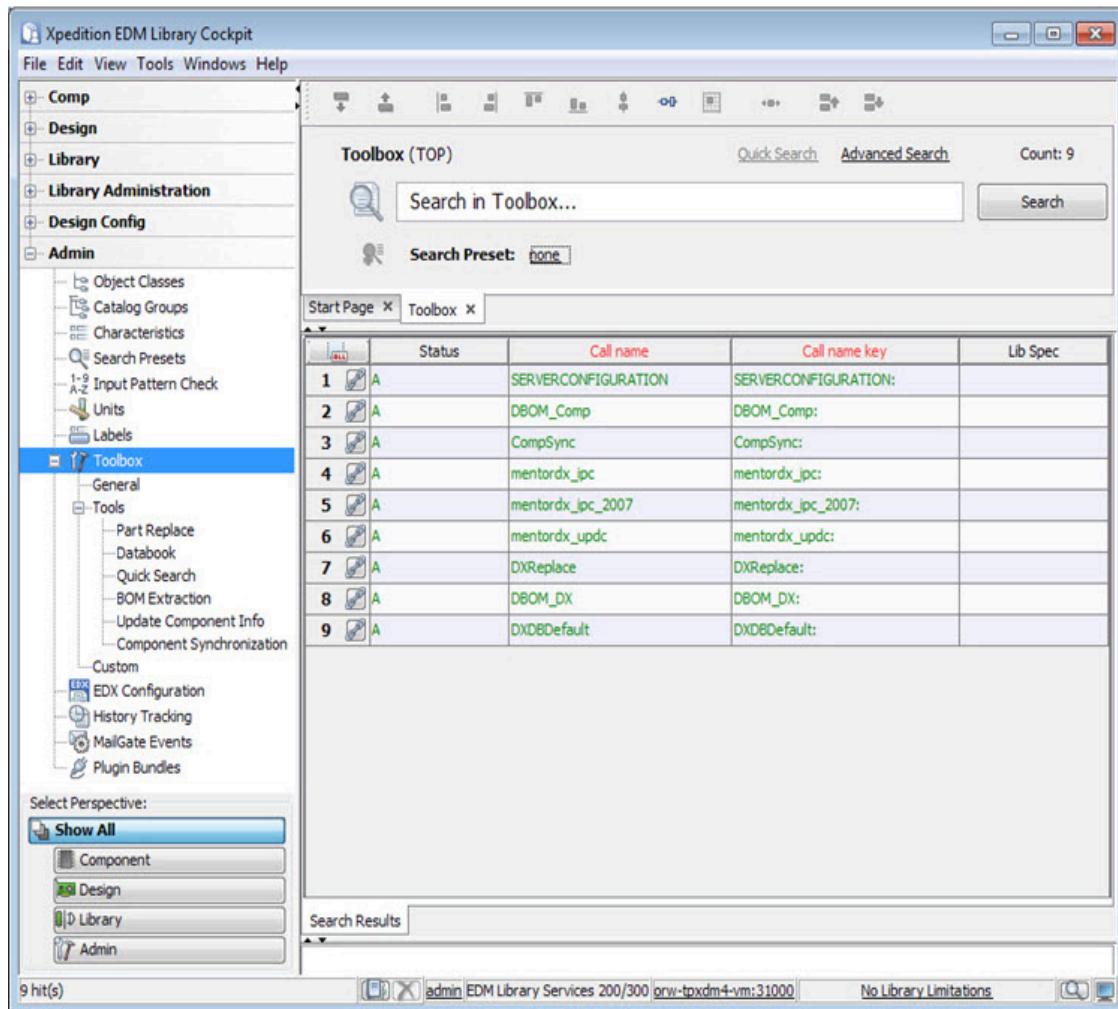
- You invoked Xpedition EDM Library Cockpit and connected to an EDM Server.
- Your user account has administrator privileges.

Procedure

1. In the classification hierarchy pane, open the Admin module and select **Toolbox**.

The search criteria pane (Figure 138) is in either quick search mode or advanced search mode, depending on your preference settings. A link at the top of the search criteria pane lets you switch between modes.

Figure 138. Toolbox Search Window





Tip

For general information about searching in Xpedition EDM Library Cockpit and how to formulate a search query, refer to “Searching and Replacing” in the *EDM Library Overview*.

2. Type a search query into the search criteria pane. Leave the search criteria blank to return all Toolbox objects to the search result pane.

Advanced search mode enables you to enter search criteria into one or more input fields on multiple tabs and place a check mark next to the characteristics that you want as columns in the search results list.

3. Click **Search**.

4. Select a row from the search results list. Then, with the cursor still in the search results area, right-click and choose an editing function.

Alternately, click the reference button to the left of a row to open an information window in view mode without first having to select the row or use the popup menu.

An information window can either be floating or docked. Your EDM Library Cockpit preferences determine the default.

Related Topics

[Toolbox Object - Top Tab](#)

[Toolbox Object - IPC Tab](#)

[Toolbox Object - MetaDataMap Tab](#)

[Toolbox Object - Menu Tab](#)

[Toolbox Object - History Tab](#)

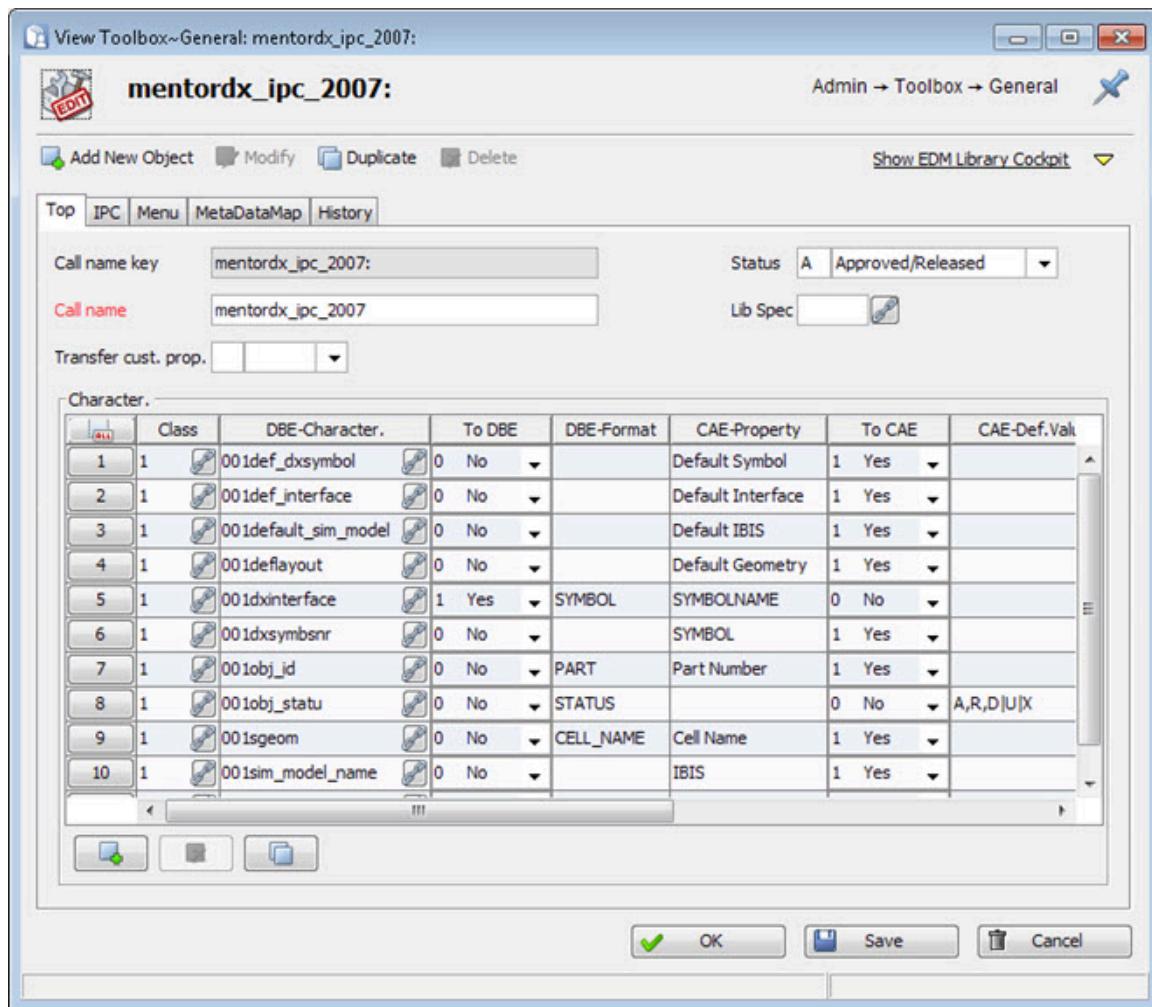
Toolbox Object - Top Tab

To access: Refer to “[Opening a Toolbox Information Window](#)” on page 264

All Toolbox objects contain a **Top** tab. The **Top** tab of the toolbox contains the main characteristics that identify the toolbox to a calling program, and the characteristic list box. The system automatically creates a value for the Call name key characteristic based on the value of the Call name and Lib Spec characteristics.

Description

Figure 139. Top Tab of the Toolbox



Entries in the characteristics list box specify the mapping and value conversion rules when transferring characteristic data between an EDM database and an IPC client. IPC clients such as Xpedition Designer use values in the Characteristics list box for instantiation, assignment, cross-probing, and synchronization operations. Many of the toolboxes in the Tools subcatalog do not contain information in the Characteristics list box, since the functionality supported by the toolbox does not require characteristic mapping or value conversion.



Note:

Toolboxes such as *mentordx_ipc_2007*, which the IPC software uses when transferring data between an EDM database and a schematic capture application, contain a minimum default set of CAE properties in the Characteristics list box. To transfer additional property values (for example, during instantiation), you must add rows to the Characteristics list box defining the CAE properties and how to transfer them. Although you should not eliminate any of the default rows (since certain application functionality requires the existence of those rows), you might need to edit some of the definitions to fit your naming conventions.



Note:

Because the EDM Library Cache Manager only extracts characteristic information from Component objects (class 1) in a database, it ignores the settings in the To DBE, DBE-Format, To CAE, and CAE-Def Value columns of the mentordx_updc toolbox. An exception is when you want to update the built-in Part Name, Part Label and Part Description attributes. Then, you can have either PART_NAME, PART_LABEL, or PART_DESC in the DBE-Format column.

You can configure transfers into both the part property and into the built-in part attribute.

For operations such as instantiation and assignment, Characteristic list box definitions are not specific to a particular catalog. Be careful not to map a dynamic characteristic to a CAE property if the same characteristic is in multiple catalog groups, as it could cause a conflict with other Characteristic list box definitions that transfer a value to the same CAE property.

For example, you add a row to the Characteristic list box to transfer the 000acme_voltage characteristic to the CAE property named VALUE. In the database, you use the 000acme_voltage characteristic in the Batteries catalog to store the battery voltage, but also to store the voltage rating in your Capacitor catalog group. The toolbox row transfers the appropriate value when a designer instantiates a battery symbol but, because you also have a row in the Characteristic list box to transfer the 000acme_capacitance value to the CAE property VALUE, you have created a conflict in the toolbox definition when instantiating a capacitor symbol.

Characteristics

Characteristics on the **Top** tab can be divided into:

- Main characteristics
- Character. List Box

Main Characteristics

- **Call Name** — The name of the toolbox or the internal program name of the client module, such as mentor_updc or mentordx_ipc_2007. Any Toolbox object always requires a Call Name value.
- **Status** — An option list with the following values and meanings:
 - **U (Under Construction)** — The object is under construction. Not all information is available and the object must not be used for production. Saving the object does not check mandatory fields.
 - **D (in Development)** — The object is still in development and not all information is available. The object can be used in designs, but not for production. Saving the object does not check mandatory fields.
 - **R (Restricted)** — The object is restricted to testing or preliminary use, or only for use in qualified projects. Saving the object checks mandatory fields and object references, but does not check mandatory fields in list boxes unless there is at least one list entry.
 - **A (Approved)** — The object is approved and has been released. It has been created correctly and tested. The object has been proven in practice. Saving the object checks

mandatory fields and object references. Does not check mandatory fields in list boxes unless there is at least one list entry.

- **X (Obsolete)** — The object is obsolete. Do not use the object.
- **Lib Spec** — The name of a Library Specification object in the database (refer to “Library Specification Object Class” in the *Xpedition EDM Library EDA Library Module Guide*).
- **Transfer cust. prop.** — Used by the part replacement functionality in the EDM Library interface to Xpedition Designer (controlled by the mentordx_ipc_2007 toolbox) and ignored by all other interfaces. A Yes value reads the Show in Part Replacement column. If that column also has a Yes value, the toolbox displays the characteristic in the Part Replacement pane (refer to “Replacing a Part” in the *Xpedition EDM Library Guide for Designers*). Setting Transfer prop. for Replacement to No ignores the values in the Show in Part Replacement column.

Character. List Box

The Characteristics list box contains the following columns:

- **Class** — The class number in the database that holds the name of the characteristic to transfer using the IPC. For example, a 1 denotes that the row pertains to a characteristic in the Components class.
- **DBE-Character** — The key identifier of the database characteristic.
- **To DBE** — A 0 (No) or 1 (Yes) value denoting whether to transfer the value of the property from the external program specified by the CAE-Property column and store it as a new value in the characteristic specified by the DBE-Character column.

The transfer direction specified by this column is opposite the transfer direction specified by the To CAE column.

- **DBE-Format** — The starting point for the assignment search. For a detailed description, refer to “Assignment Rows” on page 319. Only an assignment operation evaluates the DBE-Format column.
- **CAE-Property** — The name of the property inside the design tool (for example, PART_NO, VALUE or INSTPAR) corresponding to the database characteristic.
- **To CAE** — Specifies whether to transfer the characteristic value from the EDM database to replace a corresponding property value (specified by the CAE-Property column) on a schematic instance, as follows:
 - **0 (No)** — Do not transfer the characteristic value.
 - **1 (Yes)** — Transfer the characteristic value, even when the value field is empty.
 - **2 (Yes, non-empty)** — Transfer the characteristic value only if there is an actual value (that is, the value field of the characteristic is not empty).

The transfer direction specified by this column is opposite the transfer direction specified by the To DBE column.

- **CAE-Def. Value** — This column is used only by an assignment operation and has a special meaning as described in “[Assignment Rows](#)” on page 319.
- **Numeric** — This column is used by the mentordx_ipc and mentordx_ipc_2007 toolboxes when transferring data to the Xpedition Designer tool. The Numeric column defines to the IPC interface software whether to transfer the EDM Library Unit associated with the characteristic. There are three possible settings.
 - **No value or 0** — Transfers the value of a characteristic (such as the resistance value) to the external application, with both the data value and unit information coming from the database. Thus, a resistance value of 1k5 in the database displays as 1.500k in Xpedition Designer (that is, with trailing zeros and the ‘k’ unit designator).
 - **1** — Transfers the data value of a characteristic without its unit information. Without any formatting in Xpedition Designer, a value of 1k5 in the database then displays as 1500. Using the display formatting options of the external application, a design engineer can then format the value to 1.5k without trailing zeros.
 - **2** — Transfers the numeric value without trailing zeros and the unit. The format is the same as when the Numeric column has a ‘0’ setting, except without the zeros. Thus, a resistance value of 1k5 in a database displays as 1.5k in Xpedition Designer.
- **Synchronize** — The mentordx_ipc_2007 toolbox in uses the Synchronize column to specify which characteristics to compare and synchronize when auditing an Xpedition Designer design (refer to “Auditing and Synchronizing a Design” in the *Xpedition EDM Library Guide for Designers*).

A “(1) Yes” in the Synchronize column forces EDM Library to synchronize the property value on a schematic instance with the corresponding characteristic value in the database, even when the characteristic has no value. A “(2) Yes, non-empty” only synchronizes the property if the database characteristic has an actual value; if there is no value, the software excludes the characteristic from the synchronization and leaves the property value on the instance unchanged. The To DBE and the To CAE columns determine the direction of data transfer. Any characteristic to synchronize must be in the Component object class (that is, class 1). You cannot synchronize characteristics of type list frame or BLOB, or any of the following base properties:

 - Part Number
 - SYMBOLNAME
 - SYMBOL
 - Default Geometry
 - (OAT) Ref Designator
 - Ref designator
 - Cell Name

If the characteristic to synchronize is an unacceptable type, or a unpermitted base property, an error message occurs when saving the toolbox.

A "(0) No" does not check or synchronize the property value with the characteristic value. Any characteristic to synchronize must display in the Component object class (that is, class 1).

- **Show in Part Replacement** — A Yes in the column when the value of the Transfer cust prop characteristic is also Yes displays the characteristic in the Part Replacement pane during a part replacement operation. Only the EDM Library/Xpedition Designer interface that uses the mentordx_ipc_2007 toolbox recognizes the Show in Part Replacement column value.

Related Topics

[Opening a Toolbox Information Window](#)

[Toolbox Object - IPC Tab](#)

[Toolbox Object - MetaDataMap Tab](#)

[Toolbox Object - Menu Tab](#)

[Toolbox Object - History Tab](#)

[Defining Component Characteristic Values to Transfer to the Production Library Cache \[Xpedition EDM Library EDA Library Module Guide\]](#)

[Component Characteristic to PDB Property Mapping \[Xpedition EDM Library EDA Library Module Guide\]](#)

Toolbox Object - IPC Tab

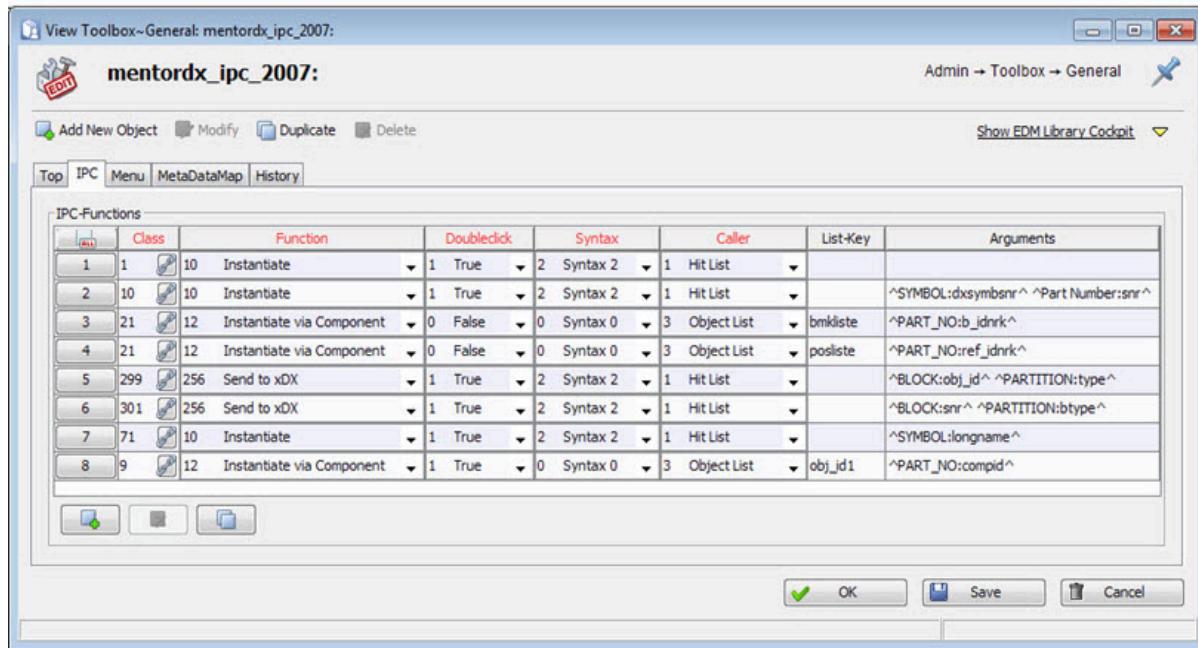
To access: Refer to [“Opening a Toolbox Information Window”](#) on page 264

The IPC connection to the Xpedition Designer application uses the **IPC** tab for instantiation and cross-highlighting operations. Customized client macro functions also can use the tab. Entries in this tab do not effect assignment operations.

Although the **IPC** tab is present on all Toolbox objects by default, programs using Toolbox objects residing in the Toolbox > Tools subcatalog usually ignore any information on the tab.

Description

Figure 140. IPC Tab of the Toolbox



Characteristics

The **IPC** tab contains the IPC Functions list box with the following characteristic columns:

- **Class** — The class number from which to instantiate or cross-highlight, or from which to send data to the RTF Print interface or to another customized macro function.
- **Function** — The following option values are available:



Note:

Only permanently running clients can use the PreOK and AfterOK functions.

- **5 Pre OK** — Used for custom macro functions. If a toolbox entry for PreOK is present, typing '5' in the PreOK Macro/CB input field of the **Macros** tab of an object class uses the toolbox to transfer data to the client function upon clicking the **OK** button.
- **6 After OK** — Used for custom macro functions. If a toolbox entry for AfterOK is present, typing '6' in the AfterOK Macro/CB input field in the **Macros** tab of an object class uses the toolbox to transfer data to the client function upon clicking the **OK** button.
- **8 Copy Master BOM** — Used to instantiate an item from a BOM into a schematic.
- **10 Instantiate** — Use the Instantiation function to define a toolbox entry for direct instantiation from the search results list or information window of the current object class.

- **12 Instantiate via Component** — Instantiates a selected component from a list of referenced components into a design.
 - **13 Compare Variants** — Compares the components on two variant BOMs and returns the differences.
 - **15 Update Partlist from DC** — EDM Library-IPC Design Capture only. Used to update the partlist from the Design Capture application.
 - **16 Cross-Highlight** — Used to cross-highlight a selected part from the **REF** tab of a part list and show that part in a schematic sheet.
 - **17 Cross-Highlight** — Used to cross-highlight a selected part from the **POS** tab of a part list and show that part in a schematic sheet.
 - **18 Edit Object, 19 Create Object, and 20 Release Object** — Used by action buttons in the information window to call default IPC clients for EDM Librarian functions.
 - **255 Extended Instantiate** and **256 Send to xDX** — Used by the EDM Library IPC interface to Xpedition Designer.
 - **901 Print Hitlist** (RTF Print client only) — Send one or several objects from the search results list to the RTF Print interface.
 - **902 Print Object:** (RTF Print client only) — Send data of one object from the information window to the RTF Print interface.
- **Doubleclick** — Used only for instantiation and defines whether or not EDM Library can transfer an instance to the design tool by double-clicking the instance inside the search results list or the information window.
 - **Syntax** — The Syntax column defines how to format the arguments specified in the Arguments column when transferring those arguments to the client function.
 - **Caller** — The Caller column informs the client about the caller of the client. The following values are available:
 - **HitList** — The search results list in Xpedition EDM Library Cockpit called the client.
 - **ObjectMacro** — An object macro called the client. An object macro can be either a PreOK or an AfterOK macro (Function 5 or 6), or one of the RTF Print interface functions (901 or 902).
 - **ObjectList** — The client function is called from a list inside an object information window.
 - **List-Key** — A value in the List-Key column is necessary to transfer data from an object information window to the client function (that is, if you typed 'Object List' in the Caller column). In this case, type the main key of the list that contains the data to transfer inside the List-Key column without the class number in front and without carets (for example, bmklst).

- **Arguments** — The arguments transferred to the client function (the Syntax column defines how EDM Library transfers the arguments). Standard clients like the Xpedition Designer interface require specific arguments and are pre-configured. The arguments transferred to a customized client function can be any number and any characteristics from EDM Library.

Related Topics

[Opening a Toolbox Information Window](#)

[Toolbox Object - Top Tab](#)

[Toolbox Object - MetaDataMap Tab](#)

[Toolbox Object - Menu Tab](#)

[Toolbox Object - History Tab](#)

Toolbox Object - Menu Tab

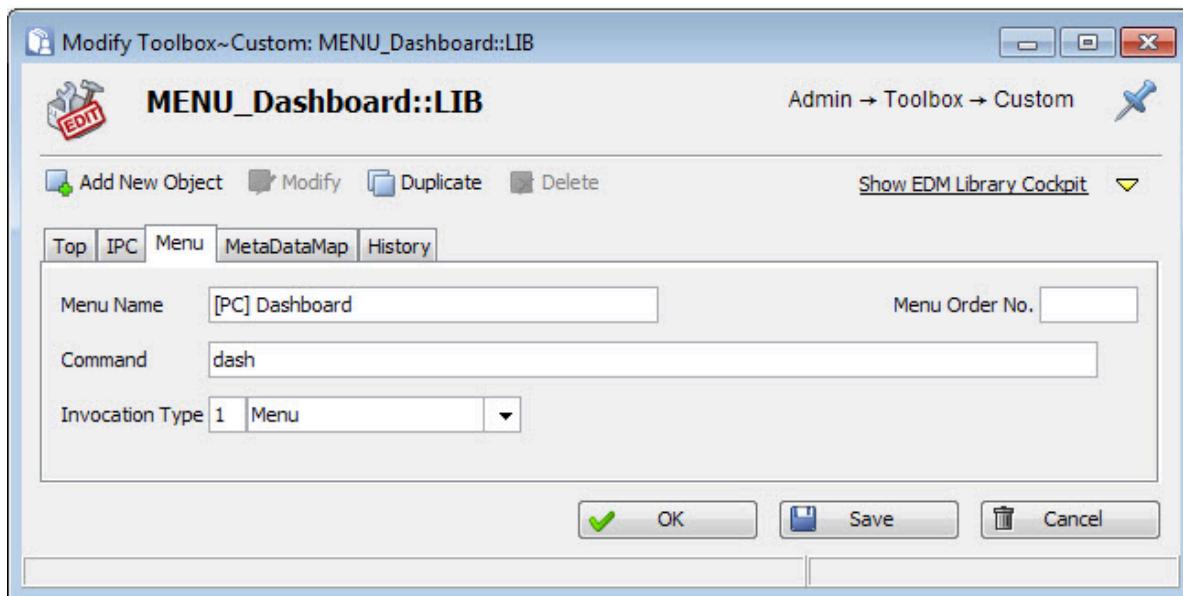
To access: Refer to “[Opening a Toolbox Information Window](#)” on page 264

The **Menu** tab is present on all Toolbox objects by default. The **Menu** tab of the Toolbox adds a item to the **Tools** pulldown menu in Xpedition EDM Library Cockpit that executes a command to run an external program.

Description

[Figure 141](#) shows the **Menu** tab of a custom toolbox that adds an item to the **Tools** pulldown menu to invoke the Dashboard application.

Figure 141. Menu Tab of the Toolbox



Characteristics

- **Menu Name** — A string value specifying the name to display in the **Tools** pulldown menu of EDM Library Cockpit.
- **Menu Order No.** — The sort order of the entries in the menu.
- **Command** — The client call (in [Figure 141](#), the client call is dash).
- **Invocation Type** — How to start the client.
 - **0 (None)** — Does not start the client automatically when logging into EDM Library Cockpit. To manually start the client, an administrator might configure an action button in the EDM Library Cockpit user interface.
 - **1 (Menu)** — Starts the command in the Command field through the **Tools** pulldown menu in EDM Library Cockpit.
 - **2 (Startup)** — Automatically starts the command in the Command field upon successfully logging into EDM Library Cockpit.
 - **3 (Startup - Only Classic)** — No longer used. EDM Library applications (including EDM Library Cockpit) ignore any commands with invocation type 3.

Related Topics

[Opening a Toolbox Information Window](#)

[Toolbox Object - Top Tab](#)

[Toolbox Object - IPC Tab](#)

[Toolbox Object - MetaDataMap Tab](#)

[Toolbox Object - History Tab](#)

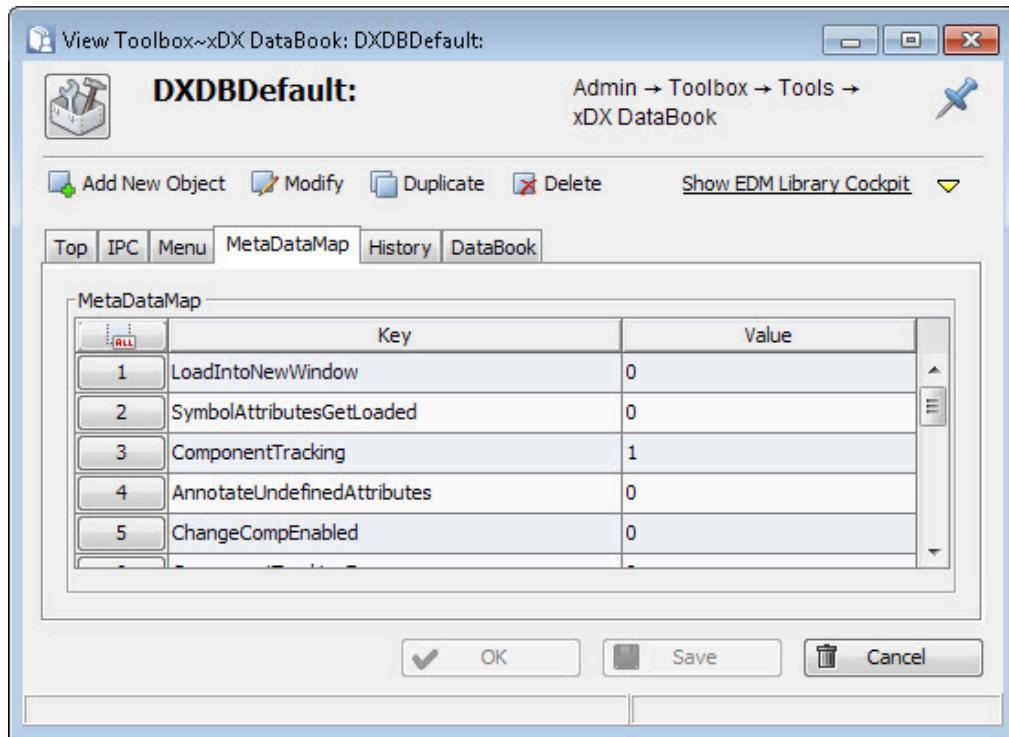
Toolbox Object - MetaDataMap Tab

To access: Refer to “[Opening a Toolbox Information Window](#)” on page 264

The **MetaDataMap** tab is present on all Toolbox objects by default. The **MetaDataMap** tab defines a specific key name and value that has meaning to a Java program that processes the name/value pairs. The Compliance Manager and EDM Library Connector applications make use of information on the **MetaDataMap** tab.

Description

Figure 142. MetaDataMap Tab of the Toolbox



Characteristics

When used by the Compliance Manager software, information on the **MetaDataMap** tab describe keys to configure simple character-based processing (refer to “Simple Character-Based Processing Configuration” in the *Xpedition EDM Library Compliance Management Module Guide*).

When used by EDM Library Connector software, the **MetaDataMap** tab specifies the settings of Databook configuration options (refer to “[Databook Toolbox - MetaDataMap Tab](#)” on page 286).

When used with the EDM Supplychain software, the **MetaDataMap** tab defines parameters for specific content provider.

Related Topics

[Opening a Toolbox Information Window](#)

[Toolbox Object - Top Tab](#)

[Toolbox Object - IPC Tab](#)

[Toolbox Object - Menu Tab](#)

[Toolbox Object - History Tab](#)

[Databook Toolbox - MetaDataMap Tab](#)

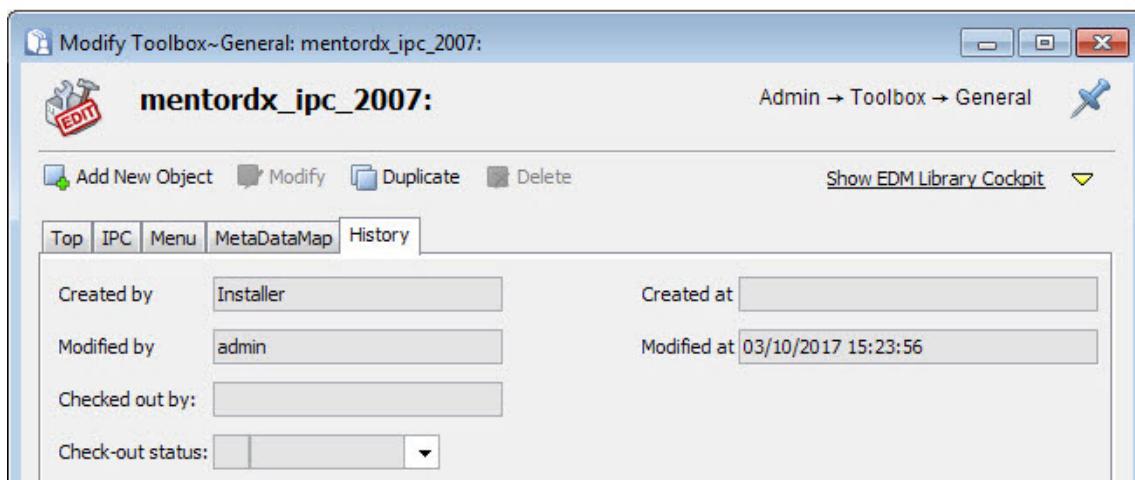
Toolbox Object - History Tab

To access: Refer to “[Opening a Toolbox Information Window](#)” on page 264

The **History** tab contains system-maintained information about the Toolbox object.

Description

Figure 143. History Tab of the Toolbox



Characteristics

The **History** tab contains these system-generated values:

- **Created by** — The user who created this object.
- **Modified by** — The user who last modified the object.
- **Created at** — The date the object was created.
- **Modified at** — The date of the last modification.
- **Checked out by** — The user that has the object checked out. Checking out an object prevents others from editing that object while checked out.
- **Check-out status** — A flag that, when set to “1” indicates the object is checked out and cannot be edited by a user other than the one identified by the Checked out by characteristic.



Note:

The EDM Library software sets and updates all characteristic data on the **History** tab. It is not possible to edit this data manually.

Related Topics

[Opening a Toolbox Information Window](#)

[Toolbox Object - Top Tab](#)

[Toolbox Object - IPC Tab](#)

[Toolbox Object - MetaDataMap Tab](#)

[Toolbox Object - Menu Tab](#)

Part Replacement Toolboxes

Performing an EDM Library part replacement in Xpedition Designer uses information on the tabs of a part replacement toolbox to determine the Xpedition Designer property values to keep, copy, or replace on an instance, and the cell name search restrictions to apply. Part replacement toolboxes reside in the **Toolbox > Tools > Part Replace** subcatalog and are denoted by the presence of the **Part Replacement** and **Replacement Restrictions** tabs.



Note:

The Part Replacement and Replacement Restrictions tabs are only available if the toolbox resides in the Toolbox > Tools > Part Replace catalog.

[Part Replacement Toolbox - Part Replacement Tab](#)

[Part Replacement Toolbox - Replacement Restrictions Tab](#)

Part Replacement Toolbox - Part Replacement Tab

To access: Refer to “[Opening a Toolbox Information Window](#)” on page 264

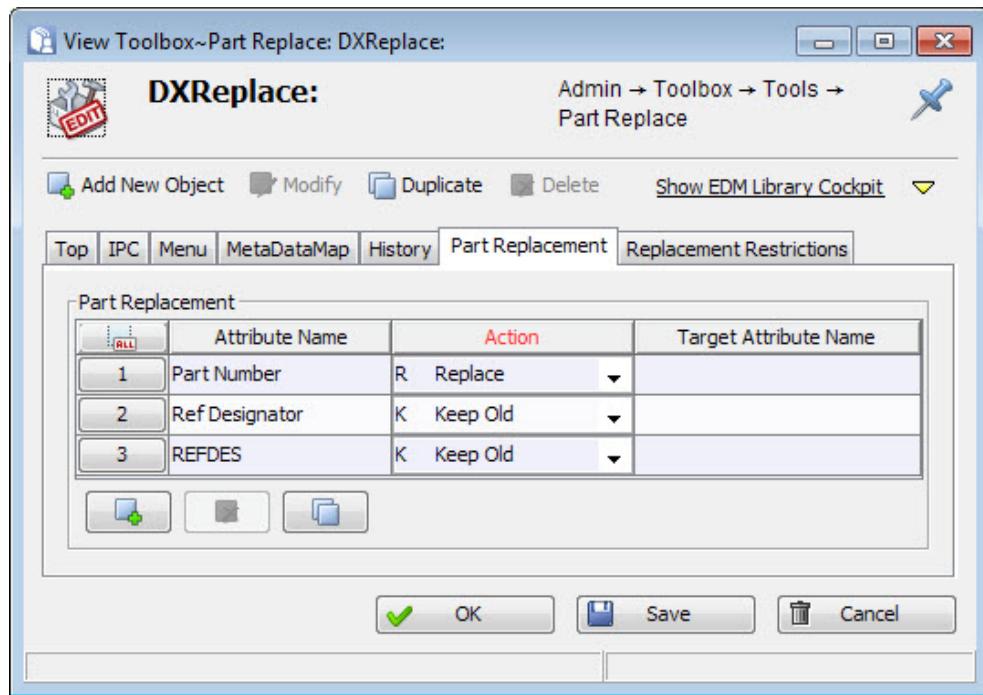
The **Part Replacement** tab defines which property attribute values to replace or copy when performing a part replacement operation in Xpedition Designer.

Description

Replacing an existing instance in an Xpedition Designer schematic replaces the symbol of that instance and transfers the values of the database characteristics listed in the Characteristics list box on the **Top** tab of the mentordx_ipc_2007 toolbox from the database to the schematic (the same process as during a normal instantiation). Part replacement then applies replacement values for attributes (properties) defined on the **Part Replacement** tab of a parts replacement toolbox.

The rows in the Part Replacement list box define which part property values to replace, and which values to copy to a new target property. If the **Part Replacement** tab does not define the attribute with a Keep action, and the attribute is not listed in the **Top** tab of the mentordx_ipc_2007 toolbox, a part replacement operation removes it from the instance.

Figure 144. Part Replacement Tab of the Default DXReplace Toolbox



An administrator can define multiple part replacement toolboxes, each with a different list of part replacement attributes and actions. From the part replacement pane, a designer can choose the part replacement toolbox to use during part replacement.

A default part replacement toolbox named DXReplace is available with the default rows shown in [Figure 144](#). Copy and modify the DXReplace toolbox to add additional attributes to update during a part replacement operation.

Characteristics



Note:

Do not delete the three default rows on the **Part Replacement** tab for the Ref Designator, REFDES, and Part Number attributes, which must exist for part replacement functionality to work correctly.

When adding rows to the Part Replacement list box, complete the column values as follows:

- **Attribute Name** — Specify the name of the property attribute as defined in Xpedition Designer.
- **Action** — Specify an action of either Keep Old, Replace, or Copy.

- **Keep Old** — Keeps the property attribute value on the existing (old) part instance, rather than replacing it with a new value from the new instance. Excluding an attribute from the list is the same as specifying the **Keep Old** action.
 - **Replace** — Updates the property attribute value on the part instance with a new value from the new instance.
 - **Copy** — Writes the old property attribute value to a new target attribute, using the value specified in the Target Attribute Name column.
- **Target Attribute Name** — The name of a new target attribute when a Copy action is specified.

Related Topics

[Part Replacement Toolbox - Replacement Restrictions Tab](#)

[Replacing a Part \[Xpedition EDM Library Guide for Designers\]](#)

[Default Toolboxes and Hierarchy](#)

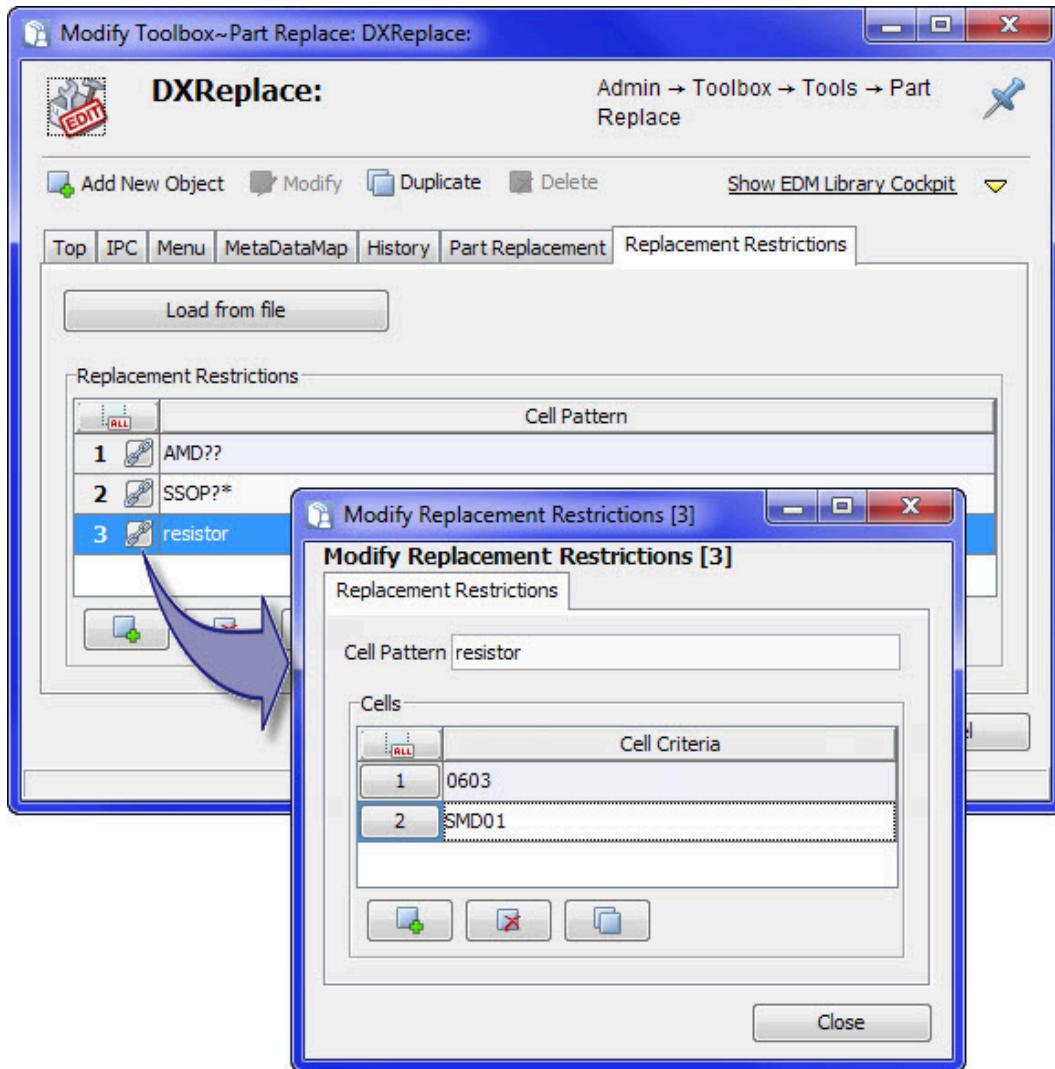
Part Replacement Toolbox - Replacement Restrictions Tab

To access: Refer to “[Opening a Toolbox Information Window](#)” on page 264

The **Replacement Restrictions** tab defines a list of cell name search restrictions to apply when using the EDM Library part replacement functionality in Xpedition Designer.

Description

Figure 145. Replacement Restrictions Tab of the Default DXReplace Toolbox



If the list on the **Replacement Restrictions** tab contains no rows, then the software does not use the cell name as a search criteria and returns a list of parts to the Part Replacement pane regardless of the type of cells used by those parts.

When at least one row exists on the **Replacement Restrictions** tab, then a part replacement operation compares the value of the Cell Name property of any potential replacement part against the list of cell patterns. Only those parts that have a Cell Name property value matching a listed search pattern and referencing one of the valid Cell objects for that pattern display in the Part Replacement pane. If a part does not have a Cell Name value defined in Xpedition Designer or the selected replacement part does not have a Cell Name value that matches one of the listed cell patterns, then the software does not permit the part as a replacement.



Note:

The Cell Pattern field accepts regular Input Pattern syntax, while the Cell Criteria column accepts a search criteria pattern. With Input Pattern syntax, * means zero or more occurrences of the preceding character. and ? means any printable character.

Characteristics

- **Replacement Restrictions list box** — A list box with a single column named Cell Pattern, which contains an EDM Library regular expression that matches the Cell Name property value in Xpedition Designer.

[Figure 145](#) on page 282 shows the following three cell name pattern matches, using Input Pattern syntax:

- **AMD??** — Only returns parts that have a five letter Cell Name property value that starts with “AMD”.
- **SSOP?*** — Returns parts that have a Cell Name property value of any length, but whose first four letters are SSOP and the name has zero or more subsequent printable characters.
- **resistor** — Only returns parts with a Cell Name property value matching “resistor”.

Clicking a reference button next to a row enables you to specify the required EDM Library Cell object reference for that pattern as either an exact match or a regular expression in search criteria format. In [Figure 145](#), the Replacement Restrictions list box is configured so that a part with a Cell Name value of “resistor” must reference a 0603 or SMD01 Cell object to be a valid replacement for a resistor. Any other resistor parts that reference different types of Cell objects do not display in the search results list in the Part Replacement pane.

As another example, if a digital part that references any DIP14 cell (such as DIP14, DIP14C, and so on) is acceptable, but material is not a criteria, then you might specify a regular expression of “DIP14*” in the Cells list rather than an exact cell name.

- **Load from File button** — Loads a set of replacement restrictions from an ASCII file. The file must be formatted as follows:

```
<cell_pattern> <cell_criteria>
<cell_pattern> <cell_criteria>
...

```

`<cell_pattern>` is an EDM Library regular expression that matches the Cell Name property value of the selected part (corresponding to the Cell Pattern column in the Replacement Restrictions list box). `<cell_criteria>` is the EDM Library search criteria for the referenced Cell object. You can use the logical OR symbol (|) to separate multiple search criteria for the same cell pattern, which is then presented as separate rows in the Cell Criteria list box.

For example, an ASCII file might list the following restrictions, resulting in the creation of three lines in the Replacement Restrictions list box:

```
resistor 0603|SMD01
AMD?? SOIC28|SOIC28C|SOIC28E
Op?* SOIC16*
```

Related Topics

[Regular Input Pattern Syntax](#)

[Part Replacement Toolbox - Part Replacement Tab](#)

[Replacing a Part \[Xpedition EDM Library Guide for Designers\]](#)

Databook Toolboxes

A Databook toolbox maps the characteristics and catalogs stored in the EDM database to the property names and classification structure that makes sense to Databook users. Databook toolboxes reside in the **Toolbox > Tools > Databook** subcatalog and are denoted by the presence of a **Databook** tab.



Note:

The **DataBook** tab is only available if the toolbox resides in the Toolbox > Tools > Databook catalog.

The Databook application provides design engineers using Xpedition Designer a way to search for and select components when placing parts, and to verify and update components contained in a single schematic or in an entire hierarchical design. A librarian or designer can configure Databook to use component data from many different ODBC-compliant sources, including an EDM database. As described in “EDM Library Connector” in the *Xpedition EDM Library EDA Library Module Guide*, the EDM Library Connector web application provides the connection conduit by which the Databook application can access information stored in an EDM database as a data source.

Before Databook can use the data from the database, the following items must exist:

- A Databook toolbox configuration residing in the Toolbox > Tools > Databook subcatalog.
-



Note:

Instead of creating a Databook toolbox configuration, you can auto-generate a **.dbc** file from the EDM Library Connector web application based on the content of the **mentordx_ipc_2007** toolbox. Such a **.dbc** file provides broad access to all database libraries based on the EDM Library component taxonomy, whereas a Databook toolbox enables more configuration options.

The **Databook** tab of a Databook Toolbox object stores the required configuration information, which EDM Connector then uses to display the information in Databook. The **MetaDataMap** tab of each Databook toolbox configuration provides settings for Databook options.

- A configured EDM Library Connector web application that uses the toolbox to service requests from Databook.

[Databook Toolbox - MetaDataMap Tab](#)

[Databook Toolbox - Databook Tab](#)

[Creating the Databook Toolbox Object](#)

Databook Toolbox - MetaDataMap Tab

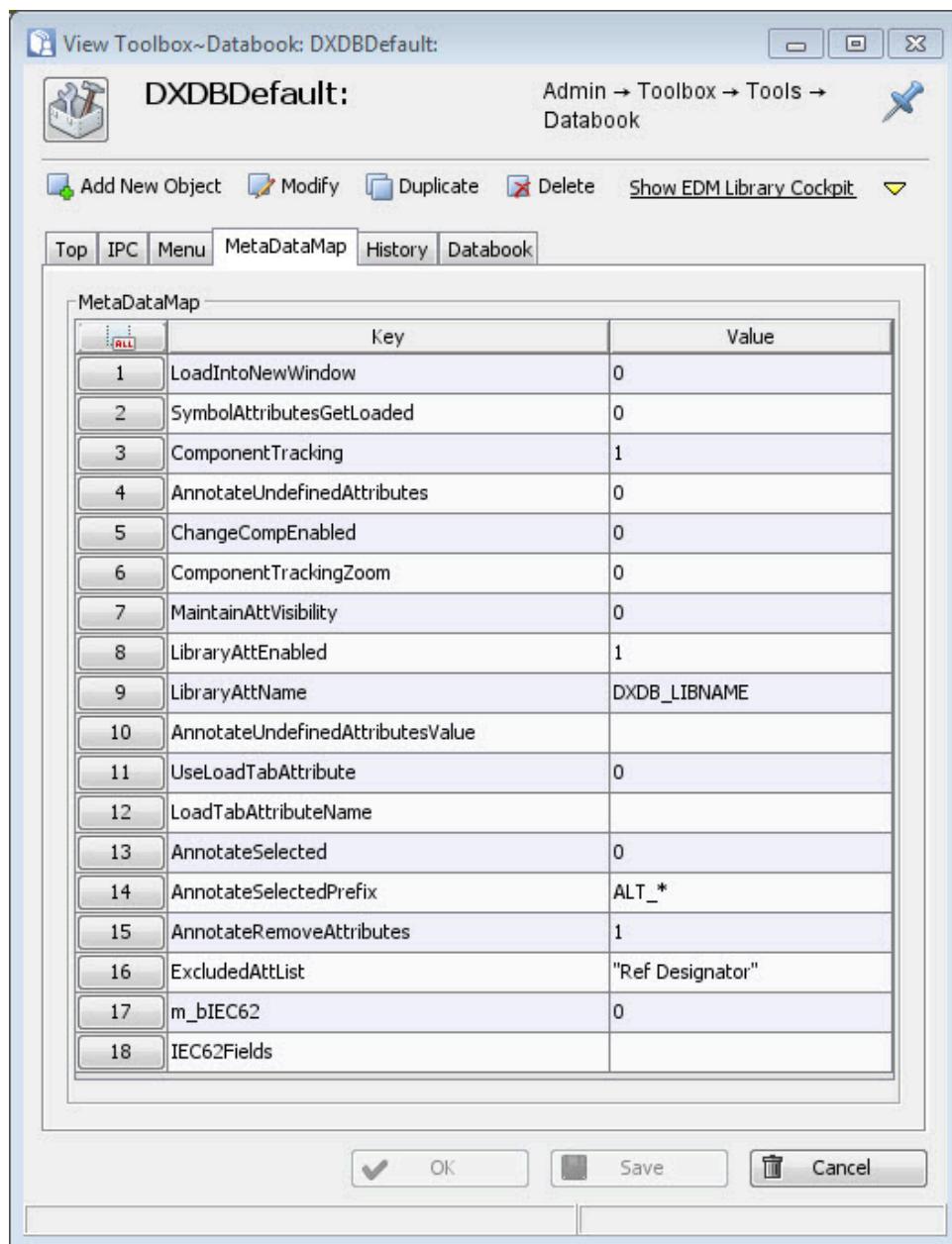
To access: Refer to “[Opening a Toolbox Information Window](#)”

Entries on the **MetaDataMap** tab of a Databook Toolbox object provide settings that correspond to most options found on the **Properties** tab of the native Databook configuration editor. You cannot set all of the Databook configuration editor options with the **MetaDataMap** tab.

Description

[Figure 146](#) shows the keys and default values on the **MetaDataMap** tab of a Databook Toolbox object.

Figure 146. MetaDataMap Tab of a Databook Toolbox



Fields

The following table lists the entry in the Key column, the corresponding Databook configuration editor **Properties** tab option, and the definition.

| Key | Corresponding Databook Configuration Editor Properties Tab Option | Description |
|-----------------------------|-------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| AnnotateRemoveAttributes | “Remove Stale Properties from Component” checkbox | <p>Removes properties from the component when annotating properties from Databook into an existing component in Xpedition Designer.</p> <p>Values are 0 (not enabled) or 1 (default).</p> <p>Refer to the Component Annotation table in “Databook Tool - Configure Dialog Box - Properties Tab” in the <i>Xpedition Designer and Xpedition System Designer Reference</i>.</p> |
| AnnotateSelected | “Enable Annotation to Selected Component” checkbox | <p>Annotates properties to designated components in Xpedition Designer.</p> <p>Values are 0 (not enabled; default) or 1.</p> <p>Refer to the Component Annotation table in “Databook Tool - Configure Dialog Box - Properties Tab” in the <i>Xpedition Designer and Xpedition System Designer Reference</i>.</p> |
| AnnotateSelectedPrefix | N/A | <p>Use AnnotateSelectedPrefix in conjunction with the AnnotateSelected attribute.</p> <p>Refer to “Databook Tool Configuration File Format” in the <i>Xpedition Designer and Xpedition System Designer Reference</i>.</p> |
| AnnotateUndefinedAttributes | “Annotate Undefined Values” checkbox | <p>When a property has no value specified in the database (the default), annotates any value (such as Blank, Empty, or Null) matching the undefined value specified by AnnotateUndefinedAttributesValue.</p> <p>Values are 0 (values not annotated; default) or 1.</p> <p>Refer to the Component Instantiation/Annotation table in “Databook Tool - Configure Dialog Box - Properties Tab” in the <i>Xpedition Designer and Xpedition System Designer Reference</i>.</p> |

| Key | Corresponding Databook Configuration Editor Properties Tab Option | Description |
|----------------------------------|--------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | | <i>Xpedition Designer and Xpedition System Designer Reference.</i> |
| AnnotateUndefinedAttributesValue | “Undefined Value” field beneath the “Annotate Undefined Values” checkbox | <p>The software uses values in this entry if AnnotateUndefinedAttributes is enabled.</p> <p>Refer to “Databook Tool - Configure Dialog Box - Properties Tab” in the <i>Xpedition Designer and Xpedition System Designer Reference</i>.</p> |
| ChangeCompEnabled | “Change Component Enabled” checkbox | <p>Changes the component when specifying a new symbol.</p> <p>Values are 0 (not enabled; default) or 1.</p> <p>Refer to the Component Annotation table in “Databook Tool - Configure Dialog Box - Properties Tab” in the <i>Xpedition Designer and Xpedition System Designer Reference</i>.</p> |
| ComponentTracking | “Enable Cross Probing” checkbox | <p>Highlights the relevant components on the schematic as Databook processes each group.</p> <p>Values are 0 (not enabled) or 1 (default).</p> <p>Refer to the Cross Probing table in “Databook Tool - Configure Dialog Box - Properties Tab” in the <i>Xpedition Designer and Xpedition System Designer Reference</i>.</p> |
| ComponentTrackingZoom | “Zoom On Cross Probing” checkbox | <p>Zooms in on the component when selected in Databook.</p> <p>Values are 0 (not enabled; default) or 1.</p> <p>Refer to the Cross Probing table in “Databook Tool - Configure Dialog Box - Properties Tab” in the <i>Xpedition Designer and Xpedition System Designer Reference</i>.</p> |
| ExcludedAttList | “Excluded Properties when Loading” field | <p>Excludes the specified comma separated list of properties when loading a component from Xpedition Designer into Databook.</p> <p>Refer to the Component Loading table in “Databook Tool - Configure Dialog Box - Properties Tab” in the <i>Xpedition Designer and Xpedition System Designer Reference</i>.</p> |

| Key | Corresponding Databook Configuration Editor Properties Tab Option | Description |
|-----------------------|--------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | | <i>Xpedition Designer and Xpedition System Designer Reference.</i> |
| IEC62Fields | List field beneath the “Use European Notation for Property” checkbox | Used in conjunction with the m_bIEC62 entry. The values can be a comma separated list of properties.

The default is no value. |
| LibraryAttEnabled | “Annotate Special Library Property” checkbox | Annotates components with a property whose value defines the database library name.

Values are 0 (not enabled) or 1 (default).

Refer to the Component Instantiation/Annotation table in “Databook Tool - Configure Dialog Box - Properties Tab” in the <i>Xpedition Designer and Xpedition System Designer Reference</i> |
| LibraryAttName | “Property Name” field beneath the “Annotate Special Library Property” checkbox | The value is the name of the property that holds the library name and is only used when LibraryAttEnabled is enabled.

The default value is “DXDB_LIBNAME”. |
| LoadIntoNewWindow | “Load Components into New Window” checkbox | Loads components into a new window instead of the open Search window.

Values are 0 (not enabled; default) or 1.

Refer to the Component Loading table in “Databook Tool - Configure Dialog Box - Properties Tab” in the <i>Xpedition Designer and Xpedition System Designer Reference</i> . |
| LoadTabAttributeName | “Property Name” field beneath the “Use a property value for the tab string” checkbox | The value is the name of a property and is only used when UseLoadTabAttribute is enabled. |
| MaintainAttVisibility | “Maintain Property Visibility” checkbox | Maintains the visibility of existing properties.

Values are 0 (not enabled; default) or 1.

Refer to the Component Instantiation/Annotation table in “Databook Tool - Configure Dialog Box - Properties Tab” in the |

| Key | Corresponding Databook Configuration Editor Properties Tab Option | Description |
|---------------------------|-------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | | <i>Xpedition Designer and Xpedition System Designer Reference</i> |
| SymbolAttributesGetLoaded | “Symbol Properties Get Loaded” checkbox | <p>Loads symbol properties when loading a component into Databook from Xpedition Designer.</p> <p>Values are 0 (not enabled; default) or 1.</p> <p>Refer to the Component Loading table in “Databook Tool - Configure Dialog Box - Properties Tab” in the <i>Xpedition Designer and Xpedition System Designer Reference</i>.</p> |
| UseLoadTabAttribute | “Use a Property Value for the Tab String” checkbox | <p>Selects a property type specified when loading a component into Databook from Xpedition Designer. By default, the Load tabs in the Databook search window use “LoadTabAttributeName”.</p> <p>Values are 0 (not enabled; default) or 1.</p> <p>Refer to the Component Loading table in “Databook Tool - Configure Dialog Box - Properties Tab” in the <i>Xpedition Designer and Xpedition System Designer Reference</i>.</p> |
| m_bIEC62 | “Use European Notation for Property” checkbox | <p>A value of 0 (not enabled; default) disables the IEC62 column in DataBook Configurator.</p> <p>A value of 1 enables the IEC62 column in DataBook Configurator, permitting property selection.</p> <p>Refer to the Component Instantiation/Annotation table in “Databook Tool - Configure Dialog Box - Properties Tab” in the <i>Xpedition Designer and Xpedition System Designer Reference</i>.</p> |

Related Topics

[Databook Toolbox - Databook Tab](#)

Databook Toolbox - Databook Tab

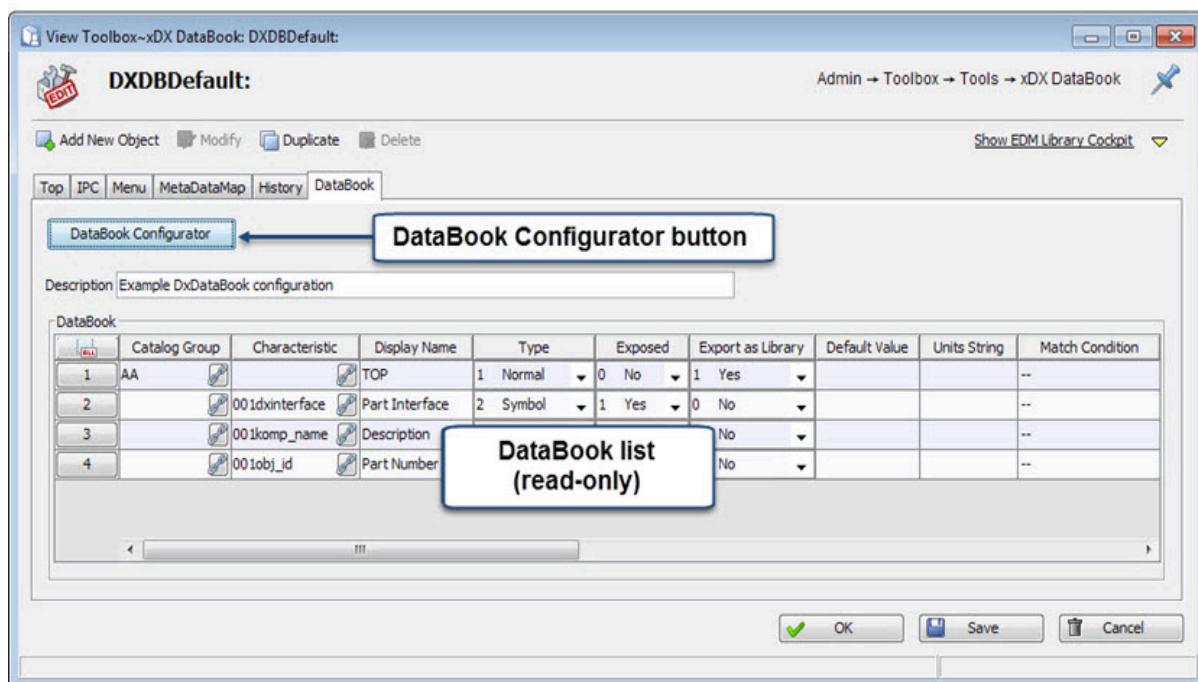
To access: Refer to “[Opening a Toolbox Information Window](#)”

Entries in the **Databook** tab of a Databook Toolbox object provide settings that correspond to most of the options on the **Libraries** tab of the native Databook configuration editor, with only a few columns specific to EDM Library data.

Description

Because of the large number of columns in the list, [Figure 147](#) does not show all columns on the **DataBook** tab.

Figure 147. DataBook Tab of a Databook Toolbox Object

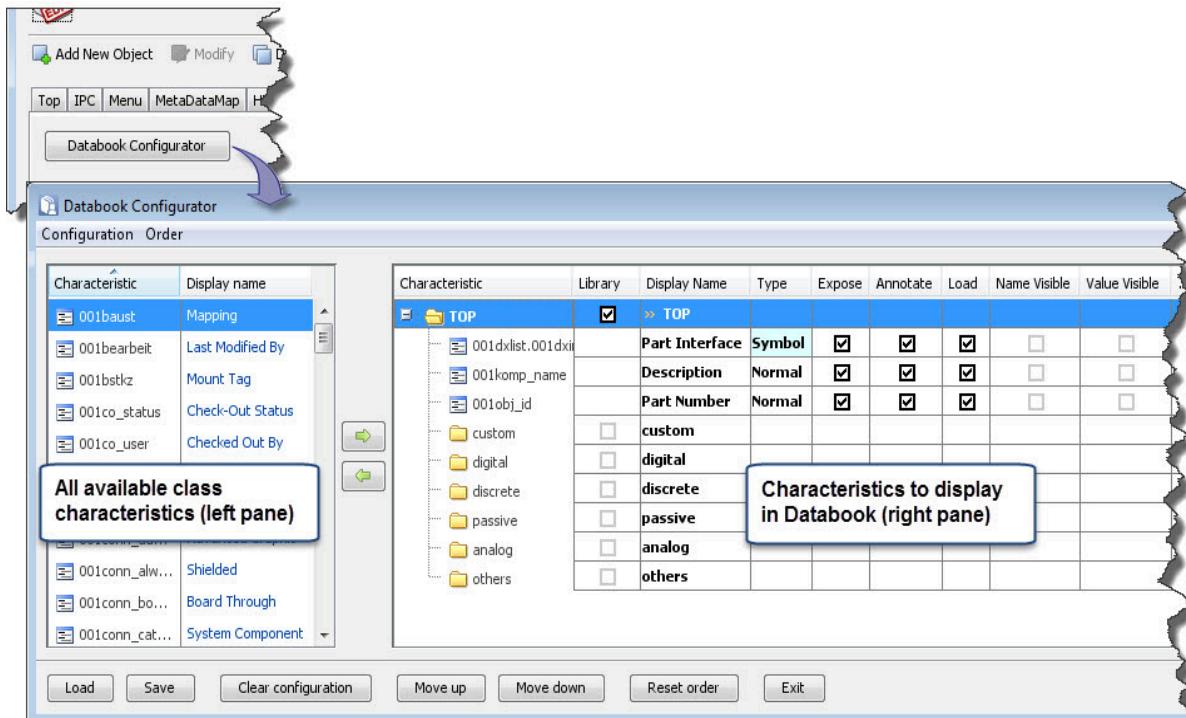


Note:

To edit the columns in the Databook list, you must use the Databook Configurator.

The **Databook Configurator** button launches an editor ([Figure 148](#)) that an administrator or librarian can use to create the Databook toolbox configuration. The Databook Configurator looks similar to the **Libraries** tab of the Configure dialog box in Databook (refer to “[Databook Tool - Configure Dialog Box - Libraries Tab](#)” in the *Xpedition Designer and Xpedition System Designer Reference*). Many columns in the Databook Configurator have the same names and set of value choices.

Figure 148. Databook Configurator



Fields

You use the Databook Configurator to map the catalog and characteristic data from an EDM database to Databook libraries and properties, and specify how Databook should use the property in Xpedition Designer.

The left pane contains a list of all class characteristics and their display names. You can choose which characteristics to move to the right pane and define for display in Databook. Any characteristics you move to the right pane show in the DataBook list on the **Databook** tab.

The following table lists the columns in the right pane of the configuration editor, and corresponding columns in the Databook list frame. When column names and value choices are the same in the configuration editor and the Databook Configure dialog box (for example, the Type column), refer to “Editing Component Library Table Properties” in the *Xpedition Designer and Xpedition System Designer Administrator’s Guide* for detailed value definitions and usage.

| Column Name in Databook Configuration Editor | Column Name in the Databook List Frame | Description |
|----------------------------------------------|----------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Characteristic | Catalog Group
Characteristic | <p>The Characteristic column displays the hierarchy of component library catalogs and dynamic characteristics from the EDM database, letting you choose and configure which catalogs and characteristics to show in Databook.</p> <p>The list frame lists the catalog group identifier in the Catalog Group column (for example, AAAIAL) and the identifier of a dynamic characteristic (if any) contained in that catalog in the Characteristic column.</p> |

| Column Name in Databook Configuration Editor | Column Name in the Databook List Frame | Description |
|-----------------------------------------------------|-----------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Library | Export as Library | <p>Checking the Library checkbox on a catalog group row displays the catalog as a library in Databook. The Library checkbox is not available in rows containing characteristics.</p> <p>The list frame represents the checkbox setting as a Yes/No value in the Export as Library column.</p> |
| Display Name | Display Name | <p>Defines the library or property name as it should be shown in Databook. The default value is the catalog label or characteristic label from the EDM database.</p> <p>This column corresponds to Field Name in the Databook Configure dialog box.</p> |
| Type | Type | <p>When the selected row is for a EDM Library characteristic, the field type values are the same as the Databook Type values of Normal, Symbol, Document, or Unused. The Type field is not active when the selected row is for a catalog group.</p> <p>This column corresponds to Field Type in the Databook Configure dialog box.</p> |
| Expose | Exposed | <p>Checking Exposed in the DataBook Configurator includes the selected characteristic as a property in Databook. The Expose checkbox is not available in rows containing catalog groups.</p> <p>The list frame represents the checkbox setting as a Yes/No value in the Exposed column.</p> |
| Annotate | Exclude when Annotating | <p>Checking Annotate causes the characteristic (property) to be shown in the Xpedition Designer property window for placed components. The list frame represents the checkbox setting as a Yes/No value in the Exclude when Annotating column (No if checked, Yes if not checked).</p> <p>This column corresponds to Annotate in the Databook Configure dialog box.</p> |
| Load | Exclude when Loading | <p>Checking Load loads the characteristic (property) during verification. The list frame represents the checkbox setting as a Yes/No value in the Exclude when Loading column (No if checked, Yes if not checked).</p> <p>This column corresponds to Load in the Databook Configure dialog box.</p> |
| Name Visible | Name Visible | <p>Checking Name Visible shows the display name of the characteristic on the schematic with placed components. The list frame represents the checkbox setting as a Yes/No value.</p> <p>This column corresponds to Name Visible in the Databook Configure dialog box.</p> |

| Column Name in Databook Configuration Editor | Column Name in the Databook List Frame | Description |
|----------------------------------------------|----------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Value Visible | Value Visible | <p>Checking Value Visible shows the characteristic (property) value on the schematic with placed components. The list frame represents the checkbox setting as a Yes/No value.</p> <p>This column corresponds to Value Visible in the Databook Configure dialog box.</p> |
| Units | Units String | <p>Specifies the appropriate string value to append to the magnitude.</p> <p>This column corresponds to Units in the Databook Configure dialog box.</p> |
| VM Match Conditions | Match Condition | <p>Used only by Variant Manager. During a property comparison, the Variant Manager uses the VM Match Condition setting to locate a matching part to replace in the variant. Test conditions are for either numeric or string values ("=" to test numeric values, "like" to perform a string compare, or "--" to not specify a condition).</p> |
| Magnitude | Magnitude | <p>When the characteristic value is a number, Magnitude defines the appropriate magnitude indicator to append to the value (for example, u for micro, m for milli, K for Kilo, and so on). Databook only uses the Valid Magnitude column when the Magnitude value is set to "automatic".</p> <p>This column corresponds to Magnitude in the Databook Configure dialog box.</p> |
| Valid Magnitude | Valid Magnitude | <p>When the Magnitude column value is "automatic", Valid Magnitude specifies the valid magnitude to append to the characteristic value.</p> <p>This column corresponds to Valid Magnitude in the Databook Configure dialog box.</p> |
| IEC62 | IEC62 | <p>Checking IEC62 shows units in the IEC62 format for components. The list frame represents the checkbox setting as a Yes/No value.</p> <p>This column corresponds to IEC62 in the Databook Configure dialog box. Note that m_bIEC62 on the MetaDataMap tab must be enabled before you are able to check the IEC62 box in the Toolbox Configuration Editor.</p> |

The Databook list box contains columns whose values are set by software, which you cannot edit, and that do not display in the Databook Configurator.

| Column Name in the Databook List Frame (does not show in the Configuration Editor) | Description |
|------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Required | No longer used and is only present for backward compatibility with earlier software versions of Databook. The software automatically sets the column value to No. |
| Add as OAT | When Occurrence Attributes (OATs) have been enabled in an Xpedition Designer design, the Add as OAT column specifies whether to process the OAT value when instantiating the component in a hierarchical design. Because Xpedition Designer no longer uses OAT values and the OAT column is no longer present in the Databook Configure dialog box, the software automatically sets the column value to No. Current versions of Xpedition Designer do not use the setting, and the Add as OAT value is only present for backward compatibility. |
| Position | Stores the position of the corresponding characteristic in the Databook Configurator. Using the Up or Down buttons to move the relative location of a characteristic in the Databook Configurator writes a new value to the Position column upon saving. |

The Databook Configurator has the action buttons shown in the following table.

| Button | Description |
|-------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Load | Populate the editor window with static characteristic and dynamic characteristics attached to catalog groups, and attach default values. Loading a configuration merges entries with the data model to ensure proper configuration, even when some characteristics are disconnected from catalog groups or removed, or when there are new characteristics in the data model. |
| Save | Write the entire structure to the DataBook list box on the Toolbox object. |
| Clear Configuration | Reload the editor window with the set of default values. |
| Move Up and Move Down | Move a selected characteristic up or down in the Databook Configurator to change where it displays in the Databook property list. |
| Reset Order | Restore the list to the original state, without accepting changes made by the Move Up or Move Down buttons. |

Related Topics

[Databook Toolbox - MetaDataMap Tab](#)

[Creating the Databook Toolbox Object](#)

Creating the Databook Toolbox Object

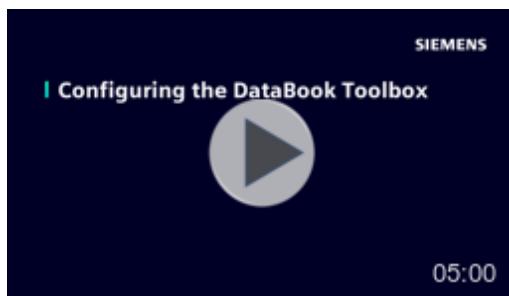
For designers to access library data in the database from Databook, you must create an Databook Toolbox object that defines how to map the library data in the database into Databook libraries.

Prerequisites

- Your user account has administrator privileges.

Video

- Duplicate the template DXDBDefault toolbox.
- Assign a new call name to the duplicate toolbox.
- Use the Databook Configurator to expose database characteristics and define libraries.
- Save and close the new Toolbox object.



Procedure

1. In the classification tree of the Xpedition EDM Library Cockpit application, navigate to the **Toolbox > Tools > Databook** subcatalog.
2. In the search results area, right-click and then choose **Add New Object**. Alternately, choose a toolbox from the search results list, right-click, and then choose **Duplicate**.



Note:

Instead of creating a new toolbox, you can edit the example DXDBDefault toolbox provided with the data model. If you use the example toolbox, skip Step 3.

3. Complete the **Top** tab to assign a call name, status, and library specification to the toolbox (refer to “[Toolbox Object - Top Tab](#)” on page 266).
4. Complete the **Databook** tab:
 - a. Type the purpose of the configuration in the Description field.
 - b. Click the **Databook Configurator** button to open the Databook configuration editor ([Figure 149](#)).

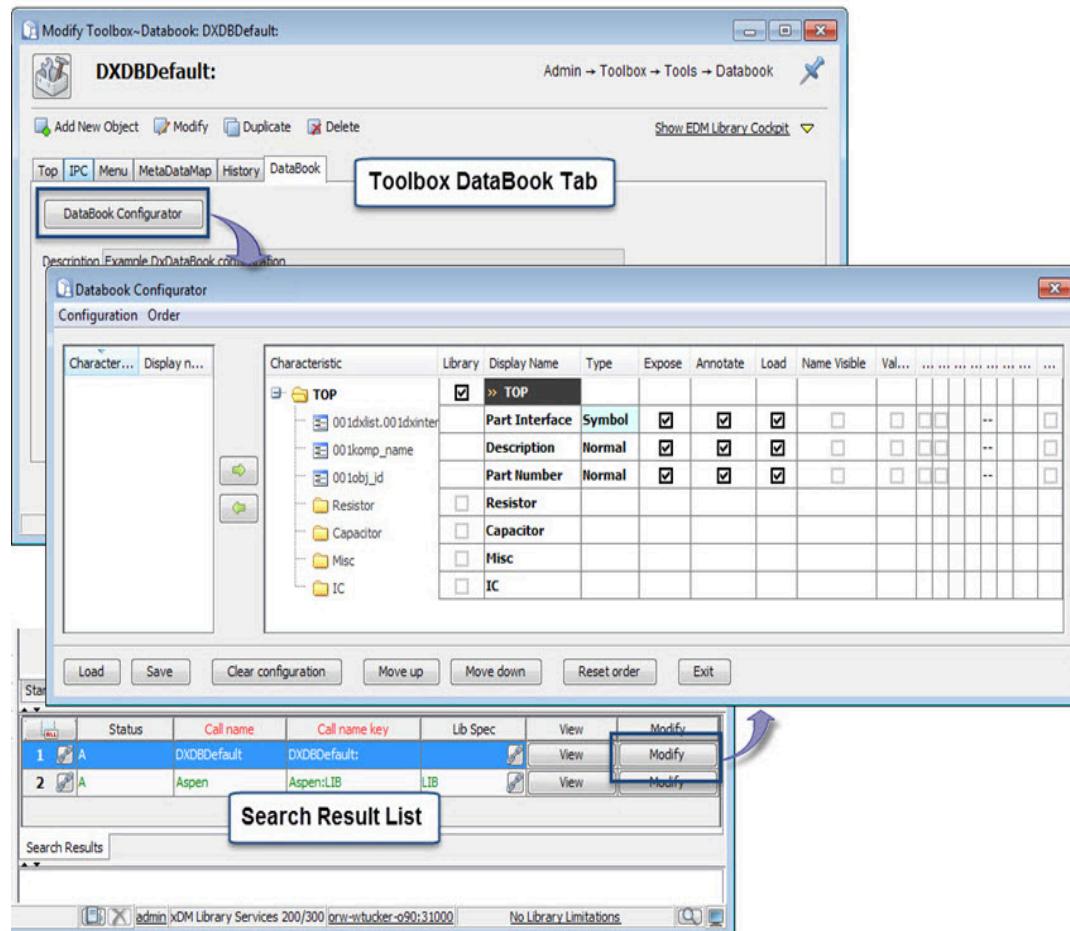


Tip

You can also invoke the Databook Configurator directly from the search results list of returned toolbox names by clicking the **Modify** button to the right of each row.

Depending on how Databook libraries are structured, an administrator or librarian might create several Toolbox objects to serve as different configurations that a design engineer can choose in Databook.

Figure 149. Invoking the Databook Configuration Editor



- c. Click a folder in the right pane to populate the left pane with a list of all the class characteristics and their display names. By default, the characteristics are sorted by their object ID. Optionally, click the Display Name column head to resort the characteristics by display name.
- d. Select a catalog in the **Characteristic** column in the right pane, and then select one or more class characteristics in the left pane. Use the right arrow button to add the characteristic or characteristics to the **Characteristic** list on the right.
- e. Edit the columns in the right pane of the configuration editor:

- For string values, click the string to edit the value or choose a value from a dropdown list. Pressing the return key saves the value, while the Escape key cancels changes.
- For True/False values, click to change the state. Checked indicates true, while unchecked indicates false.

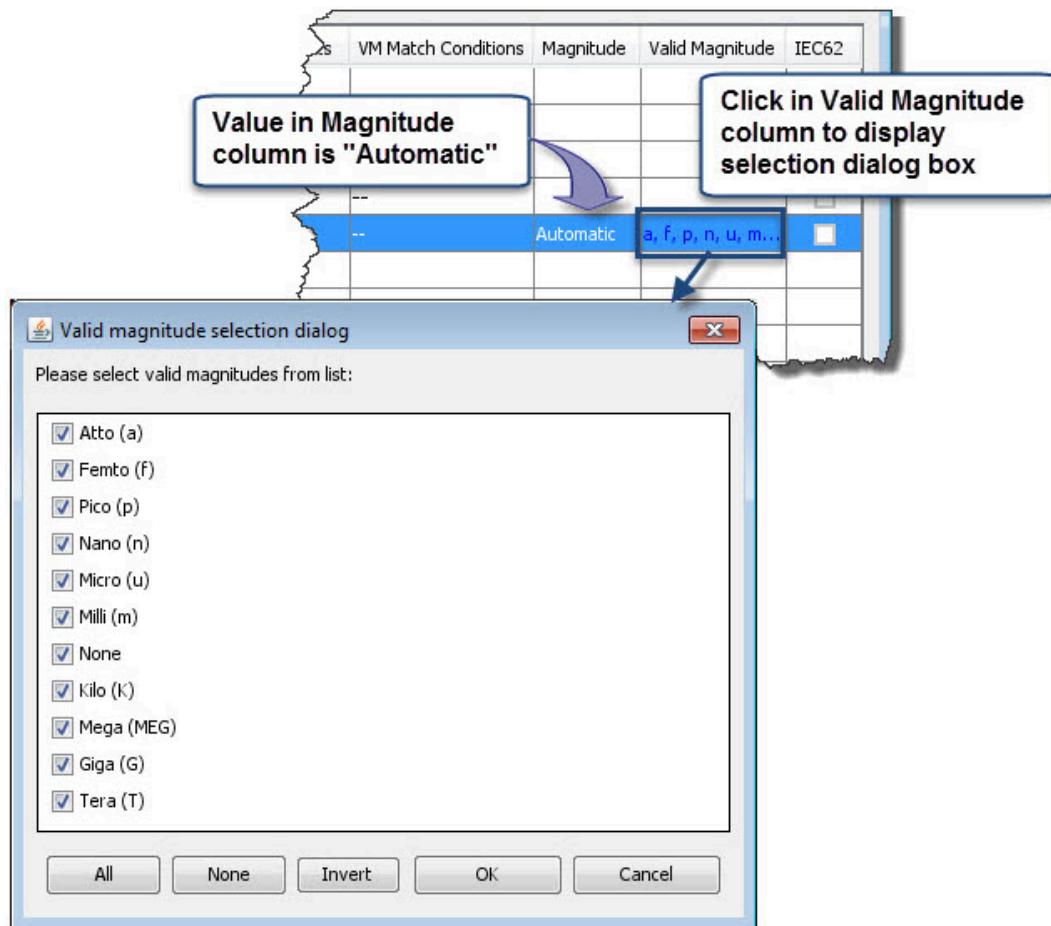
A check next to a catalog in the Library column uses the catalog as a Databook library, even if no characteristics are selected for exposure. A check next to at least one dynamic characteristic exports the entire catalog containing the characteristic to Databook as a library.

- For list entry values, select a value from the dropdown list.
- Use the **Move Up** and **Move Down** buttons to create a characteristic order that is different from the default order.

Most columns have the same name and provide the same function as the **Libraries** tab of the native configuration file editor in the Databook application. For more information, refer to “Editing Component Library Table Properties” in the *Xpedition Designer and Xpedition System Designer Administrator’s Guide*.

The Valid Magnitude column only contains a value if the Magnitude column value is “Automatic” (Figure 150). After setting the Magnitude column value to “Automatic”, click the Valid Magnitude column to display a dialog box with selectable magnitude values.

Figure 150. Displaying the Valid Magnitude Selection Dialog Box



5. Click the **Save** button to save the information in the Databook configuration editor to the **Databook** tab of the Toolbox object.
6. Click the **OK** button to save the Toolbox object and make it available as a Databook configuration.

Results

You can now configure the EDM Library Connector web application to reference the toolbox (refer to “EDM Library Connector” in the *Xpedition EDM Library EDA Library Module Guide*).

After setting up EDM Library Connector, a user can access the configuration through the **Configure > New** or **Configure > Open** menu items in Databook.

Related Topics

[EDM Library Connector \[Xpedition EDM Library EDA Library Module Guide\]](#)

[Default Toolboxes and Hierarchy](#)

[Opening a Toolbox Information Window](#)

Quick Search Toolbox

An optional quick search toolbox configuration restricts any quick search to a subset of characteristics in a given object class. When used, the Quick Search toolbox resides in **Toolbox > Tools > Quick Search** subcatalog and is denoted by the presence of a **Quick Search** tab. The Xpedition EDM Library Cockpit application only permits one active QuickSearch toolbox configuration.



Note:

The **QuickSearch** tab is only available if the toolbox resides in the Toolbox > Tools > Quick Search catalog.

A Quick Search toolbox configuration is not required to use the quick search functionality. If a Quick Search toolbox configuration exists, then the quick search uses the definitions on the **QuickSearch** tab. If a Quick Search toolbox does not exist, or the Quick Search toolbox does not define any characteristics for the object class being searched, then a user must manually choose the characteristics to include in the quick search by checking characteristics in the advanced search criteria pane.

A Quick Search toolbox configuration only requires information on the **Top** tab and the **QuickSearch** tab. During a quick search operation, EDM Library Cockpit ignores any information on other tabs.

[Quick Search Toolbox - QuickSearch Tab](#)

[Creating a Quick Search Toolbox](#)

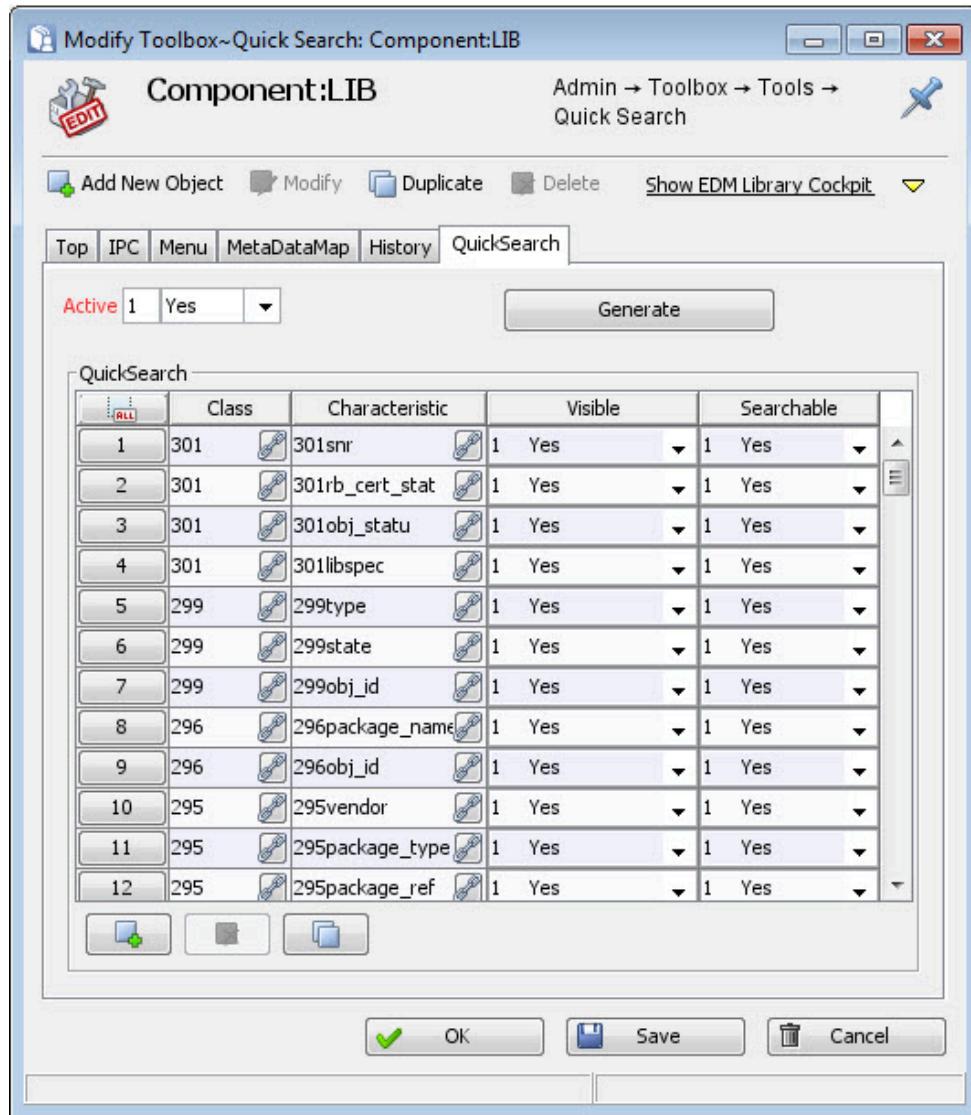
Quick Search Toolbox - QuickSearch Tab

To access: Refer to “[Opening a Toolbox Information Window](#)”

The **QuickSearch** tab contains a collective list of characteristics from each object class that a user can search while in quick search mode. Once active, a user cannot override the toolbox configuration to include more characteristics in the quick search.

Description

Figure 151. QuickSearch Tab of the Toolbox



Characteristics

- **Active** — Indicates if a quick search should use the toolbox configuration. Use No while creating the toolbox and then, when finished, change the value to Yes to make the toolbox active.
- **Generate button** — Populates the QuickSearch list with characteristics from the database that meet the following criteria:
 - A status of “A” (Approved) or “S” (System).
 - The Show and Search Characteristic status bits are enabled.

- A characteristic type of 0 (Standard).
 - A value type of 1, 2, 3, or 4 (Integer, Double, Char, or Long).
- **QuickSearch list** — A list of characteristics to include in a quick search.
- Each row can have one of the following Visible settings:
- **0 (Neutral)** — Checking or unchecking the characteristic in the advanced search criteria pane controls if the characteristic is a column in the search results list.
 - **1 (Yes)** — Always include the characteristic as a column in the search results list.
 - **2 (No)** — Do not display the characteristic as a column in the search results list.

Each row can have one of the following Searchable settings:

- **0 (No)** — Do not use the characteristic during a quick search.
- **1 (Yes)** — Use the characteristic during a quick search.

Deleting a row from the QuickSearch list is the same as if the Visibility and Searchable values for that characteristic were set to No. If there are no characteristic rows for a given object class, then the quick search operation ignores the presence of the toolbox.

Related Topics

[Creating a Quick Search Toolbox](#)

[Default Toolboxes and Hierarchy](#)

[Opening a Toolbox Information Window](#)

[Search String Format \[Xpedition EDM Library Overview\]](#)

Creating a Quick Search Toolbox

When a user does a search while in Quick Search mode, the software uses the information on the **QuickSearch** tab of a Toolbox object to identify which characteristics to include in the search and which characteristics should be columns in the search results list.

Procedure

1. Navigate to the Toolbox > Tools > Quick Search catalog, right-click, and then choose **Add New Object** to display a new toolbox information window.
2. In the **Top** tab, assign a name, status, and library specification to the toolbox.
3. Click the **QuickSearch** tab, and then click the **Generate** button to populate the QuickSearch list.
Optionally, you can manually add rows to the QuickSearch list instead of using **Generate**.

4. Change the values in the Visible and Searchable columns of individual characteristics. By default, the **Generate** button includes all characteristics with a value in the Visible and Searchable columns set to 1 (Yes).

Deleting a row from the QuickSearch list is the same as if the Visibility and Searchable values for that characteristic were set to No. If there are no rows for a given object class, then the quick search operation ignores the presence of the toolbox.

5. Change the value of the Active characteristic at the top of the **QuickSearch** tab to 1 (Yes) to make the toolbox configuration available to users.
-



Note:

The EDM Library Cockpit only permits one active Quick Search toolbox at a time. If you attempt to save a Quick Search toolbox configuration as active when another Quick Search toolbox configuration is already active, a message prompts you to disable the active toolbox first.

6. Save the toolbox configuration.

Related Topics

[Search String Format \[Xpedition EDM Library Overview\]](#)

BOM Extraction Toolbox

The BOM extraction toolbox, named DBOM_DX, enables you to define rules to filter the components in an extracted BOM object. The toolbox resides in the **Toolbox> Tools > BOM Extraction** catalog and is denoted by a **Filtering** tab and two extra characteristics on the **Top** tab.

The software only uses the information in the DBOM_DX toolbox when a user extracts a BOM from Xpedition Designer with one of the following methods:

- Choosing the **Save to EDM Library** menu item in Xpedition Designer
- Clicking **Update BOM** on a variant BOM object in Xpedition EDM Library Cockpit



Note:

The **Filtering** tab and the extra characteristics on the **Top** tab are only available if the toolbox resides in the **Toolbox > Tools > BOM Extraction** catalog.

[BOM Extraction Toolbox - Filtering Tab](#)

[BOM Extraction Toolbox - Top Tab](#)

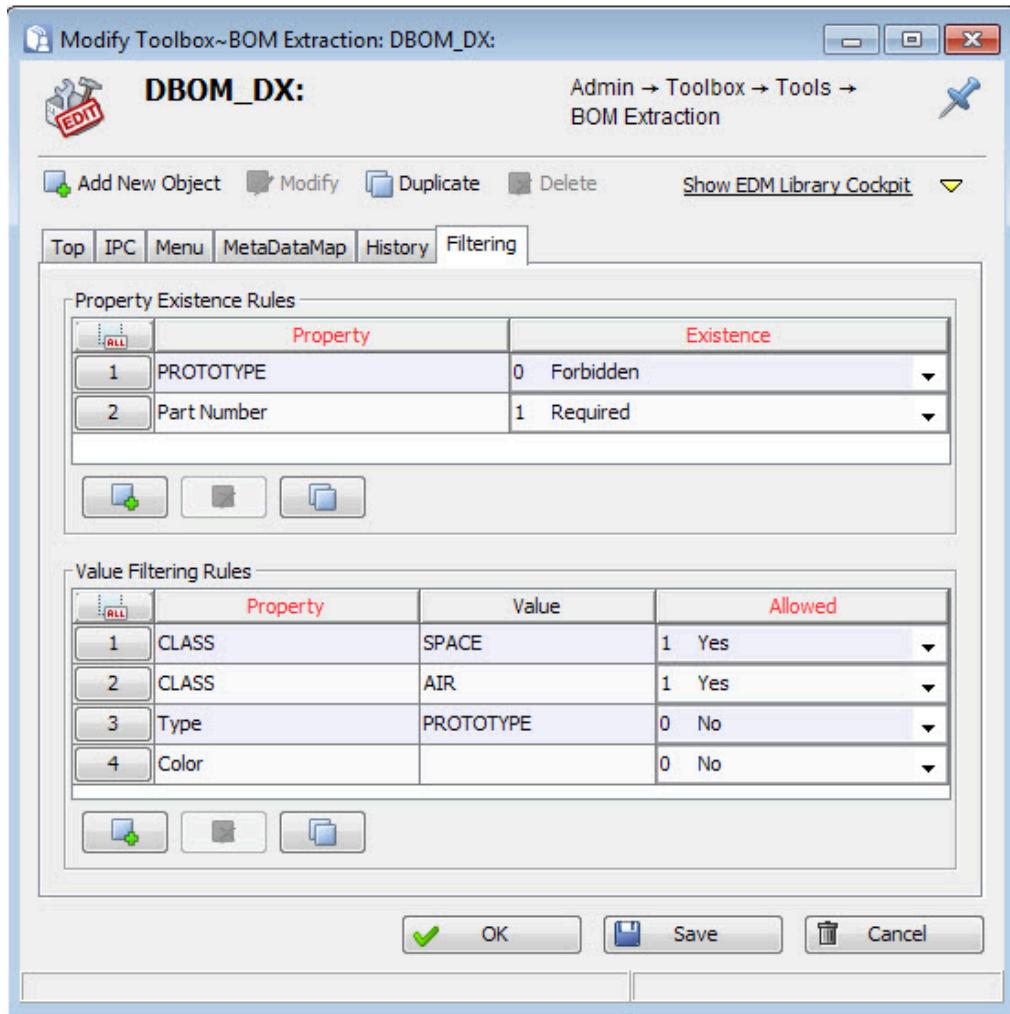
BOM Extraction Toolbox - Filtering Tab

To access: Refer to “[Opening a Toolbox Information Window](#)”

Use the **Filtering** tab on the DBOM_DX Toolbox object to apply a filter to design BOM extractions from Xpedition Designer. Rules on the tab can include or exclude components that may or may not have properties or property values defined by the rules. Property names you specify on the **Filtering** tab are case-sensitive, but property values are case-insensitive. You cannot use wildcards.

Description

Figure 152. Filtering Tab of the DBOM_DX Toolbox



Characteristics

- **Property Existence Rules list** — Each row defines a filter based on whether a property exists or does not exist on a component in the design. Each row in the list has the following columns:
 - **Property** — The property name
 - **Existence** — Defines if the property is required or forbidden

A Required value indicates that the property must be on the component for it to be in the extracted design BOM. A Forbidden value omits any component containing the property from the extracted design BOM.

For example, [Figure 152](#) shows that a component in a design must have a Part Number property to be included in the extracted design BOM, but filters out any components with a PROTOTYPE property.

- **Value Filtering Rules list** — Each row defines a filter based on a property value. Each row in the list has the following columns:
 - **Property** — The property name
 - **Value** — The property value
 - **Allow** — A Yes/No value indicating if the property is permitted

Property value filtering works the same as property name filtering. You can define multiple values for one property as separate filtering rules. For example, [Figure 152](#) shows that if a component has a CLASS property, the property value can only be SPACE or AIR to include the component in the design BOM. The component also cannot have a Type property value of PROTOTYPE, and the Color property cannot have an empty value.

Usage Notes

When a user performs a BOM extraction on an Xpedition Designer design, the software processes the filtering rules in the following order:

1. Checks property required restrictions
2. Checks allowed property values
3. Checks all forbidden properties

Related Topics

[BOM Extraction Toolbox - Top Tab](#)

[Design BOM User Procedures \[Xpedition EDM Library Guide for Designers\]](#)

[Default Toolboxes and Hierarchy](#)

[Opening a Toolbox Information Window](#)

BOM Extraction Toolbox - Top Tab

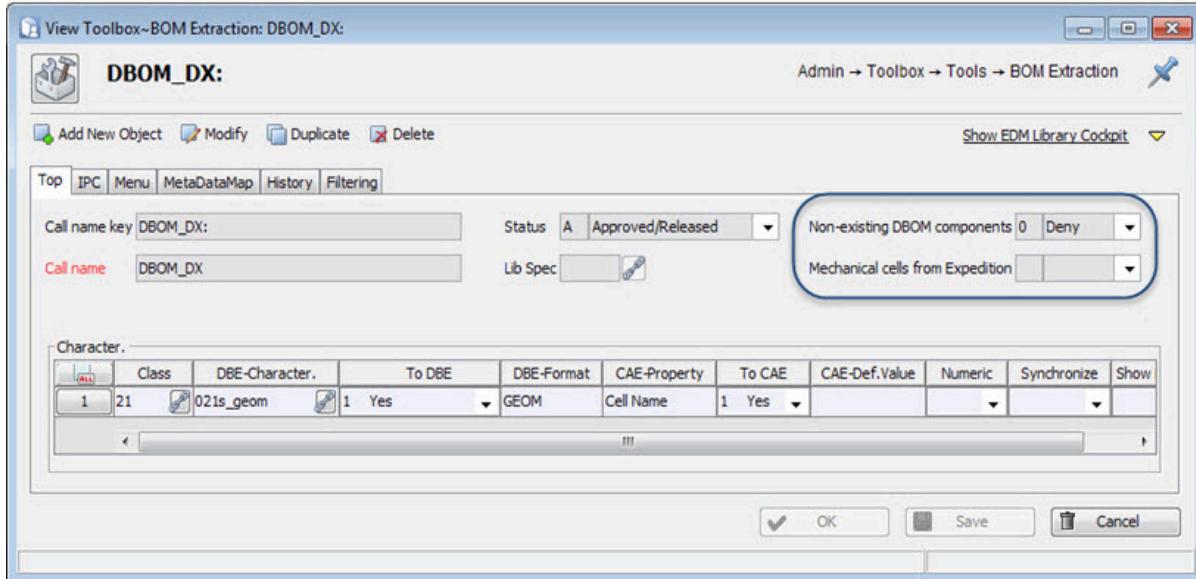
To access: Refer to “[Opening a Toolbox Information Window](#)”

The **Top** tab of a DBOM_DX toolbox has two characteristics on the top right that do not display on the **Top** tab of other toolboxes.

Description

The “Non-existing DBOM components” and “Mechanical cells from Expedition” characteristics (circled in [Figure 153](#)), affect how the BOM extraction software handle certain types of components or cells representing mechanical parts.

Figure 153. Top Tab of the Default DBOM_DX Toolbox



Characteristics



Note:

For information about other characteristics on the **Top** tab of a toolbox not related to BOM extraction, refer to “[Toolbox Object - Top Tab](#)” on page 266.

- **Non-existing DBOM components** — A value of Allow includes components in the extracted variant BOM object that are present in the design but do not exist in the database. A value of Deny does not include components in the variant BOM object that do not also exist in the database.
- **Mechanical cells from Expedition** — A value of Allow includes mechanical cells in the extracted variant BOM object, as long as those cells are placed on the layout side of a circuit board and the design is enabled with variant manager. A value of Deny does not include mechanical cells that do not also exist in the database.

In Xpedition Designer, a mechanical part is a component that has the Cell Name property set. In the database, a mechanical cell is a Cell object with a value for the Mech. Part Number characteristic.

If no value is set for either characteristic, the extraction software uses Deny as the default behavior.

For information about how to complete the Characteristics list box to specify the properties to include in a variant BOM object, refer to Adding Properties to a Variant BOM in the *Xpedition EDM Library Guide for Designers*.

Usage Notes

After extracting a BOM, a user can click the **Check Components** button on the **Variants** tab of a Design (MBOM) object to create a report of any components in the BOM that are not in the database.

Related Topics

[BOM Extraction Toolbox - Filtering Tab](#)

[Extracting Part List Data from a Design \[Xpedition EDM Library Guide for Designers\]](#)

[Design \(MBOM\) Object - Variants Tab \[Xpedition EDM Library Guide for Designers\]](#)

[Default Toolboxes and Hierarchy](#)

[Opening a Toolbox Information Window](#)

Update Component Info Toolbox

To access: Refer to “[Opening a Toolbox Information Window](#)”

The Update Component Info toolbox, named DBOM_Comp, supports the ability to update an existing Design (MBOM) or Variant BOM object with the latest Component object characteristic values. The toolbox resides in the **Toolbox > Tools > Update Component Info** subcatalog and is denoted by an **Update Component Info** tab.



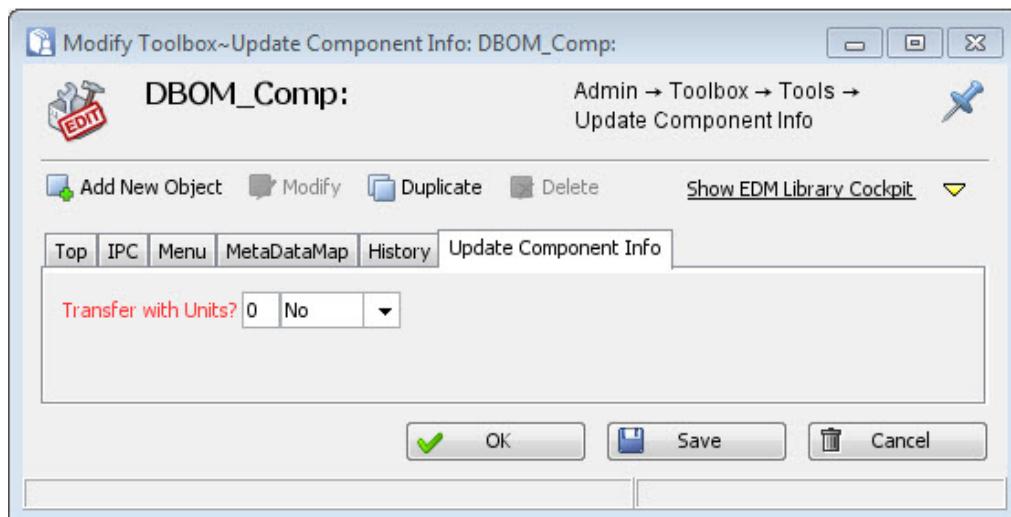
Note:

The **Update Component Info** tab is only available if the toolbox resides in the **Toolbox > Tools > Update Component Info** catalog.

Description

The **Update Component Info** tab is used in conjunction with the **Update Comp Infos** button on the **General** tab of a Design (MBOM) object and establishes the value transfer rules for the EDM Library DBOM Update Component Information functionality.

Figure 154. Update Component Info Tab of the Toolbox



Characteristics

- **Transfer with Units?** — Specifies whether to transfer component information with or without units

Related Topics

[BOM Object Classes \[Xpedition EDM Library Guide for Designers\]](#)

[Update Component Information Functionality \[Xpedition EDM Library Guide for Designers\]](#)

[Default Toolboxes and Hierarchy](#)

[Opening a Toolbox Information Window](#)

Compliance Management Toolboxes

Compliance Management toolboxes describe to the Compliance Manager software the object classes on which to perform a compliance calculation and the Java class to use to process the data for that calculation. Compliance Management toolboxes reside in the **Toolbox > Tools > Compliance** subcatalog and are denoted by the **Compliance** and **Characteristic-Specification** tabs.



Note:

The Compliance and Characteristic-Specification tabs are only available if the toolbox resides in the **Toolbox > Tools > Compliance** catalog.

When a database is configured with the Compliance Manager module, a **ComplianceManagerConfig:** toolbox is located in the Admin > Toolbox > Tools > Compliance catalog group that contains the following two tabs:

- **Compliance** — Describes object classes on which to perform a compliance calculation and the Java class to use to process the data for that calculation.
- **Characteristic-Specification** — When using multivalue characteristic-based processing the tab lists the characteristics, and the values of those characteristics, that denote compliance with a given specification.

By default, all tabs of the toolbox are empty except for the **Compliance** tab. For information about how to configure the **ComplianceManagerConfig:** toolbox for use with the Compliance Manager module, refer to “Administrative Tasks” in the *Xpedition EDM Library Compliance Management Module Guide*.

Related Topics

[Administrative Tasks \[Xpedition EDM Library Compliance Management Module Guide\]](#)

[Default Toolboxes and Hierarchy](#)

[Opening a Toolbox Information Window](#)

Component Synchronization Toolbox

The component synchronization toolbox, named CompSync, contains the parameters to use when propagating values from characteristics located on a “parent” Component object to the same characteristics on one or more “child” Component objects that are related to the parent. The Component Synchronization toolbox resides in the **Toolbox > Tools > Component Synchronization** subcatalog and are denoted by a **Component Synchronization** tab.



Note:

The **Component Synchronization** tab is only available if the toolbox resides in the **Toolbox > Tools > Component Synchronization** catalog.

Component Synchronization Toolbox - Component Synchronization Tab

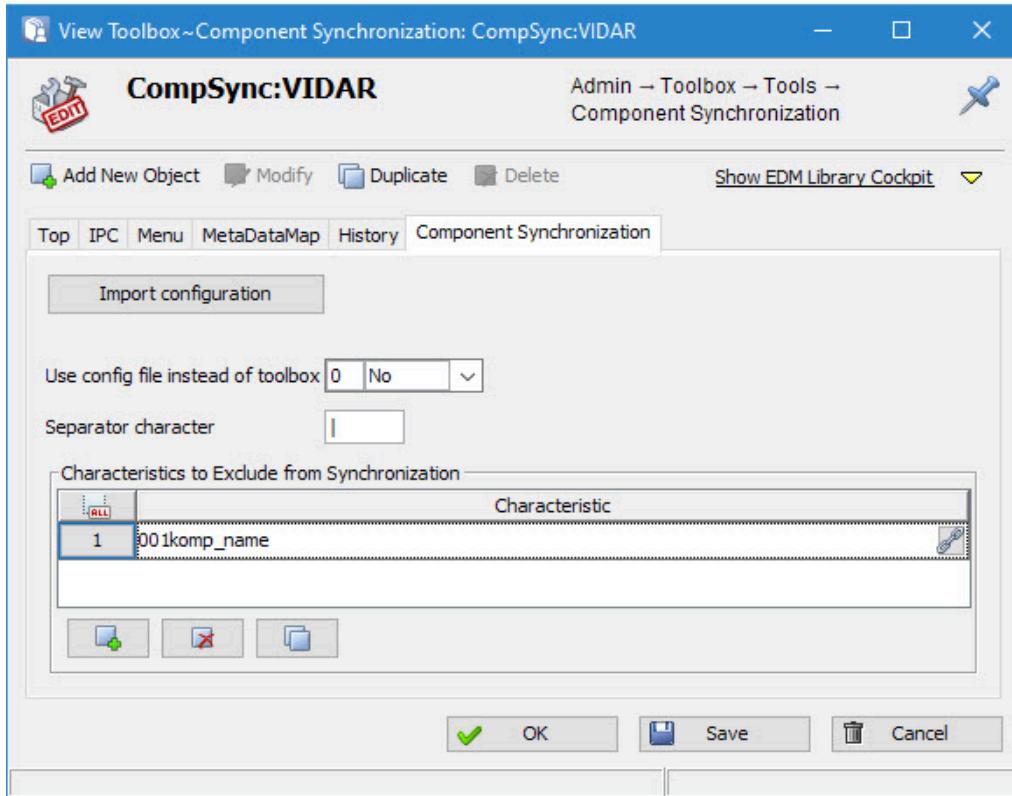
Component Synchronization Toolbox - Component Synchronization Tab

To access: Refer to “[Opening a Toolbox Information Window](#)”

When propagating data from parent to child components with the **Synchronize Components** menu item, the **Component Synchronization** tab specifies the separator characteristic found in the part number of a child component, and a list of characteristics to exclude from synchronization.

Description

Figure 155. Component Synchronization Tab of a Component Synchronization Toolbox



Characteristics

- **Import configuration button** — Import the settings in a *compsync.properties* file into the toolbox.
- **Use config file instead of toolbox** — A Yes value ignores information in the CompSync toolbox during a component synchronization and reads component synchronization settings from a *compsync.properties* file instead. A No value uses settings from the toolbox.
- **Separator character** — The character in the Part Number value of a child component that separates the child value from the parent Part Number value. For example, a parent component might have a Part Number value of PN-12345, while the child components have Part Number values of PN-12345|DIP and PN12345|SOIC. The presence of the pipe (|) separator character in the Part Number enables the component synchronization process to recognize the component as a child with a parent.

The default separator is a pipe (|) character.
- **Characteristics to Exclude from Synchronization** — A list of characteristics to exclude from propagation to a child component from the parent. By default, the list excludes the 001komp_name (Description) characteristic so as not to overwrite the description on the child component with the description from the parent.

Related Topics

[Synchronizing Parent and Child Components \[Xpedition EDM Library Guide for Component Engineers\]](#)

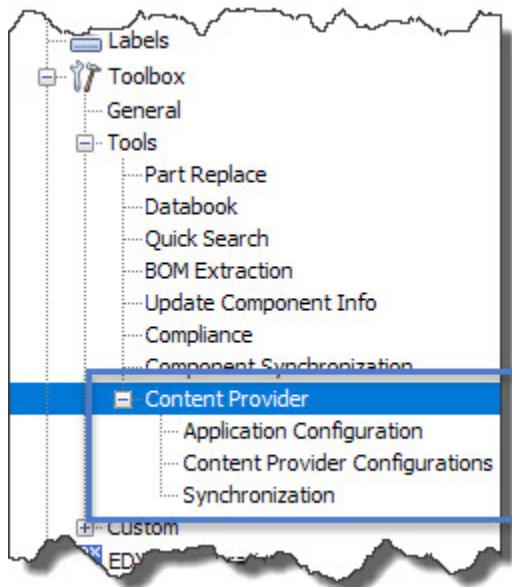
[Default Toolboxes and Hierarchy](#)

[Opening a Toolbox Information Window](#)

Content Provider Toolboxes

The Content provider toolboxes reside in the **Toolbox > Tools > Content Provider** subcatalog and are denoted by a **Content Provider** tab. Toolboxes in the Content Provider Configurations subcatalog contain the parameters to use when defining how the optional EDM Supplychain product interfaces with an external content provider (right now, the only available content provider configurations toolbox is named Supplyframe). A toolbox in the Application Configuration subcatalog defines the proxy host setting on the Content Provider Preferences dialog box. The Supplychain software does not currently recognize toolboxes in the Synchronization subcatalog.

Figure 156. Content Provider Toolbox Hierarchy



[Application Configuration Toolbox - Content Provider Tab](#)
[Content Provider Configurations Toolbox - MetaDataMap Tab](#)
[Content Provider Configurations Toolbox - Content Provider Tab](#)

Application Configuration Toolbox - Content Provider Tab

To access: Refer to “[Opening a Toolbox Information Window](#)”

The **Content Provider** tab of a Application Configuration toolbox contains sections to enable usage of a proxy, to enable usage of an SSL certificate, and to specify a Java class for an application implementation when using EDM Supplychain functionality. If enabled through the Application Configuration toolbox, you can use the Content Provider Preferences dialog box to disable proxy or certificate usage when not needed.

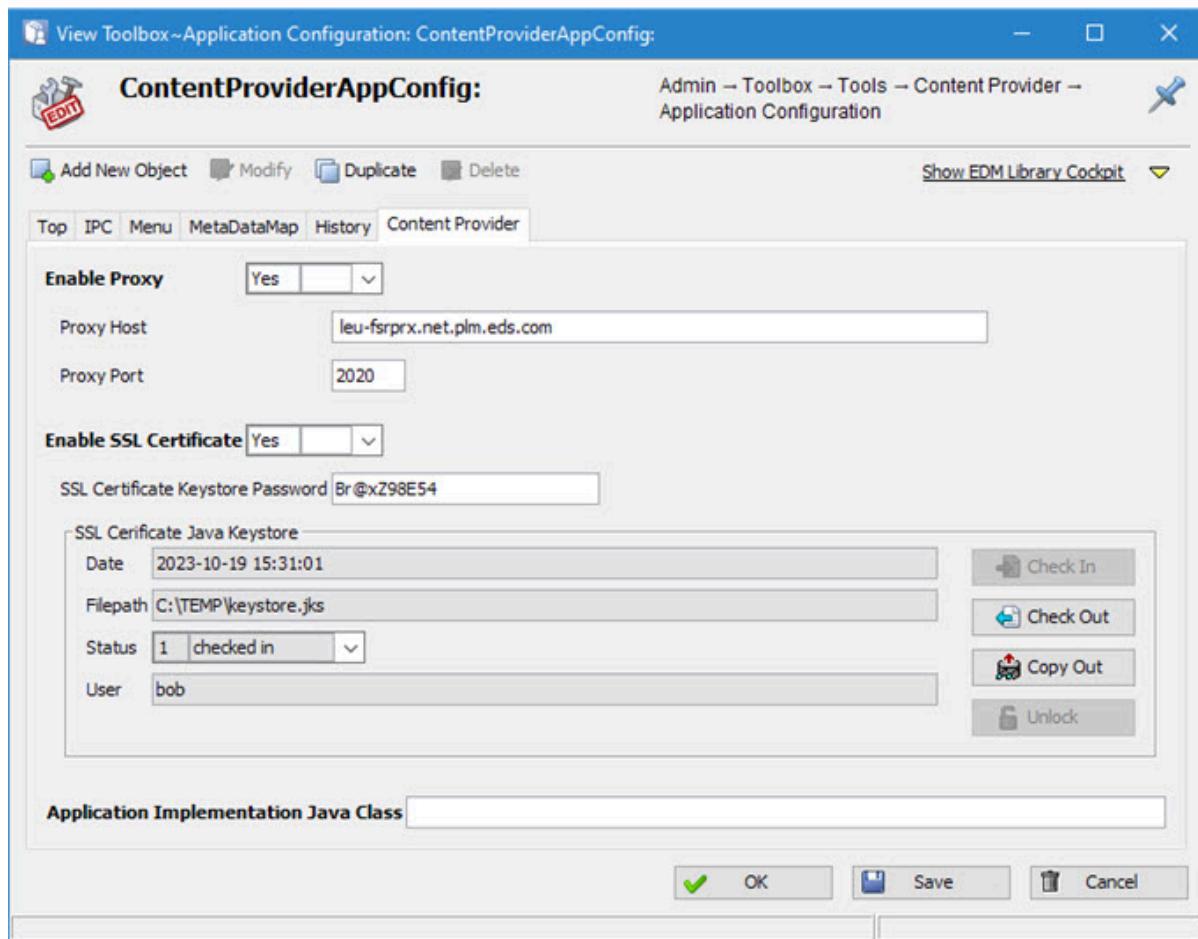
Description



Note:

The **Content Provider** tab is only available if the toolbox resides in the Toolbox > Tools > Content Provider catalog.

Figure 157. Content Provider Tab of an Application Configuration Toolbox



Characteristics

- **Enable Proxy** - Setting the value of Enable Proxy to Yes makes the HTTP Proxy field visible on the Content Provider Preferences dialog box and sets the default to "Use toolbox settings". Type a name or IP address into the Proxy Host field and a port number into the Proxy Port field. EDM Library Cockpit users can now use the Content Provider Preferences dialog box to use the proxy when required (for example, when behind the firewall of your company) and disable the proxy when not required.
- **Enable SSL Certificate** - To use an SSL certificate, which is sometimes required by a proxy server, you must have generated a java keystore in which you have stored any downloaded certificates. Setting the value of Enable SSL Certificate to Yes activates the SSL Keystore field on

the Content Provider Preferences dialog box and sets the default value to "Use toolbox settings". Type the password of the SSL certified keystore into the password field, and use the **Check In** button to import the keystore file into the database. EDM Library Cockpit users can now use the Content Provider Preferences dialog box to use the SSL certificate when required, or disable certificate use when not required.

Application Implementation Java Class - The software currently does not recognize a value in this field, and reserves this characteristic for future use.

Related Topics

[Setting Content Provider Preferences \[Xpedition EDM Library Guide for Component Engineers\]](#)

Content Provider Configurations Toolbox - MetaDataMap Tab

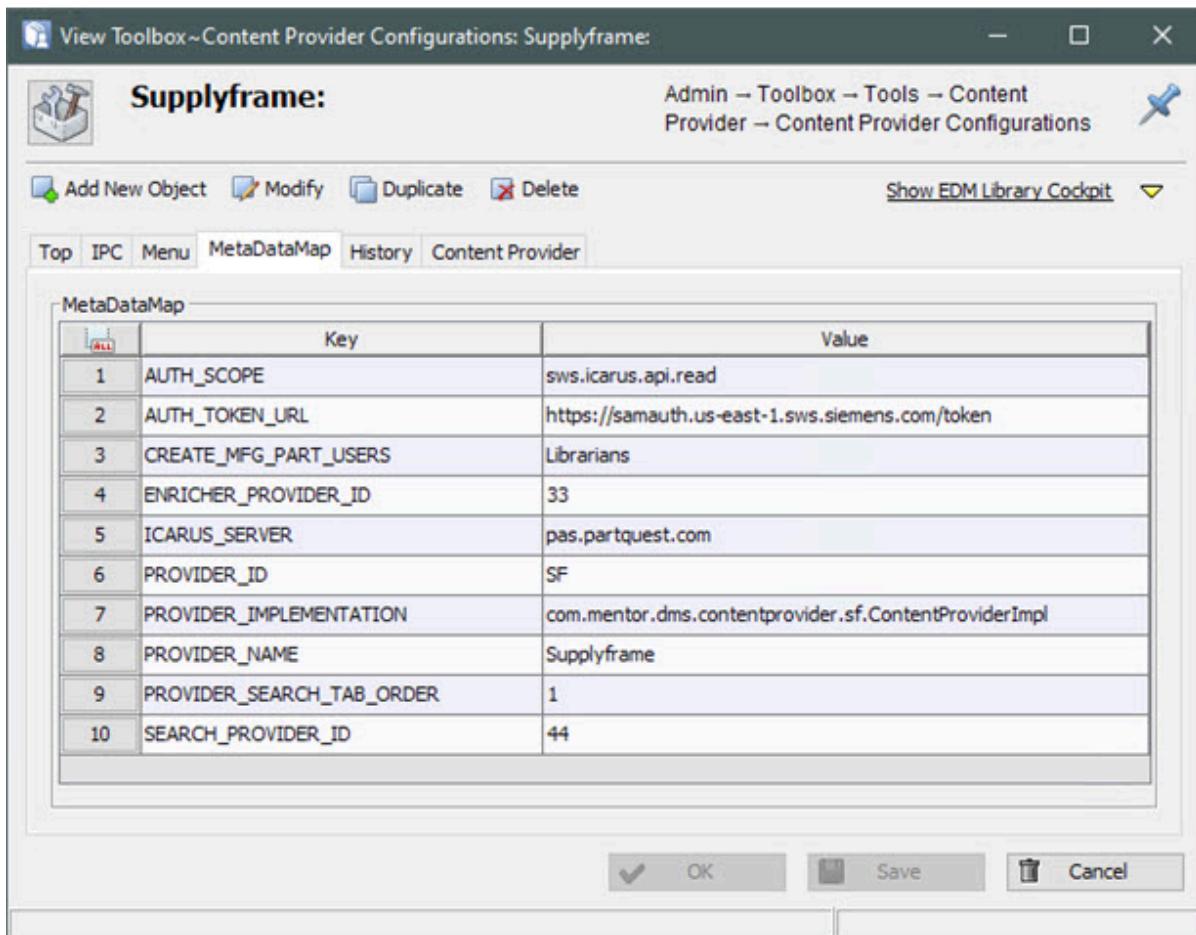
To access: Refer to "[Opening a Toolbox Information Window](#)"

Entries on the **MetaDataMap** tab of a Content Provider Configurations Toolbox object provide configuration parameters required by both the application framework and the Content Provider implementation. Most values are set by the software at installation and, with the exception of PROVIDER_NAME, should not be changed.

Description

[Figure 158](#) shows the keys and default values on the **MetaDataMap** tab of a Content Provider Toolbox object.

Figure 158. MetaDataMap Tab of a Content Provider Toolbox



Fields

The following table lists the entries in the Key column required by the Content Provider framework to uniquely describe each Content Provider.

| Key | Value |
|---------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PROVIDER_ID | Uniquely identifies data associated with this content provider implementation. For example, SF. |
| PROVIDER_NAME | The name of the Content Provider to display in the Extended Search window. For example, Supplyframe. |
| PROVIDER_SEARCH_TAB_ORDER | If more than one content provider is configured, then this parameter indicates the ordering of the tab within the Extended Search window. Because Supplyframe is the only supported external content provider, this value is 1. |
| PROVIDER_IMPLEMENTATION | The name of the Java class that implements the AbstractContentProviderinterface for this content provider. |

Related Topics

[Toolbox Object - MetaDataMap Tab](#)

Content Provider Configurations Toolbox - Content Provider Tab

To access: Refer to “[Opening a Toolbox Information Window](#)”

The **Content Provider** tab of a Content Provider Configurations toolbox contains sections for the Mapping Configuration File, the Roles, and the Property Name List of the Content Provider implementation.

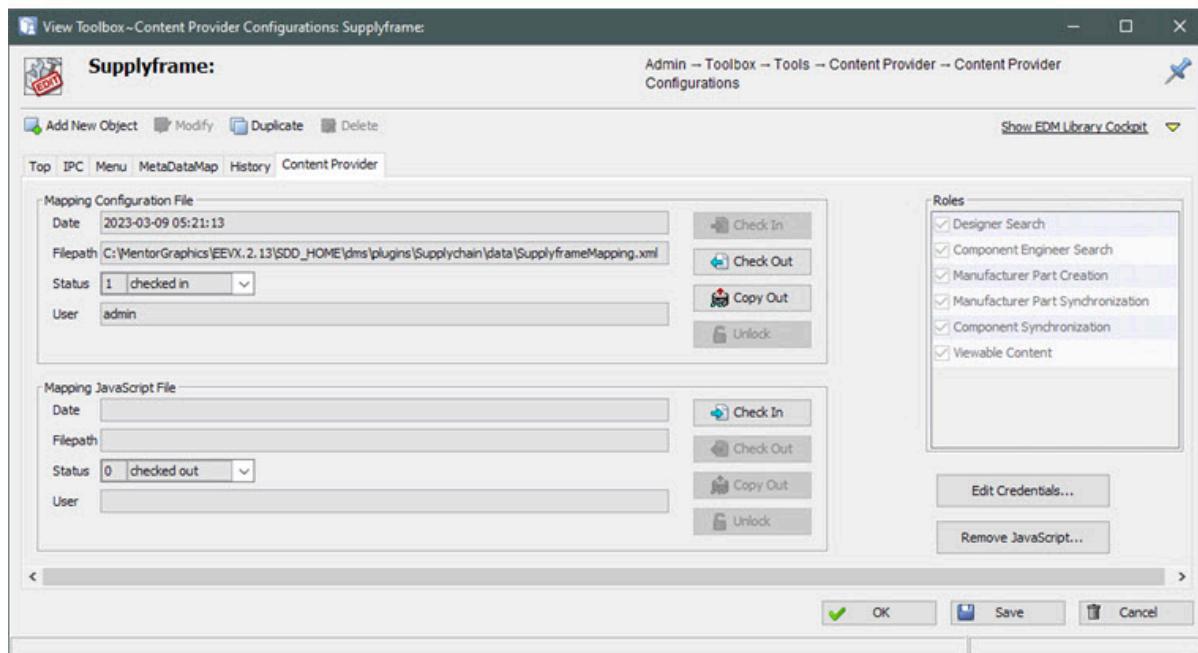
Description



Note:

The **Content Provider** tab is only available if the toolbox resides in the Toolbox > Tools > Content Provider catalog.

Figure 159. Content Provider Tab of a Content Provider Toolbox



Characteristics

- **Mapping Configuration File list box** - Used to check in a mapping configuration file that describes the required behavior for mapping the Content Provider part classes and parameters to EDM Library Manufacturer Part and Component catalogs and characteristics. For more information, refer to "Importing the Mapping Configuration File" in the *Xpedition EDM Library Guide for Component Engineers*.

- **Mapping Javascript File list box** - Used to check in a plain text mapping JavaScript file with supporting JavaScript code for implementing mapping functions for the ScriptedComponentPropertyMap mappings in the Content Provider Mapping Configuration File.
- **Roles** - The Content Provider Roles indicate to the framework what roles or functions the content provider plays within the application. For example, whereas IHS or SiliconExpert may participate in all of these roles, Avnet may only participate in Designer/Component Search and Viewable Content.
 - **Designer Search** – Display this Content Provider’s tab in the Designer search context.
 - **Component Engineer Search** – Display this Content Provider’s tab in the Component Engineer search context.
 - **Manufacturer Part Creation** – Permit creation of Manufacturer Parts/External Content from this Content Provider.
 - **Manufacturer Part Synchronization** - Permit synchronization of Manufacturer Parts/External Content to updates from the Content Provider.
 - **Component Synchronization** – Permit synchronization of Components from Manufacturer Parts based on mappings from the Content Provider.
 - **Viewable Content** - Display a “View/Compare” or “View External Content” window for parts from this Content Provider.
- **Edit Credentials button** - Use to securely enter account credentials required by a particular Content Provider. An administrator should only provide Edit Rights for this action button to users or groups who are designated to maintain Content Provider account credentials. For more information, refer to "Restricting Edit Rights to Content Provider Credentials" in the *Xpedition EDM Library Guide for Component Engineers*.

Assignment Rows

A designer uses the EDM Library assignment function to assign part numbers (components) to generic instances in Xpedition Designer. The rows in the **Top** tab of a toolbox that apply to assignment define the mapping between EDM Library characteristic names and property names inside the CAE tool and have special configuration rules.

Assignment

The assignment function must have at least one starting point and one target point for the search path. Normally, the starting point is the name of the Symbol object. The target point is usually the ID of a Component object.

The DBE-Character, and CAE-Property columns on the [Toolbox Object - Top Tab](#) establish the mapping between the EDM Library characteristic name and the CAE properties. The Class column contains the class number of the EDM Library characteristic to transfer.

The To DBE and To CAE columns define the transfer direction, as follows:

- **To CAE** — Transfers the value in the EDM database to the CAE property in the schematic.
- **To DBE** — Transfers the value of the CAE property in the schematic to the EDM database and uses that value as a search restriction inside the search window of the assignment function.

The DBE Format column defines the type of characteristic by means of specific keywords:

- **PART** — Marks the target point of the search, normally the Component ID.
-



Note:

Because the assignment operation uses the **Search Ref.** tab of an object to find the target point, beginning from the starting point, the **Search Ref.** tab of the object must be correctly configured.

- **STATUS** — Contains information about the status of the object (refer to [EDM Library Characteristic Value Restrictions](#)).
- **REF** — Contains the CAD reference (for example, R23). The reference is only displayed for information purposes and does not have an influence on searching or assignment.
- **SYMBOL** — Marks the starting point for the search and contains information about the Interface or Symbol. The corresponding property must already contain a data value.

[Table 17](#) shows an example for the basic assignment rows in the **Top** tab of the `mentordx_ipc_2007` toolbox:

Table 17. Example of Basic Assignment Entries

| Class | DBE Character. | To DBE | DBE Format | CAE Property | To CAE |
|-------|----------------|--------|------------|--------------|--------|
| 1 | 001obj_id | 0 | PART | PART_NO | 1 |
| 1 | 001obj_statu | 0 | STATUS | | 0 |
| 1 | | 1 | REF | REF | 0 |
| 11 | 012soft_path | 1 | SYMBOL | PATHNAME | 0 |

Property Value Restrictions

The search results in the search window of the assignment function can also be restricted. Therefore, EDM Library characteristic names must be allocated to the corresponding CAE property names. It is possible to map one CAE property to several (catalog) characteristics, as shown in [Table 18](#). The CAE property **VALUE** can be the resistance value or the capacitance value, stored in two different catalog characteristics in the EDM database, for example.

Therefore, data values have to be entered into the following columns of the toolbox:

- **Class**— The class number referencing the catalog characteristic, normally the same class number as in the PART line.
- **To DBE** — This value can be set to '1' or '0':
 - '1' uses the value of the corresponding property as a search restriction in the EDM Library assigning operation search and to create the instance groups.
 - '0' does not use the value of the property for the assigning operation search and the value does not show in the instance group names.
- **To CAE** — Setting this flag to '1' transfers the EDM database value to the CAE property value.

Table 18. Example Toolbox To Restrict Property Values

| Class | DBE Character. | To DBE | DBE Format | CAE Property | To CAE |
|-------|----------------|--------|------------|--------------|--------|
| 1 | 000resistanc | 1 | | VALUE | 1 |
| 1 | 000capacitan | 1 | | VALUE | 1 |
| | 000toler | 1 | | TOL | |

EDM Library Characteristic Value Restrictions

In addition, the assigning operation search result can also be restricted by characteristics only available in the EDM database. It is possible, for example, to define that the components to allocate to generics must have a certain Status. Different values can be specified, as follows:

- Values permitted for component assignment.
- Values for which the system asks whether or not to do the component assignment.
- Values not permitted for assignment and where attempting an assign results in an error message.

Such a status characteristic is defined by the STATUS keyword in the DBE Format column. Therefore, data values have to be entered into the following columns of the toolbox:

- **Class** — The class number of the object class containing the characteristic.
- **DBE Character** — The name of the characteristic in the EDM database.
- **CAE-Def. Value**— This column has to contain the value groups for 'allowed', 'inquire' or 'not allowed' in exactly this order. Components with a status defined in the inquire group generate a question dialog box during instantiation, part assignment, and part replacement.

Separate values in the CAE-Def. Value column by a comma, and separate the value groups by the pipe character ('|'). A group can also have asterisk ('*'), which means that all values different from the ones specified in the other value groups are used.

[Table 19](#) shows an example that allows assignment of all components with a Status value of "A", asks before allowing assignment of components with a Status value of "D" or "U", and does not permit

assignment of any other components with any other values (for example, components with Status values of "X" or "R").

Table 19. Example Toolbox for Restrictions on EDM Library Values

| Class | DBE Character. | DBE Format | CAE-Def. Value |
|-------|----------------|------------|----------------|
| 1 | 001obj_statu | STATUS | A D,U * |

Several Parallel Search Paths

It is also possible to define several parallel search paths in the toolbox. This makes sense if there are several possibilities to find a part number from the Interface or Symbol.

Therefore, any number of alternate starting points can be defined. In this case, the DBE Format column contains a SYMBOL entry and a SYMBOL_ALT_X entry, where 'X' denotes the number of the alternate starting point, beginning with '1' and incremented by '1' for additional alternates.

An **Assigning operation...** menu entry displays in the menu group of each object class that has a defined starting point.

[Table 20](#) shows an example for defining two starting points for the assigning operation search, one for class number 10 and one for class number 1:

Table 20. Example Toolbox for Several Alternate Starting Points

| Class | DBE Character. | To DBE | DBE Format | CAE Prop. | To CAE |
|-------|----------------|--------|--------------|-----------|--------|
| 10 | 010topelm | 1 | SYMBOL | COMP | 0 |
| 1 | 001element | 1 | SYMBOL_ALT_1 | NAME | 0 |

Related Topics

[Assigning Parts \[Xpedition EDM Library Guide for Designers\]](#)

[Toolbox Object - Top Tab](#)

[Default Toolboxes and Hierarchy](#)

[Opening a Toolbox Information Window](#)

History Tracking

The History Tracking object class provides a system-generated record of changes made to characteristic values by a user or process. The system generates one History Tracking object for each value change that captures information, such as the name of the changed characteristic, the object where the value changed, the modification date, who made the modification, the old and new characteristic value, and the type of action (delete, insert, or modify).

To track the change history of a specific characteristic, the **Status** tab of the characteristic definition must have the History Tracking bit set, and you must initialize the object class to register the characteristic.

The data model enables history tracking of a few characteristics by default, but not for most characteristics. In addition to objects in the History Tracking class, the following object classes have a **Hist Tracking** tab that shows the change history of tracking-enabled characteristics on a specific object:

- Components (class 1)
- Mapping (class 10)
- Interface (class 70)
- Padstack (class 120)
- Pad (class 122)
- Hole (class 123)
- Cell (class 130)

Because the system creates and maintains History Tracking objects, you cannot edit their content.



Tip

Because history tracking records every change to a characteristic value, storing the history of many different characteristics on many different types of objects can require large amounts of storage space. Therefore, you should enable history tracking only for those characteristics critical to corporate processes.

For maintenance purposes, you might want to periodically delete objects from the History Tracking class to reduce the amount of stored history information. Deleting a History Tracking object also removes the reference data from the list on any **Hist Tracking** tab.

The following video visually shows how a typical user might use the History Tracking capability of EDM Library.



[Enabling History Tracking](#)

[Creating a History Tracking Tab](#)

[Finding Characteristics Recorded by History Tracking](#)

[Opening a History Tracking Information Window](#)

[History Tracking Object - Top Tab](#)

Enabling History Tracking

An administrator can enable history tracking for a characteristic, and then have the system generate a new History Tracking object when the characteristic value on an object changes.

Prerequisites

- You invoked the Xpedition EDM Library Cockpit application and connected to a database.
- You are using an account with administrative privileges that permits editing the characteristic class and object classes class.

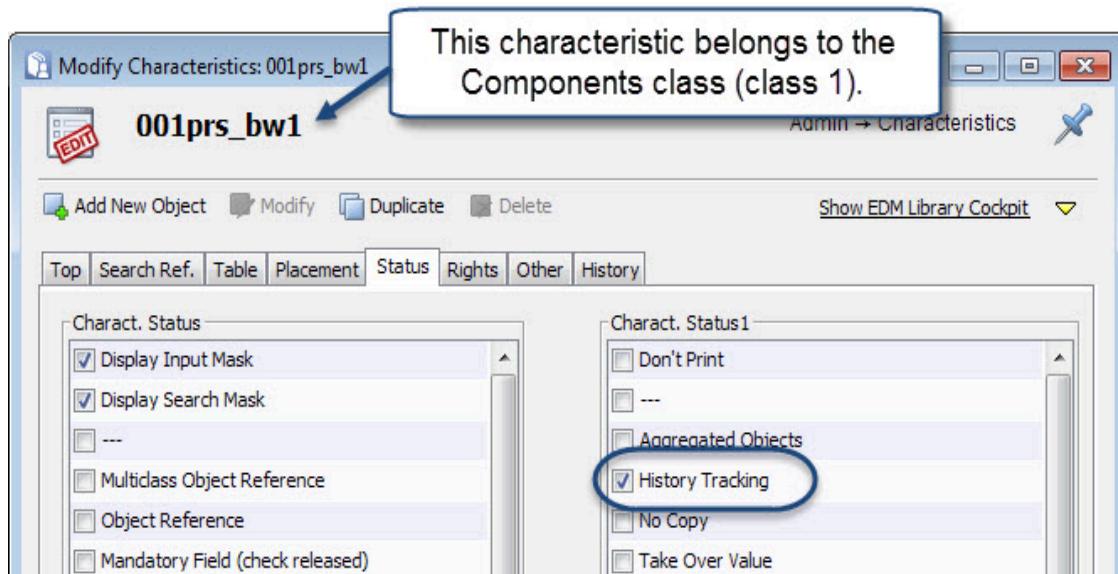
Procedure

1. Locate the characteristic whose history you want to track, and then open the information window for the characteristic in Modify mode.

Note the class to which the characteristic belongs (you will need this information in Step 4). For class characteristics, you can usually determine the class number by the first three digits of the characteristic name ([Figure 160](#)). For dynamic characteristics, use the parent class number.

2. Click the **Status** tab and check the History Tracking status bit ([Figure 160](#)).

Figure 160. Setting the History Tracking Status Bit



3. Click **OK** to close the Characteristic information window.
4. Open the Class object to which the characteristic belongs in Modify mode.



CAUTION:

Do not initialize history tracking on the Dynamic Characteristics class (class 0) because it can cause database integrity problems. For dynamic characteristics, initialize history tracking on the parent class. The system automatically tracks both class characteristics and dynamic characteristics associated with the class.

5. Click the **History** tab and click the **Initialize History Tracking** button.
6. Click **OK** to close the Class information window.

Results

Whenever a user or process edits the characteristic value, the software creates a new History Tracking object. If the characteristic belongs to an object class with a **Hist Tracking** tab on the information window, the change also displays on that tab after editing and saving an object.

If the object class does not have a **Hist Tracking** tab, the system still creates a History Tracking object. To create a **Hist Tracking** tab when an object class does not have one, refer to “[Creating a History Tracking Tab](#)” on page 326.

Related Topics

- [Opening a Characteristic Information Window](#)
- [Opening an Object Class Information Window](#)
- [Characteristic Object - Status Tab](#)

[Object Classes Object - History Tab](#)

[Opening a History Tracking Information Window](#)

Creating a History Tracking Tab

If you enable history tracking for characteristics in a class, and the class does not have a **Hist Tracking** tab, you can create one. The tab is only necessary if you want to display a change list in the information window of the changed object in addition to generating objects in the History Tracking class.

Prerequisites

- You invoked the Xpedition EDM Library Cockpit application and connected to a database.
- You are using an account with administrative privileges.

Procedure

1. Open an object information window in the class where you want to initialize history tracking and ensure there is a system-maintained “Last Modified By” characteristic (usually visible on the **General**, **Top**, or **History** tab). Within the data model, it often has a name of <class_no>bearbeit.

Note that a class might define a <class_no>bearbeit characteristic but not make it visible in the object information window. In this case, do a search to verify that <class_no>bearbeit exists. If it does not, copy and rename 001bearbeit (for example, to 922bearbeit) while making the following changes to the characteristic definition:

- On the **Top** tab, change the characteristic name, class number, and reference class to the new class number.
 - On the **Table** tab, specify the correct class table name.
 - (Optional) On the **Placement** tab, change parameters such as whether the characteristic is visible, the tab displaying the characteristic, and the characteristic position.
2. Open the <class_no>obj_id characteristic definition and, on the **Status** tab, ensure the Class No. bit is set.
 3. Copy and rename the History Tracking list frame and column characteristics from the Components class or from one of the library object classes ([Table 21](#)):
 - On the **Top** tab of each copied characteristic, change the three digit prefix of the name to reflect the destination class, and change the Ref. Class and Class values to the new class number.
 - On the **Status** tab of the new list frame characteristic, make sure that the Class No. bit is set.

Table 21. History Tracking List Frame Characteristics

| | Components Object Class | Library Object Classes |
|----------------------------------|-------------------------|------------------------|
| List Frame Characteristic | 001hist_lst | <class_no>ht_hist_lst |
| Column Characteristics | 001action | <class_no>ht_action |

| | Components Object Class | Library Object Classes |
|--|----------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | 001act_time
001characid
001line_no
001user
001newval
001oldval
001refer_to | <class_no>ht_act_time
<class_no>ht_characid
<class_no>ht_line_no
<class_no>ht_user
<class_no>ht_newval
<class_no>ht_oldval
<class_no>ht_refer_to |

4. Initialize history tracking on at least one characteristic in the class using the procedure in “[Enabling History Tracking](#)” on page 324.

Results

Objects in the class now have a **Hist Tracking** tab similar to the one on Component objects or library objects (depending on the characteristic set you copied).

Related Topics

- [Component Object - History Tracking Tab \[Xpedition EDM Library Guide for Component Engineers\]](#)
- [Characteristic Object - Top Tab](#)
- [Characteristic Object - Table Tab](#)
- [Characteristic Object - Placement Tab](#)
- [Characteristic Object - Status Tab](#)

Finding Characteristics Recorded by History Tracking

You can return a list of characteristics for which history tracking is enabled.

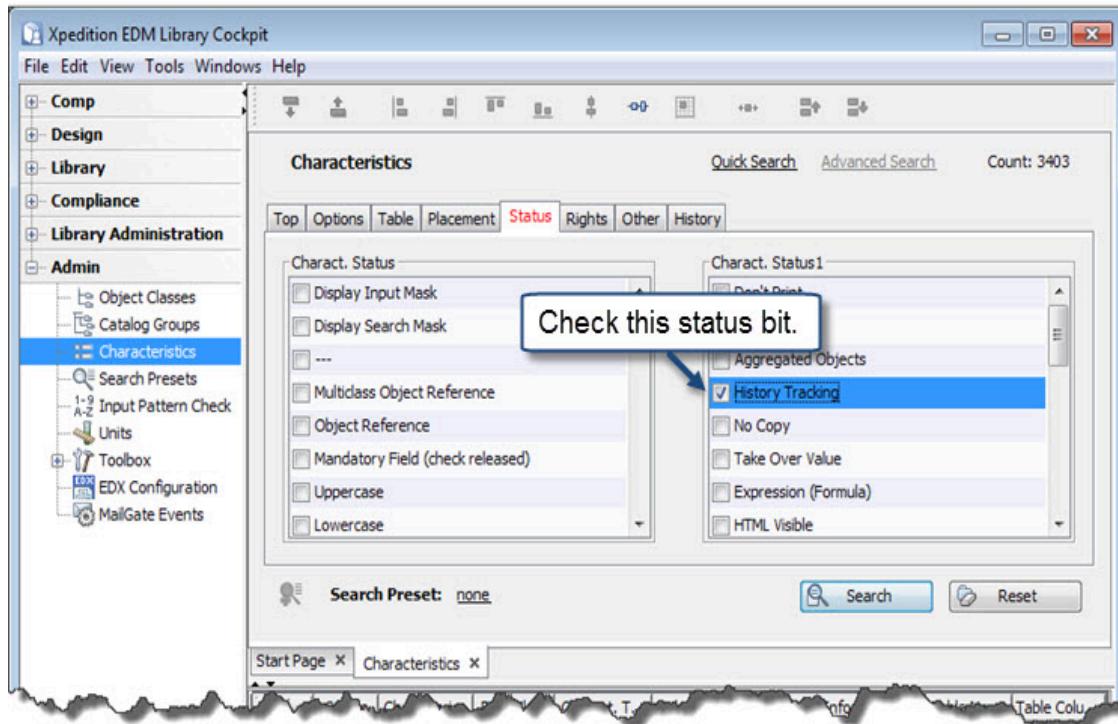
Prerequisites

- You invoked the Xpedition EDM Library Cockpit application and connected to a database.
- You are using an account with administrative privileges.

Procedure

1. In the object classification pane, open the **Admin** module, then select **Characteristics**.
2. Make sure the search criteria pane is in Advanced Search mode.
3. Click the **Status** tab and check History Tracking ([Figure 161](#)).

Figure 161. Searching for Tracked Characteristics



4. Click **Search**.

Results

Xpedition EDM Library Cockpit returns a list of Characteristic objects that have the History Tracking status bit set.

Opening a History Tracking Information Window

You can return a list of History Tracking objects to the search results window, and then select and open an information window. For each change to a given Characteristic object, the History Tracking information window enables you to view who made the change, when the change occurred, the type of change action, and the old and new characteristic values.

Prerequisites

- You invoked the Xpedition EDM Library Cockpit application and connected to a database.
- Your user account has administrator privileges.
- The Show All perspective is selected.

Procedure

1. In the classification hierarchy pane, open the **Admin** module and choose the **History Tracking** class. Alternately, type Admin/History Tracking in the location bar.

The search criteria pane can be in either quick search mode or advanced search mode, depending on your preference settings. A link at the top of the search criteria pane lets you switch between modes.

In advanced search mode, you can enter search criteria on multiple tabs before executing a search. Leave the search criteria blank to return all History Tracking objects to the search results list.



Tip

For general information about searching in Xpedition EDM Library Cockpit and how to formulate a search query, refer to “Searching and Replacing” in the *Xpedition EDM Library Overview*

2. Click **Search** to return a list of History Tracking objects that match your query.
3. Click the reference button to the left of a row to display an information window in View mode.

Alternately, select a row from the list of returned objects. Then, with the cursor still in the search results area, right-click and choose **View**.



Note:

Because the software maintains the content of History Tracking objects, the items for editing are absent from the popup menu. **View** and **Delete** are the only valid choices.

The information window ([Figure 132](#) on page 250) contains only a **Top** tab and a **History** tab.

Related Topics

[Enabling History Tracking](#)

[History Tracking Object - Top Tab](#)

History Tracking Object - Top Tab

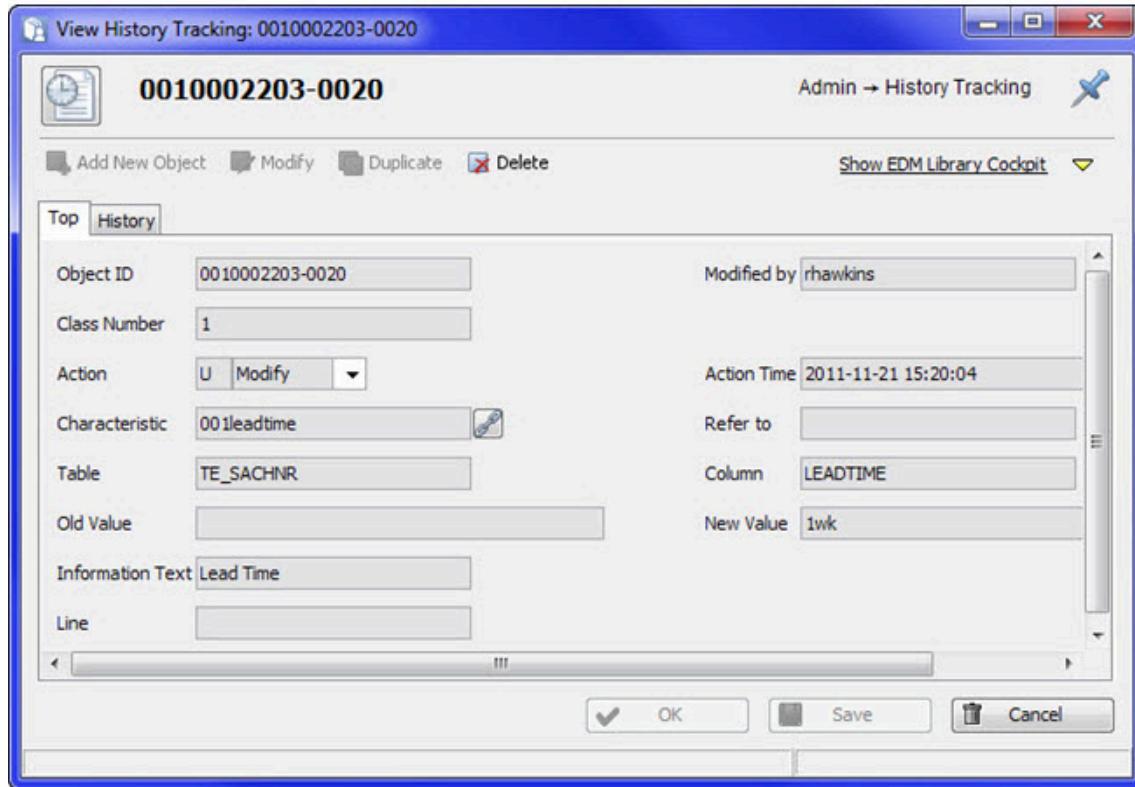
To access: [“Opening a History Tracking Information Window”](#) on page 328

The **Top** tab of a History Tracking object contains system-generated information about a characteristic for which history tracking is enabled.

Description

[Figure 162](#) shows an example history tracking information window. Because the system generates the data in the information window, you cannot change values.

Figure 162. History Tracking Information Window



Characteristics

The History Tracking information window records the following information about an object:

- The object ID.
- Who made the modification.
- The class number.
- The specific action and action time. Possible values are D(Delete), I(Insert) and U(Update).
- Information text used to identify the characteristic.
- The table and column in the database where the modification took place.
- The old value and the new value.
- If the characteristic is a list characteristic, the line that was modified.

Related Topics

[Enabling History Tracking](#)

[Opening a History Tracking Information Window](#)

MailGate Events

The MailGate Events class contains objects that define specific types of events that act as a message trigger, and a message template for the resulting notification email subject and body. An administrator can register individual users or user groups to a MailGate Events object so that when the event (such as the checking out of a specific object within an object class) occurs, a notification service sends the user or group members an automatic email. The message template subject and body text can include special variables that specify when the notification service should substitute actual values into the sent message.

Deploying an EDM Server automatically starts a notification service that can recognize MailGate events (along with other events for EDM Design products), but requires specifying the SMTP server parameters needed for mail delivery. For more information, refer to “Configuring an SMTP Server” in the *Xpedition EDM Server and Utilities Guide*.

User-specific registration rules define whether to receive an email notification when the event happens to all objects in all classes, to all objects in a specific class, to all objects in a specific catalog group, or only to a single object.

[MailGate Events Objects](#)

[Registering Users and Groups to MailGate Events](#)

MailGate Events Objects

MailGate Event objects reside in the **Admin > MailGate Events** object class and contain the template for a particular type of event notification. Registering a group or user to a MailGate Events object provides a way to receive email when the event happens on an object or objects in the database.

[Opening a MailGate Events Information Window](#)

[Default MailGate Events Objects](#)

[MailGate Events Object - Top Tab](#)

[Filtered MailGate Events Objects](#)

[Custom Events From EDM Library Applications](#)

Opening a MailGate Events Information Window

Open a MailGate Events object for viewing or editing to customize the message automatically sent to users by the event, create an event chain, or to view the users registered to the event.

Prerequisites

- You invoked the Xpedition EDM Library Cockpit application and connected to an EDM Server.
- Your user account has administrator privileges.

Procedure

1. In the classification hierarchy pane, open the Admin module and choose **MailGate Events**.

The search criteria pane is either in quick search mode or advanced search mode, depending on your preference settings. A link at the top of the search criteria pane lets you switch between modes.

In advanced search mode, you can enter search criteria on multiple tabs before executing a search. Leave the search criteria blank to return all MailGate Event objects to the search results list.



Tip

For general information about searching in Xpedition EDM Library Cockpit and how to formulate a search query, refer to “Searching and Replacing” in the *Xpedition EDM Library Overview*

2. Enter your search criteria and click **Search**.

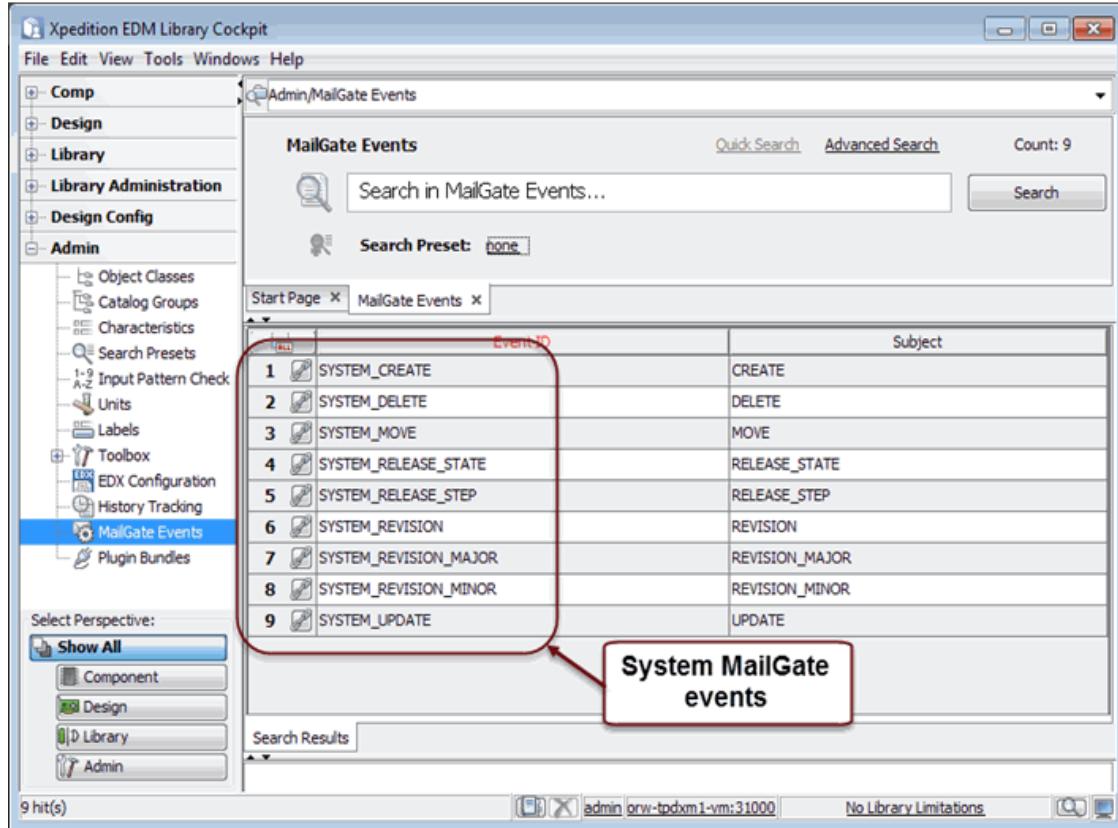
A default set of predefined system events whose Event ID begins with “SYSTEM_” are available for common actions on objects ([Figure 163](#)). Each application can also define custom events unique to that application on which to create emails.



Note:

MailGate events that begin with “DDM_” are no longer used in VX.2 and later releases.

Figure 163. The MailGate Events Object Class



A running notification service automatically adapts to changes made to the MailGate Events class for future email notifications.

- With the cursor still in the search results area, select a MailGate Events object from the list, right-click, and then choose an editing function from the popup menu.

Alternately, click the reference button to the left of a row to display an information window in view mode without using the menu.

Related Topics

[Default MailGate Events Objects](#)

[MailGate Events Object - Top Tab](#)

[Filtered MailGate Events Objects](#)

[Custom Events From EDM Library Applications](#)

[Registering Users and Groups to MailGate Events](#)

Default MailGate Events Objects

A database always contains a set of default system MailGate Events objects that can send emails to registered users.

**Note:**

MailGate events that begin with “DDM_” are no longer used in VX.2 and later releases.

Table 22. Default MailGate Events Objects

| MailGate Events Object | Action That Sends An Email |
|------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SYSTEM_CREATE | An object is created. |
| SYSTEM_UPDATE | An object is modified. |
| SYSTEM_DELETE | An object is deleted. |
| SYSTEM_MOVE | An object is moved. |
| SYSTEM_REVISION | A new revision of an object is created (used for objects without major/minor versioning). |
| SYSTEM_RELEASE_STEP | An object moves to the next release step (that is, when a user executes the Release > Process menu item on an object). |
| SYSTEM_RELEASE_STATE | An object moves to the next release state (that is, when someone executes the Release > Status menu item on an object). |
| SYSTEM_REVISION_MAJOR | An object moves to the next major revision level (used for objects with major/minor versioning; refer to “ Release and Revision Control ” on page 129). |
| SYSTEM_REVISION_MINOR | An object moves to the next minor revision level (used for objects with major/minor versioning; refer to “ Release and Revision Control ” on page 129). |

Related Topics

[Opening a MailGate Events Information Window](#)

[MailGate Events Object - Top Tab](#)

[Filtered MailGate Events Objects](#)

[Custom Events From EDM Library Applications](#)

[Registering Users and Groups to MailGate Events](#)

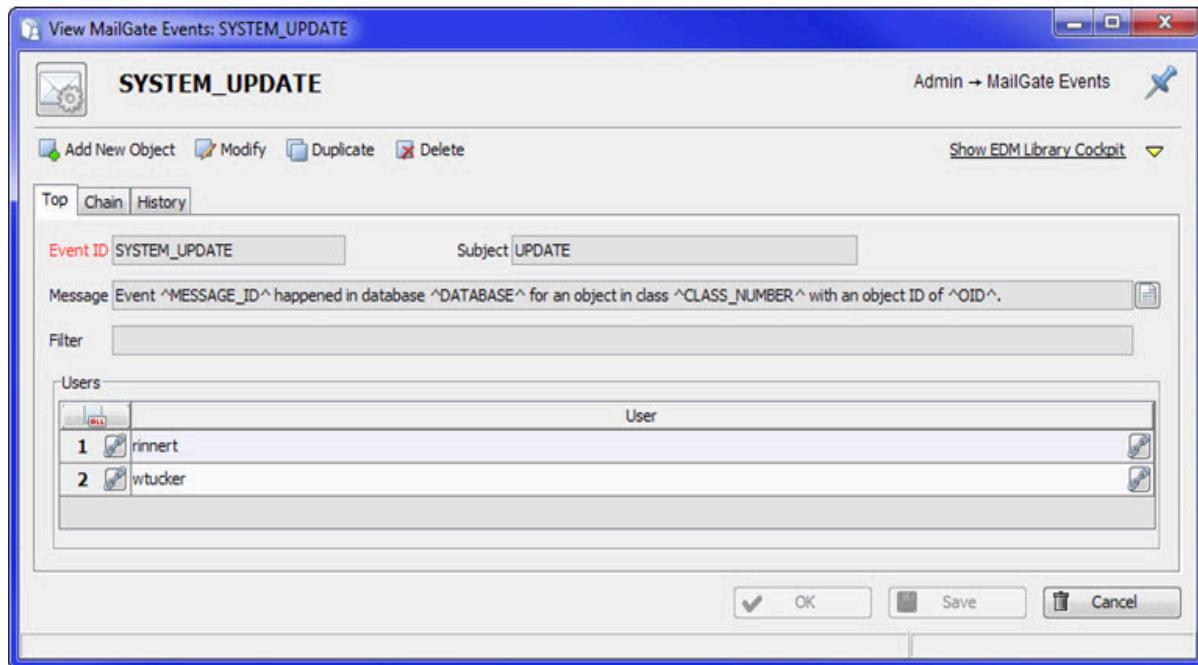
MailGate Events Object - Top Tab

To access: “[Opening a MailGate Events Information Window](#)” on page 333

The **Top** tab of a MailGate Event object describes the event, specifies the message to send when the event occurs, and contains a list of users registered to the event.

Description

Figure 164. Example MailGate Event Object - Top Tab



Characteristics

- **Event ID** — Identifies the event. By convention, system events begin with "SYSTEM_". Custom application events should add an application prefix to minimize confusion with other application events. For example, use an Event ID of MYAPP_CHECK_IN and MYAPP_CHECK_OUT instead of just CHECK_IN or CHECK_OUT.
- **Subject** — The subject line of the sent message.
- **Message** — A template for the message text. To edit the message, click the button to the right.

The notification service replaces a variable bracketed by caret characters before sending the email. For example, ^CLASS_NUMBER^ returns the class number for the object. To encode a value with HTML tags before sending the message, use the ^encode(variable_name)^ format instead (for example, ^encode(OID)^).

All events provide these global variables:

- MESSAGE_ID is the Event ID value in the MailGate Events object that triggered the email.
- DATABASE returns the name of the database where the event originated.
- CLASS_NUMBER is the class number where the affected object resides.
- OID is the object ID of the affected object.

Custom events might provide additional variables.



Note:

To add a value from the 111ersteller (Requestor) characteristic of a Request object to the generated message, you must put the characteristic name in the filter field of the event definition, otherwise the requestor name is not added to the email message.

- **Filter** — Defines one or more conditions that trigger the event in specific object classes. Filtered events must follow syntax rules and involve using the **Chain** tab as described in “[Filtered MailGate Events Objects](#)” on page 337.
- **Users** — Contains system-generated information that shows the users or groups registered to receive email about the event.

Related Topics

[Opening a MailGate Events Information Window](#)

[Default MailGate Events Objects](#)

[Filtered MailGate Events Objects](#)

[Custom Events From EDM Library Applications](#)

[Registering Users and Groups to MailGate Events](#)

Filtered MailGate Events Objects

The default SYSTEM_UPDATE object is an example of an unfiltered event that triggers the notification service to send an email message when the event occurs on any class. You can create a new MailGate Events object with a value in the Filter field on the **Top** tab to define an event that only occurs under certain conditions for certain classes.

A filtered event must be a child of a more broad parent event. For example, you might create a filtered event named SYSTEM_UPDATE_FILTERED that only operates on two object classes and specify it as a child of the SYSTEM_UPDATE event. The **Chain** tab lists any parents or children of an event. When the parent event matches a delivered message, the event forwards to all dependent filtered events. This lets you build filter chains where one MailGate event triggers another MailGate event that triggers a third MailGate event, and so on.

A filtered event provides more freedom to decide when to send an email notice of that event. Furthermore, the message template associated with the filtered event can also specify values of characteristics on the processed object. For example, you might use a filtered event to restrict SYSTEM_UPDATE messages to only objects with a specific Status value.

When creating a filtered event, you must specify the class for which the event is valid in the Filter field using the following format:

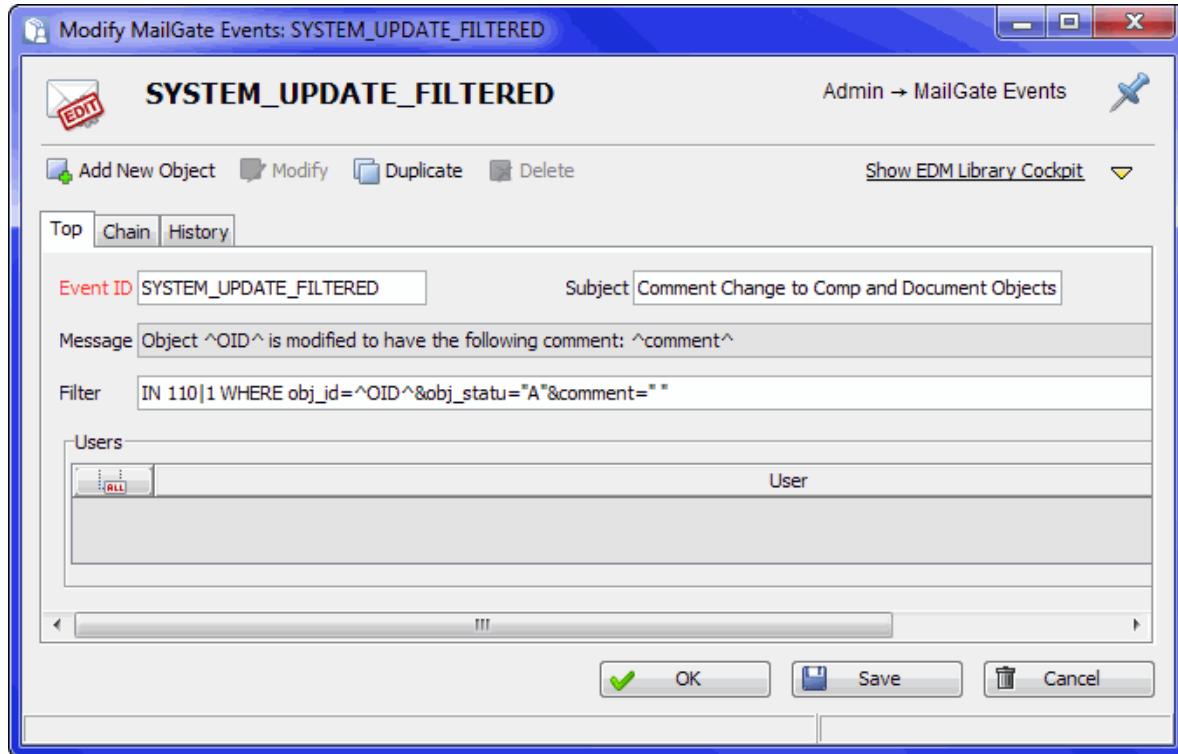
```
IN <class(es)> WHERE <characteristic>=<value>
```

The message and subject template of a filtered event can also contain wildcards for fields used in the filter.

When specifying more than one class, use a pipe (|) to separate class numbers. Multiple restrictions can follow each other separated by an ampersand (&). For example, [Figure 165](#) shows a filtered MailGate event named SYSTEM_UPDATE_FILTERED with the following filter definition in the Filter field:

```
IN 110|1 WHERE obj_ID="^OID^"&obj_statu="A"&comment=" "
```

Figure 165. Example Filtered MailGate Event (Top Tab)



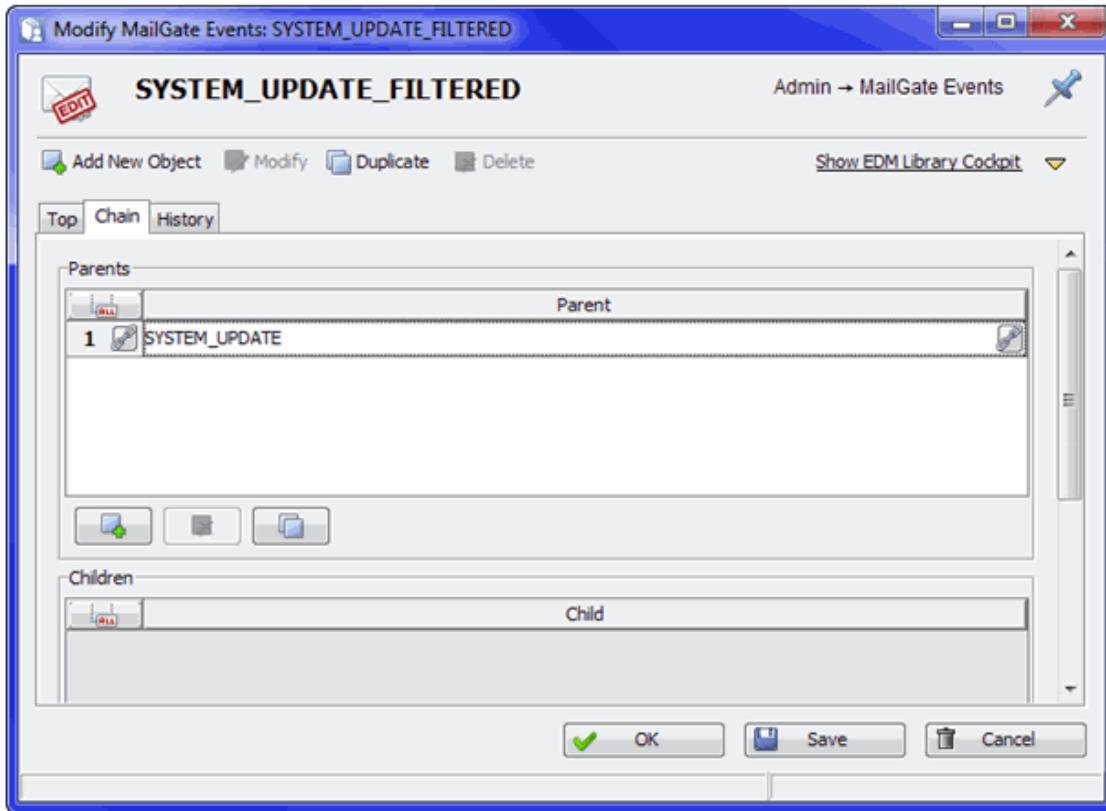
In this example, the filter only operates on object classes 110 (documents) and 1 (components) and looks for three conditions before sending an event message:

- The object has a unique identifier, denoted by ^OID^.
- The object status is "A" (approved).
- The object has a comment. The empty field in the restriction (&comment=" ") retrieves values for that characteristic without a filter. Thus, any object with a value for the comment characteristic qualifies.

The message template can use conditions specified in the restriction list. In [Figure 165](#), the message template uses the value of the comment characteristic in the message sent to the user.

To use the filter, the **Chain** tab must identify a parent event. For the SYSTEM_UPDATE_FILTERED event, the parent is the SYSTEM_UPDATE event ([Figure 166](#)).

Figure 166. Example of a Filtered MailGate Event (Chain Tab)



Adding the SYSTEM_UPDATE event to the **Chain** tab of the SYSTEM_UPDATE_FILTERED object as a parent causes the software to automatically list the SYSTEM_UPDATE_FILTERED object as a child on the SYSTEM_UPDATE object.

Related Topics

[Opening a MailGate Events Information Window](#)

[Default MailGate Events Objects](#)

[MailGate Events Object - Top Tab](#)

[Custom Events From EDM Library Applications](#)

[Registering Users and Groups to MailGate Events](#)

Custom Events From EDM Library Applications

A developer that writes an API program to incorporate custom events into an EDM Library application must include code that sends the events to the "DF_MailGate" channel.

Any custom event must always define the following four global properties:

- MESSAGE_ID is used by the service to identify the event in the MailGateEvents(35) class. The type is string.
- DATABASE returns the name of the database where the event originated. The type is string.
- CLASS_NUMBER is the class number of the object that generated the event. The type is integer.
- OID is the object ID of the object that generated the event. The type is string.

The application can also add custom properties to an event that replace wildcards in the subject and message templates of the event. For example, the following code for a new event defines the four global properties along with a custom property named LAST_MODIFIED:

```
...
import com.mentor.datafusion.oi.OIException;
import com.mentor.datafusion.oi.OIOBJECTMANAGERFACTORY;
import com.mentor.datafusion.oi.OISERVERCONNECTION;
import com.mentor.datafusion.oi.LOGIN.OIAUTHENTICATE;
import com.mentor.datafusion.oi.LOGIN.OIAUTHENTICATEFACTORY;
import com.mentor.datafusion.oi.NOTIFICATION.CHANNEL;
import com.mentor.datafusion.oi.NOTIFICATION.MESSAGE;
import com.mentor.datafusion.oi.NOTIFICATION.NOTIFICATIONSERVICE;
import com.mentor.datafusion.oi.NOTIFICATION.NOTIFICATIONSERVICEHELPER;
import com.mentor.datafusion.oi.NOTIFICATION.PROPERTY;
...
OIAUTHENTICATE authenticate =
OIAUTHENTICATEFACTORY.CREATEBATCHAUTHENTICATE("my_login_profile");
OIOBJECTMANAGERFACTORY factory = authenticate.LOGIN("my_application");
OISERVERCONNECTION connection = factory.getSERVERCONNECTION();
NOTIFICATIONSERVICE service =
NOTIFICATIONSERVICEHELPER.GETNOTIFICATIONSERVICE(connection);
CHANNEL channel = service.getChannel("DF_MailGate");
MESSAGE msg = new MESSAGE();
// Required Properties
msg.add(new PROPERTY("MESSAGE_ID", service.createAny("DDSA_CHECK_IN")));
msg.add(new PROPERTY("CLASS_NUMBER", service.createAny(10)));
msg.add(new PROPERTY("OID", service.createAny("object id")));
// Application-specific Properties
msg.add(new PROPERTY("LAST_MODIFIED", service.createAny("10/10/15")));
channel.send(msg);
```

Related Topics

[Opening a MailGate Events Information Window](#)

[Default MailGate Events Objects](#)

[MailGate Events Object - Top Tab](#)

[Filtered MailGate Events Objects](#)

[Registering Users and Groups to MailGate Events](#)

Registering Users and Groups to MailGate Events

Registering to a Mailgate event and creating one or more rules define the conditions under which to receive the event email. For example, one user registered to the SYSTEM_CREATE event might want to receive event email whenever a new object is created anywhere in the database. Another user registered to the SYSTEM_UPDATE event might want to receive the event email when only a specific object is updated.

Restrictions and Limitations

- Administrator privileges are required to register other users and groups to MailGate events. Without Administrator privileges, individual users can only register events and create event rules for their own user account.

Prerequisites

- An administrator has defined the SMTP server parameters needed for mail delivery by the EDM Server.
- Any user or user group that you want to register for a MailGate event must have a valid email address and must inherit the notification role (a user has the notification role by default if they are a member of the Design Administrators or Project Manager groups).
- You have invoked the Xpedition EDM Library Cockpit and connected to a database through the EDM Server.

Procedure

- In Xpedition EDM Library Cockpit, choose the **Tools > Administration > MailGate Configuration** pulldown menu item.
- In the MailGate Configurator window, make sure the view is set to “Event.”



Tip

The Event view lets you assign users and user groups to events. The User view lets you assign events to users or user groups.

- Click the **Add Event** button (



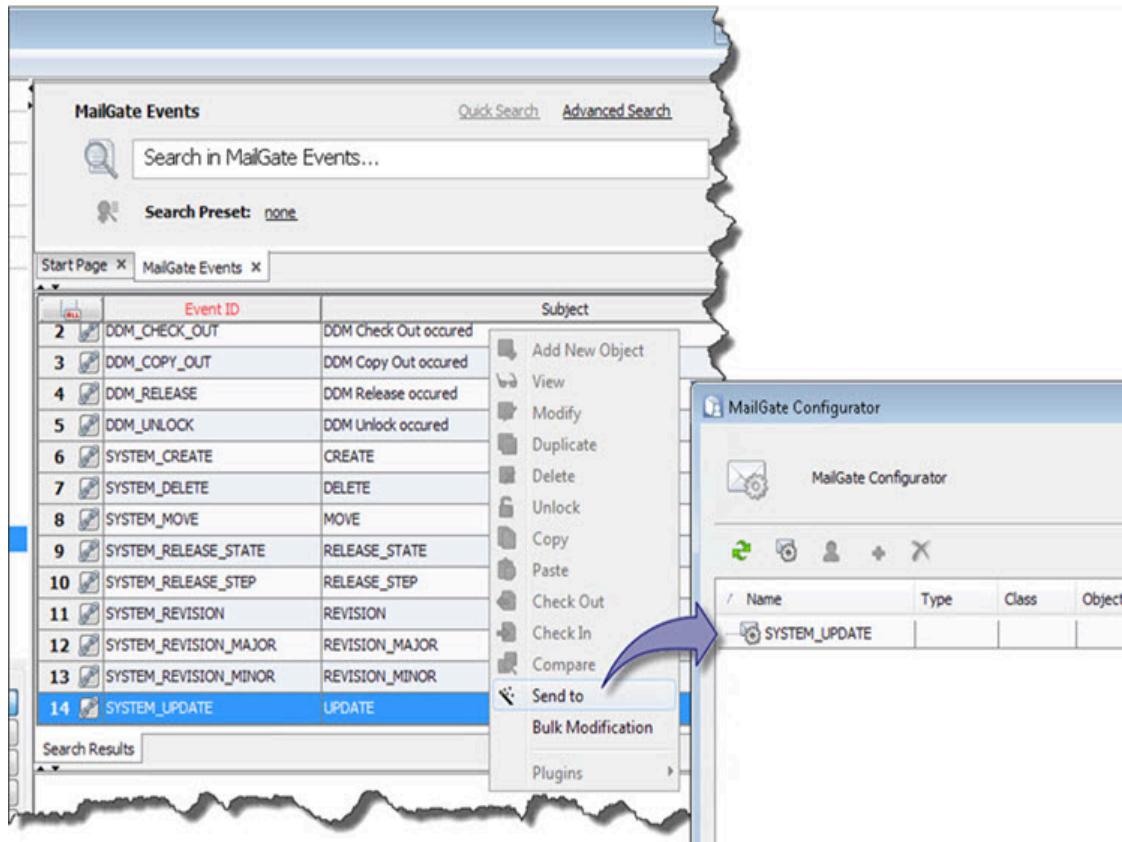
).

The search pane for the MailGate Events object class displays in Xpedition EDM Library Cockpit and the cursor changes to the magic wand.

- Perform a search to return a list of MailGate Events objects to the search result area.
- Select an object and either double-click or right-click and choose the **Send to** popup menu item.

The object returns to the events list in the MailGate Configurator ([Figure 167](#)).

Figure 167. Sending a MailGate Event to the MailGate Configurator



6. Optionally, repeat steps 3, 4, and 5 to populate the MailGate Configurator with multiple events before continuing.
7. Select the row containing the event in the MailGate Configurator and choose the **Add User(s)/Group(s)** button ().

The search pane for users displays and the cursor again changes to the magic wand.



Note:

If you do not have administrative rights, you can only add yourself to an event, and you cannot see any other users that might be registered to events in the MailGate Configurator.

8. Perform a search to return a list of users or user groups to the search result area.

Users have a value of 1 in the Type column, while user groups have a Type value of 2.



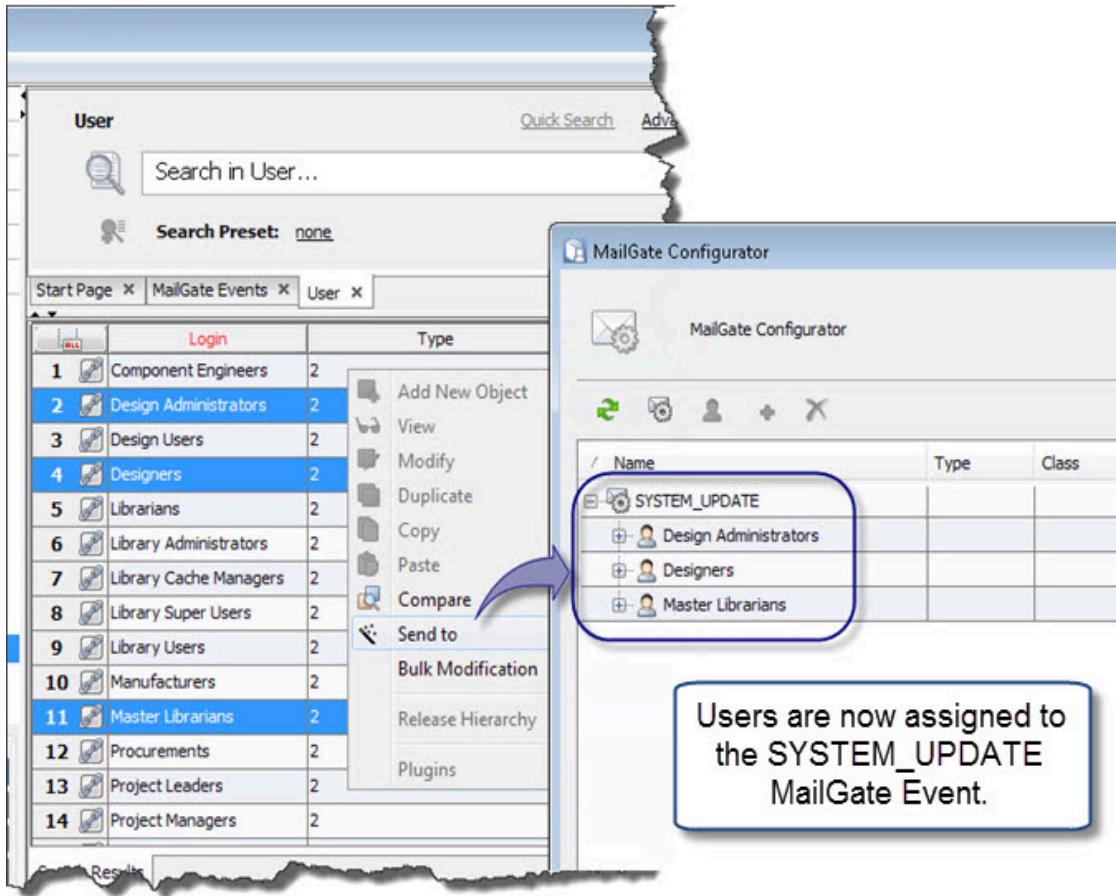
Note:

Any user or user group that you assign to an event must have a valid email address defined for the account and must inherit the notification role.

9. Select one or more user or user groups to register to the event and either double-click or right-click and choose the **Send to** popup menu item.

The users or user groups display beneath the event in the MailGate Configurator window. **Figure 168** shows three user groups assigned to the SYSTEM_UPDATE event in the MailGate Configurator window.

Figure 168. Assigning Users to the MailGate Event



10. Select a user in the list and expand the hierarchy to show any rule(s) defined for that user.

- To add a rule, click the **Add Rule** button (), and then specify values in the Type, Class, and Object columns as shown in [Table 23](#).
- To delete a rule, select the rule row and click the **Delete** button ().

Table 23. Rules List Box Columns

| Column | Value |
|--------|--------------------------------------|
| Type | Choose one of the following options: |

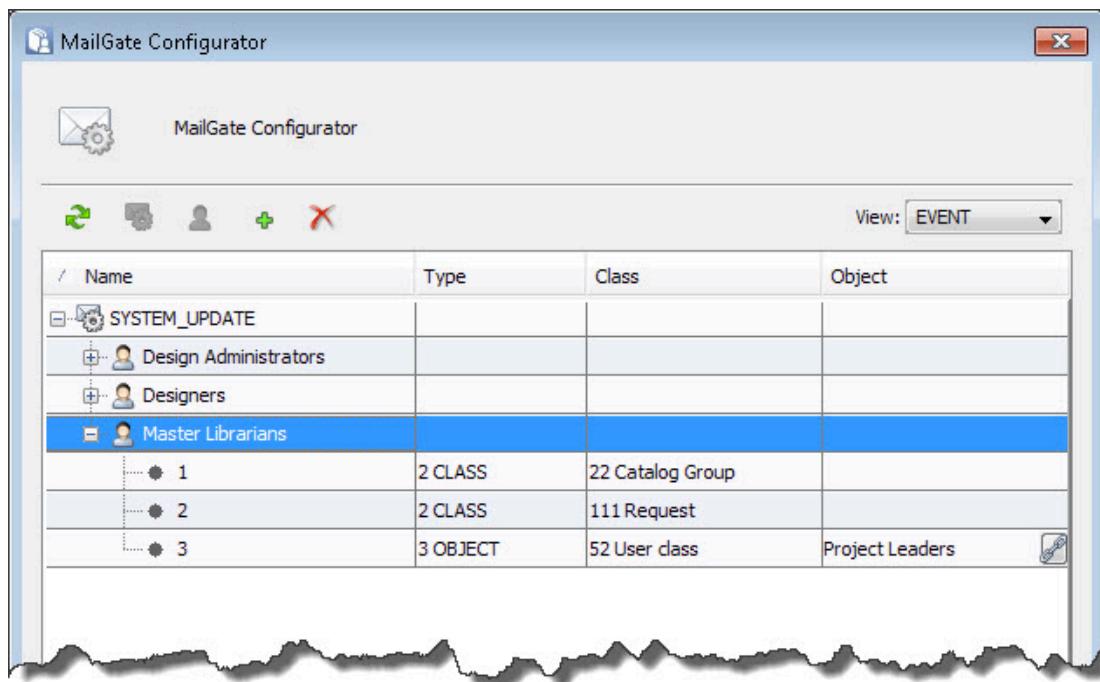
| Column | Value |
|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | <ul style="list-style-type: none">• NONE — Disable the current rule in this row. Values in the Class and Object columns have no effect.• ALL — Receive an event email if the event happens to any object in any object class in the database. Values in the Class and Object columns have no effect.• CLASS — Receive an event email if the event happens to any object belonging to the object class given in the Class column. Typing a Class column value permits it to differ from the choices in the Class option list. When using the CLASS option, the Object column value has no effect.• OBJECT — Receive an event email if the specified event happens to an object in the Object column in a class specified in the Class column. You must select the Class column value from the option list available for the Class field. |
| Class | The number of the object class affected by the specified event according to the Type column definition. You must choose from the option list choices if the value in the Type column is OBJECT. If required, you can edit the characteristic definition to add classes to the option list (refer to “ Opening a Characteristic Information Window ” on page 146). |
| Object | The key of an object affected by the specified event according to the definition in the Type and Class columns. To search for available objects and return a key value to the column, use the reference button at the right. |

11. Click **OK** to close the MailGate Configurator and save changes.

Examples

In [Figure 169](#), members of the Master Librarians user group registered to a SYSTEM_UPDATE event receive an email when an object in the Catalog Groups class (22) or the Design Model class (111) changes. In addition, user group members receive an email if the Project Leaders user group changes (for example, when a user is added or deleted from the group).

Figure 169. Rule Set for the Master Librarians Group Assigned to the SYSTEM_UPDATE Event



Related Topics

[User Administration](#)

[Characteristic Administration](#)

[MailGate Events Objects](#)

Plugin Bundles

The Plugin Bundles object class enables the integration of plug-in Java applications or subprograms into the Xpedition EDM Library Cockpit application. EDM Library Cockpit implements bundles in the form of a Java *.jar* file, which is stored in the database as a binary large object (BLOB) and attached to the Bundles object.

Each Bundles object represents a Java application that either runs automatically upon invoking Xpedition EDM Library Cockpit or that can be launched by a user (for example, through a menu item or action button), or launched through an API call. The **Tools > Plugins** pulldown menu or **Plugins** popup menu in Xpedition EDM Library Cockpit provide a default location from which to launch plug-in applications.

Bundles objects might be present if a developer has created one or more customized Java programs using the object interface (OI) API and integrated those program into EDM Library Cockpit. Java programs represented by the *.jar* file can either be signed with a digital security signature, or unsigned.

[keygen Command](#)

[Running keygen Using the GUI](#)

[Plugin Bundles Object - Top Tab](#)

keygen Command

Before creating a Bundles object and attaching the *.jar* file, a developer can run the <SDD_HOME> \dms\bin\keygen.bat (Windows) or the <SDD_HOME>/dms/bin/keygen.sh (Linux) command to create a digital security signature for the *.jar* file. With a digital security signature, the program cannot be executed without signature verification.



Note:

The **keygen** command cannot create a digital signature for a *.jar* file if the file was already signed by java **jarsigner**. Always use **keygen** before **jarsigner**



Note:

The **keygen** program is not available in a client software tree and must be invoked from a server software tree.

Syntax

```
keygen -help|-interactive|-file path|-folder path|-tmpfolder path
```

Arguments

Only one of the following flags or switches can follow the **keygen** command:

- **-file path**

- New list item

Use the **-file** switch to specify a path to a *.jar* file for which to create a digital signature.

- **-folder path**

Use the **-folder** switch to specify a path to a folder containing *.jar* files. The **keygen** command then creates a digital signature for all *.jar* files inside the folder.

- **-help**

Display help information about command syntax.

- **-interactive**

Display a GUI with which to assemble a list of *.jar* files from multiple directories.

- **-tmpfolder path**

Use the **-tmpfolder** switch to specify the path to a temporary folder containing *.jar* files to sign.

Related Topics

[Running keygen Using the GUI](#)

[Command Line Syntax Conventions](#)

Running keygen Using the GUI

You can invoke the **keygen** program with a GUI, which is useful to assemble a list of *.jar* files from multiple directories.

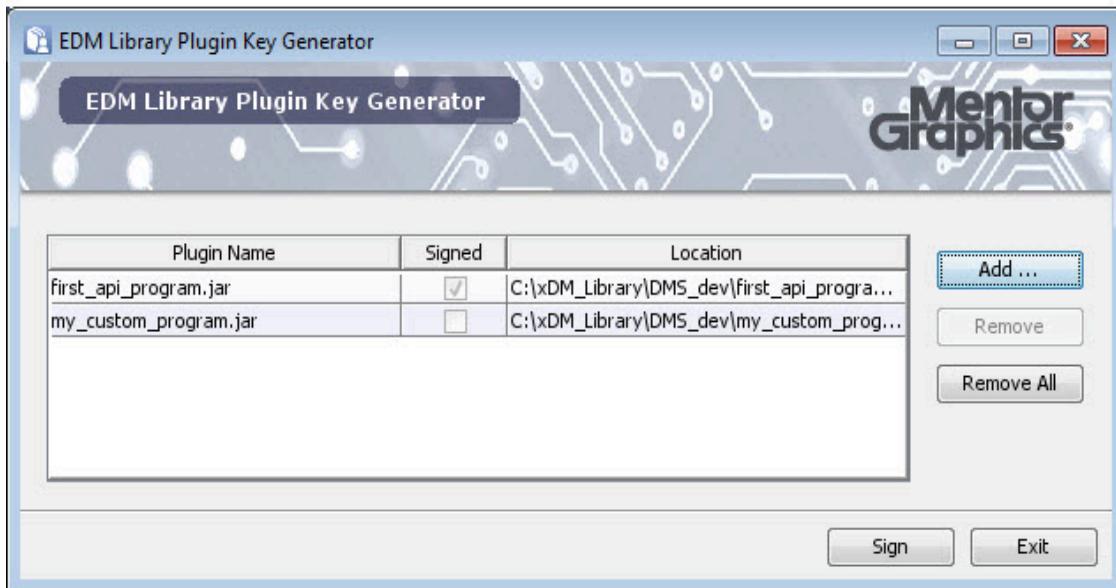
Procedure

1. Navigate to the <SDD_HOME>\dms\bin folder in a command window and type the following:

```
keygen -interactive
```

The Plugin Key Generator window displays (Figure 170). Typing the **keygen** command without any switches also invokes **keygen** in interactive mode.

Figure 170. Plugin Key Generator Window



2. Click **Add** to navigate to and include the *.jar* files to sign.
3. After you have all the *.jar* files in your list to sign, click **Sign**.
A check shows in the list as keygen creates a digital signature for each *.jar* file.
4. Click **Exit** to exit the **keygen** program.

Results

The signed *.jar* file or files can now be attached to a bundles object for execution in Xpedition EDM Library Cockpit.

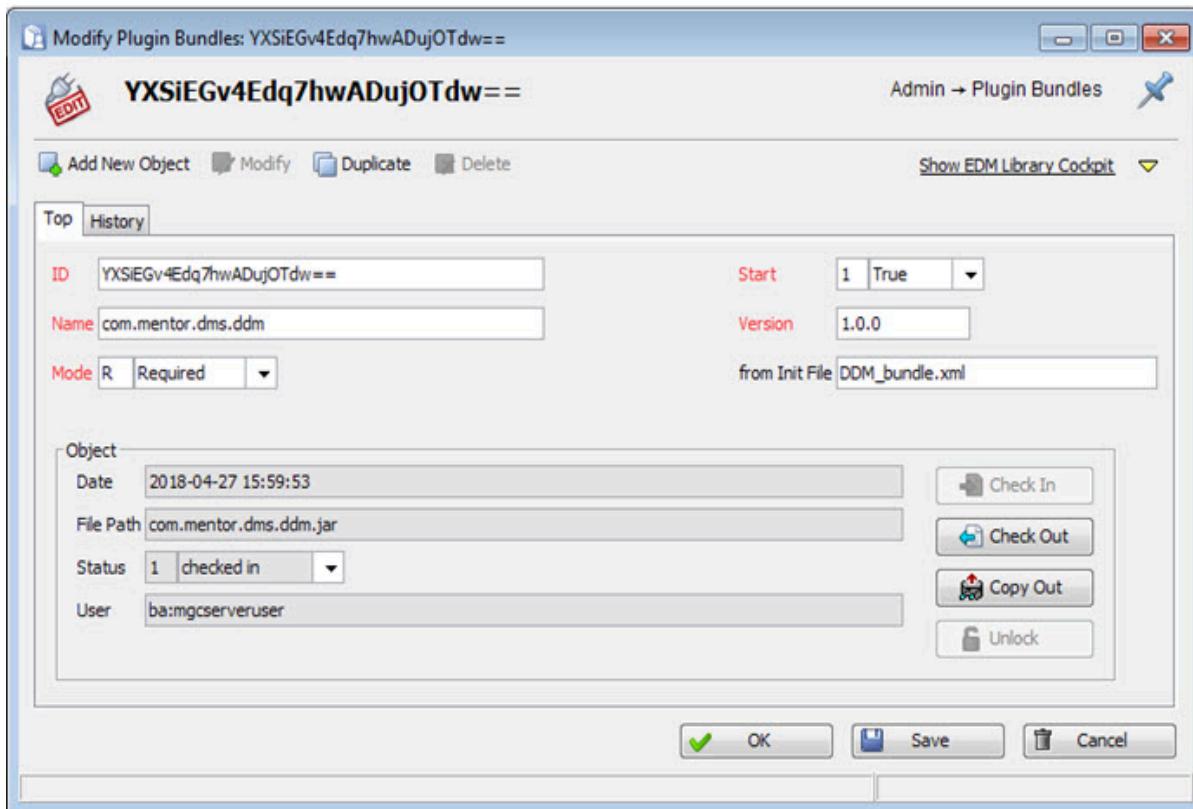
Plugin Bundles Object - Top Tab

To access: **Admin > Plugin Bundles** in the classification pane

Depending on the settings in the Plugin Bundles object, attached Java programs can be either automatically loaded upon invoking Xpedition EDM Library Cockpit or started on demand by the user.

Description

Figure 171. Top Tab of a Plugin Bundles Object



Because Bundles objects represent system functionality or a custom Java program created by a developer, administrators usually do not edit Bundles objects unless it is to disable the functionality from displaying in Xpedition EDM Library Cockpit. Disabling the functionality represented by the Bundle object involves changing the value of the Mode characteristic to "D" (Disabled) and reloading the data model.

Characteristics

- **ID** — A system-generated value that uniquely identifies the Bundles object.
- **Start** — Controls the life cycle of the bundle. If the value of Start is True, and the value of Mode is Required, then the Java program represented by the Bundles object automatically loads and starts upon invoking Xpedition EDM Library Cockpit. Otherwise, a user starts the Java program on demand (for example, through a pulldown menu, action button, or API call).
- **Name** — The name of the bundle, as assigned by the developer. With default system bundles, the convention is to name the bundle the same as the .jar file, but without the file extension.
- **Version** — The version of the Java program attached to the Bundles object.

- **Mode** — Indicates if the attached plug-in Java program is required, optional, or disabled.
 - **Required** — The application loads when invoking Xpedition EDM Library Cockpit. If the value of Start is True, then the Java application also automatically starts.
 - **Optional** — Xpedition EDM Library Cockpit references the application attached to the bundle but does not automatically load nor start it when invoking. The program can then be started on demand (for example, through an API call).
 - **Disabled** — The plug-in application is disabled and cannot be loaded or started.
- **from Init File** — Contains a system-generated value if the Bundles object was automatically created by importing the contents of an initialization file using the **batchadmin** program or by deploying the EDM Server with EDM Library services.
- **Object** — Contains information about the *jar* file imported into the database as a BLOB and attached to the Bundles object.

Chapter 5

Administrative Programs

Administrative programs can assist in a management task or provide administrative information. Items in the **Tools** pulldown menu in Xpedition EDM Library Cockpit control the unlock manager, path query, and database backup and restore functionality. In addition to menu items, the **batchadmin** shell program can initialize the database schema with object classes, catalog groups, and characteristics from information in a *.init* file. After extending the EDM Library data model with new object classes, catalogs, and characteristics, use the **data_model_checker** command to test the validity of the entire data model, or the **domain_model_checker** command to check that unique domain model name values are unique.

[Tools Menu Administration Programs](#)

[The batchadmin Program](#)

[data_model_checker](#)

[domain_model_checker](#)

Tools Menu Administration Programs

The **Tools > Administration** menu item in the Xpedition EDM Library Cockpit session window contains several administrative submenu items.

- **Reload Data Model** — Exit the application, restart all EDM Library superservices, and then restart Xpedition EDM Library Cockpit. **Reload Data Model** forces a cache update of all local clients so that all users connecting to the database can see the change. Refer to “[Reinitializing the Cache](#)” on page 31.
- **Compliance** — Contains subitems to load a compliance configuration and open a compliance manager report window. The **Compliance** submenu is only present when EDM Library Services is initialized with the compliance manager module. Refer to the *Xpedition EDM Library Compliance Management Module Guide*.
- **Configuration Settings** — Contains an item to invoke the EDM Batch Login Setup program. Refer to Automatic Login Configurations in the *Xpedition EDM Library Overview*.
- **Designer Settings** — Provides subitems to display the toolboxes used for the interface to Xpedition Designer. For toolbox information, refer to “[The Toolbox](#)” on page 260.
- **MailGate Configuration** — Invokes the MailGate Configurator described in “[Registering Users and Groups to MailGate Events](#)” on page 341.
- **PathQuery** — Enables an administrator or API programmer to allocate a search path to characteristics that display values from foreign object classes (refer to “[Creating a Characteristic to View Data From a Foreign Class](#)” on page 204). An API client programmer can also use the path query function to find the path from a source object class to a target characteristic.

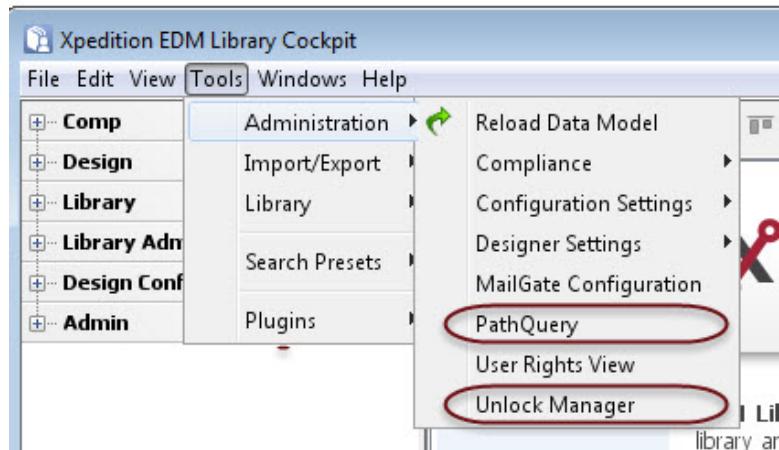
Refer to “[PathQuery Dialog Box - Migration Tab](#)” on page 355 and “[PathQuery Dialog Box - Path Builder Tab](#)” on page 359.

- **User Rights View** — Displays the User Rights Window (refer to “Viewing User Account Rights” in the *Xpedition EDM Library Overview*).
- **Unlock Manager** — Provides a method for administrators to view locked objects and then force an unlock of object classes or individual objects, if required.

For more information, refer to “[Unlock Manager Window](#)” on page 353.

Because most menu items have discussions in other locations (as cited in the previous list), this chapter discusses only the menu items circled in [Figure 172](#).

Figure 172. The Tools > Administration Pulldown Menu



[Unlock Manager Window](#)

[PathQuery Dialog Box - Migration Tab](#)

[PathQuery Dialog Box - Path Builder Tab](#)

Unlock Manager Window

To access: **Tools > Administration > Unlock Manager**

In Xpedition EDM Library Cockpit, the Unlock Manager displays locked objects and provides a way to unlock them.

Description

When a user has an object open for modification, EDM Library locks that object to prevent changes by other users. Closing the object information window releases the lock. EDM Library can also lock objects at other times, such as when releasing or revisioning an object, or during bulk modification. External programs might also temporarily lock objects as the program accesses those objects. Sometimes an unexpected network or system problem can result in an object remaining in an inappropriate locked state.

To unlock an object that should not be locked, use the Unlock Manager window.

Usage Notes

To display the Unlock Manager window, choose the **Tools > Administration > Unlock Manager** pulldown menu item. The two dropdown choices in the upper right switches between two view orientations:

- User view ([Figure 173](#)) shows a list of users in the Name column. You can click a user name to expand the hierarchy and show a sublist of object classes and individual objects that the user has locked.
- Class view ([Figure 174](#)) shows a list of classes in the Name column. You can click a class name to expand the hierarchy and show a sublist of users who have objects in that class locked.

Figure 173. Unlock Manager Window (User View)

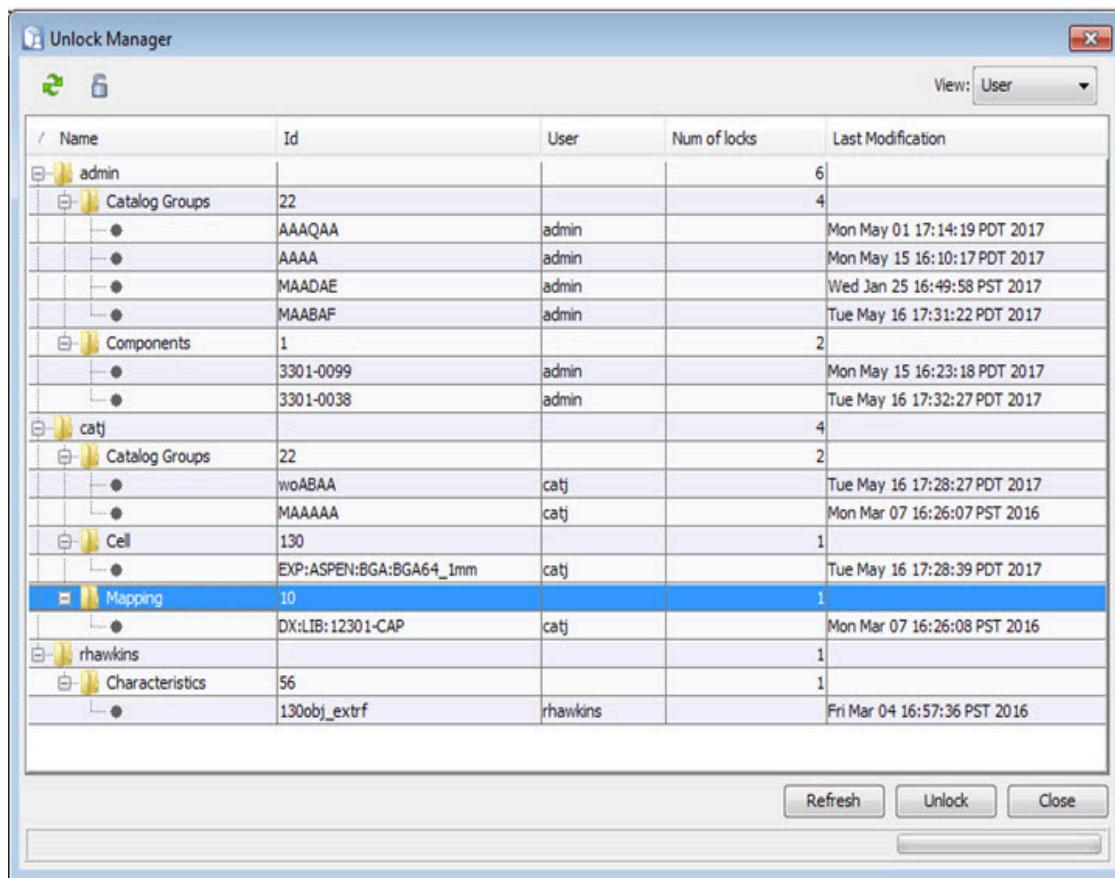
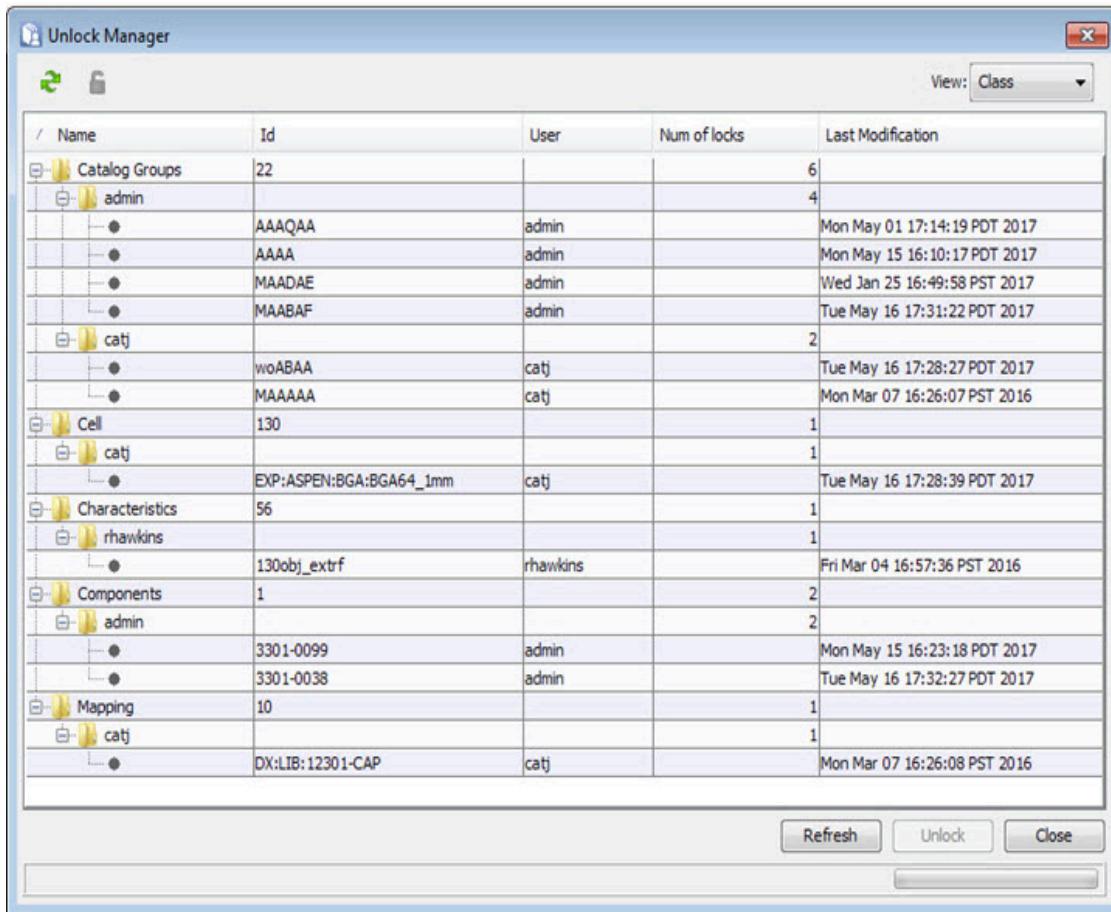


Figure 174. Unlock Manager Window (Class View)



To unlock one or more objects, select those object rows and click the **Unlock** button. Most users can usually unlock objects that they have locked. But, to see and unlock objects owned by others, you must be logged with an account that has administrative rights.

When many users are connected to a server, objects are constantly being locked and unlocked. To avoid a constantly updating display that makes it hard to look for the object you want to unlock, EDM Library displays the state of locked objects at the time you open the Unlock Manager window. To refresh the window with the latest system information, click the **Refresh** button.

PathQuery Dialog Box - Migration Tab

To access: **Tools > Administration > PathQuery**

In Xpedition EDM Library Cockpit, an administrator can use the **Migration** tab of the Path Query dialog box to specify the search path through the data model to use when a characteristic obtains data from a foreign object class.

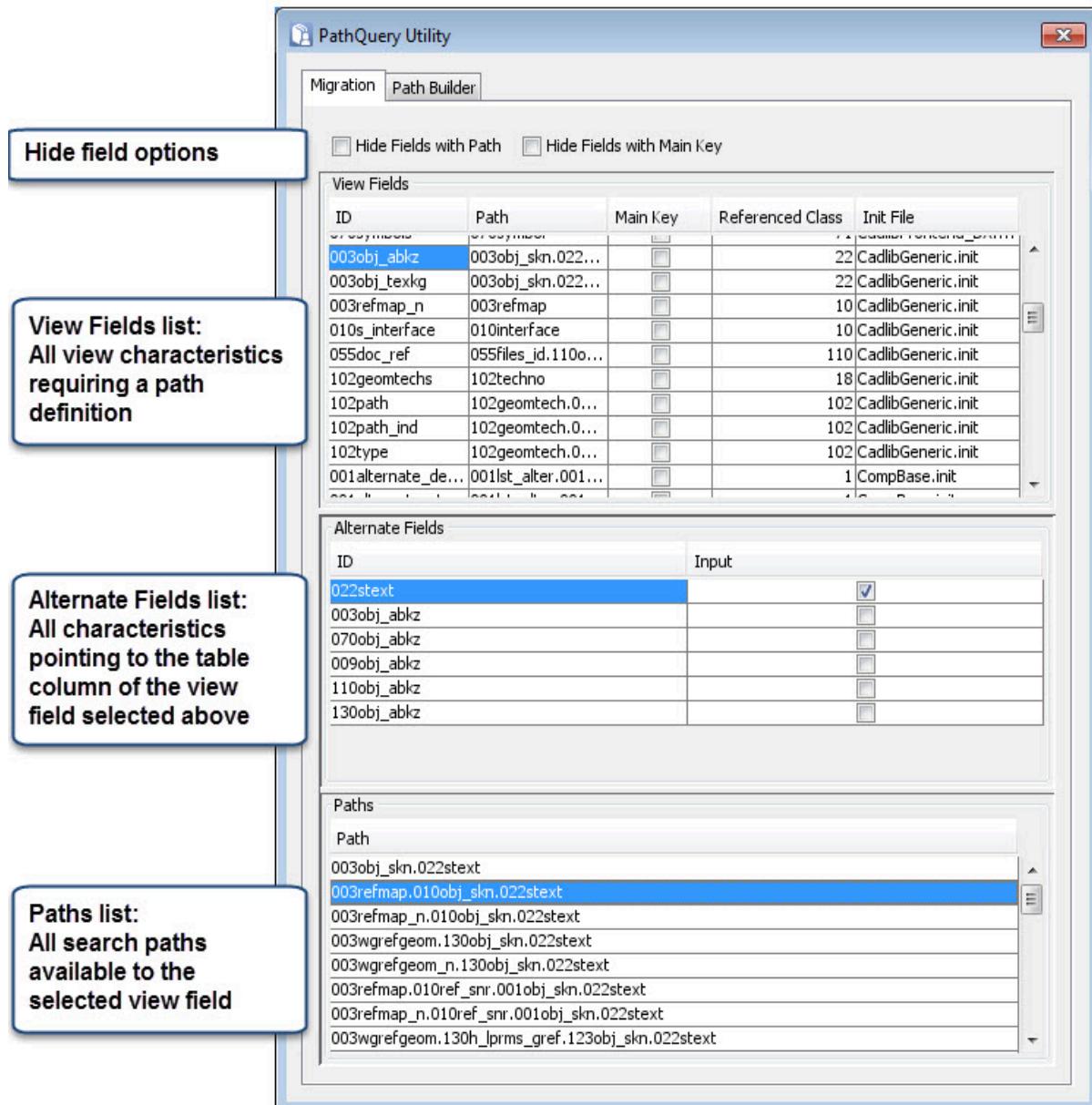


Note:

You do not have to use PathQuery functionality for default characteristics because the data model already defines all required search paths.

Description

Figure 175. The PathQuery Dialog Box - Migration Tab



Fields

- **Hide check boxes** — The following checkboxes hide rows that are optional when you assign new search paths in the View Fields list:
 - **Hide Fields with Path** — Checked, does not display characteristics in the View Fields list that already have an assigned search path in the Path column.
 - **Hide Fields with Mainkey** — Checked, does not display characteristics that have a check in the Main Key column because Main Key characteristics usually do not require a search path definition.
- **View Fields list** — Displays all characteristics that provide view information from foreign object classes and thus require a search path definition. A value is in the Path column if a characteristic already has an assigned search path definition. A check is in the Main Key column if the characteristic definition has the Main Key status bit checked.
- **Alternates Field list** — After you select a characteristic in the View Fields list, the Alternates Field list displays all characteristics that point to the same table column to obtain a characteristic value.
- **Paths list** — Shows all possible paths from the selected characteristic in the View Fields list to the selected characteristic in the Alternate Fields list. Double-clicking a row in the Paths list populates the Path column of the selected characteristic in the View Fields list.

Example

The following example assigns a search paths to a view characteristic in a customized data model (refer to [Figure 176](#)):



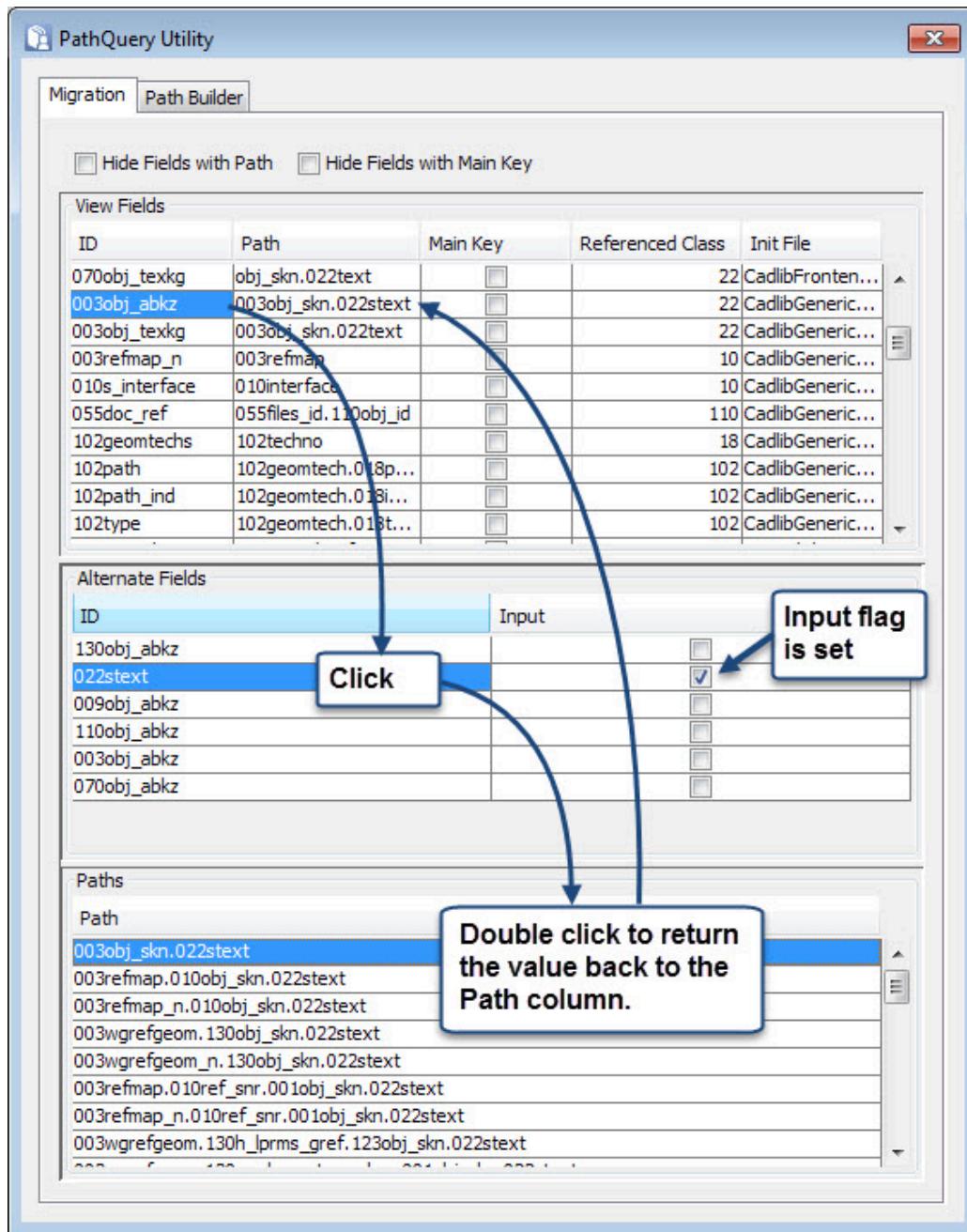
Note:

When using the default EDM data model, there is no requirement to use the PathQuery functionality because the data model already defines all required search paths.

1. In the View Fields list, select the 003obj_abkz (Abbreviated Catalog Group) characteristic in the Package object class for path assignment. All characteristics that point to the same table column (bnk_tsh) as the 003obj_abkz characteristic display in the Alternate Fields list. In this example, there are six characteristics that obtain a value from the bnk_tsh column in the te_bnk table.
2. Scroll down in the Alternate fields list until there is a characteristic with the Input status bit checked. In this example, the only input characteristic is the 022stext (Abbreviation) characteristic in the Catalog Groups class. This is the characteristic that enables a user to input data into the table column, while other characteristics in the list simply view data stored in that table column.

There is usually only one Input characteristic, although there might be more (if there are two or more input characteristics for a selected View field, carefully check if this is intentional or accidental). The Path list now shows the paths that map to the target characteristic.

Figure 176. Migration Procedure I



3. Double-click a path in the Paths list (in this example, 003obj_skn.022stext) to populate the Path column of the selected 003obj_abkz characteristic in the View Field list.



Note:

For data model consistency, the path definitions should be the same as the ones you see in the **Search Ref.** tab sheet of the source object class.

Related Topics

[PathQuery Dialog Box - Path Builder Tab](#)

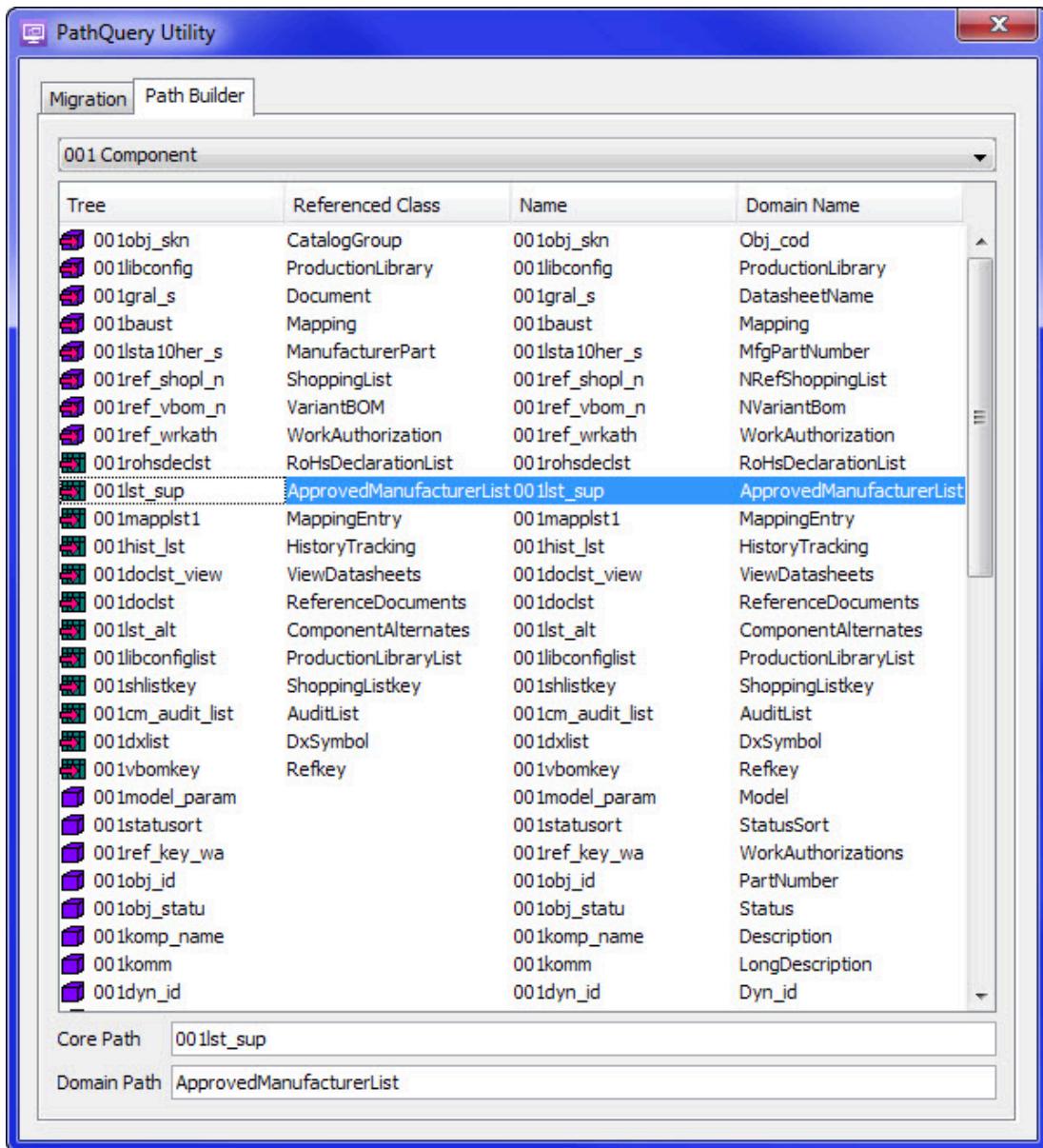
[Creating a Characteristic to View Data From a Foreign Class](#)

PathQuery Dialog Box - Path Builder Tab

To access: Tools > Administration > PathQuery

A programmer uses the **Path Builder** tab of the Path Query dialog box to find possible search paths through the data model to pass to an API client program.

Figure 177. PathQuery Window - Path Builder Tab



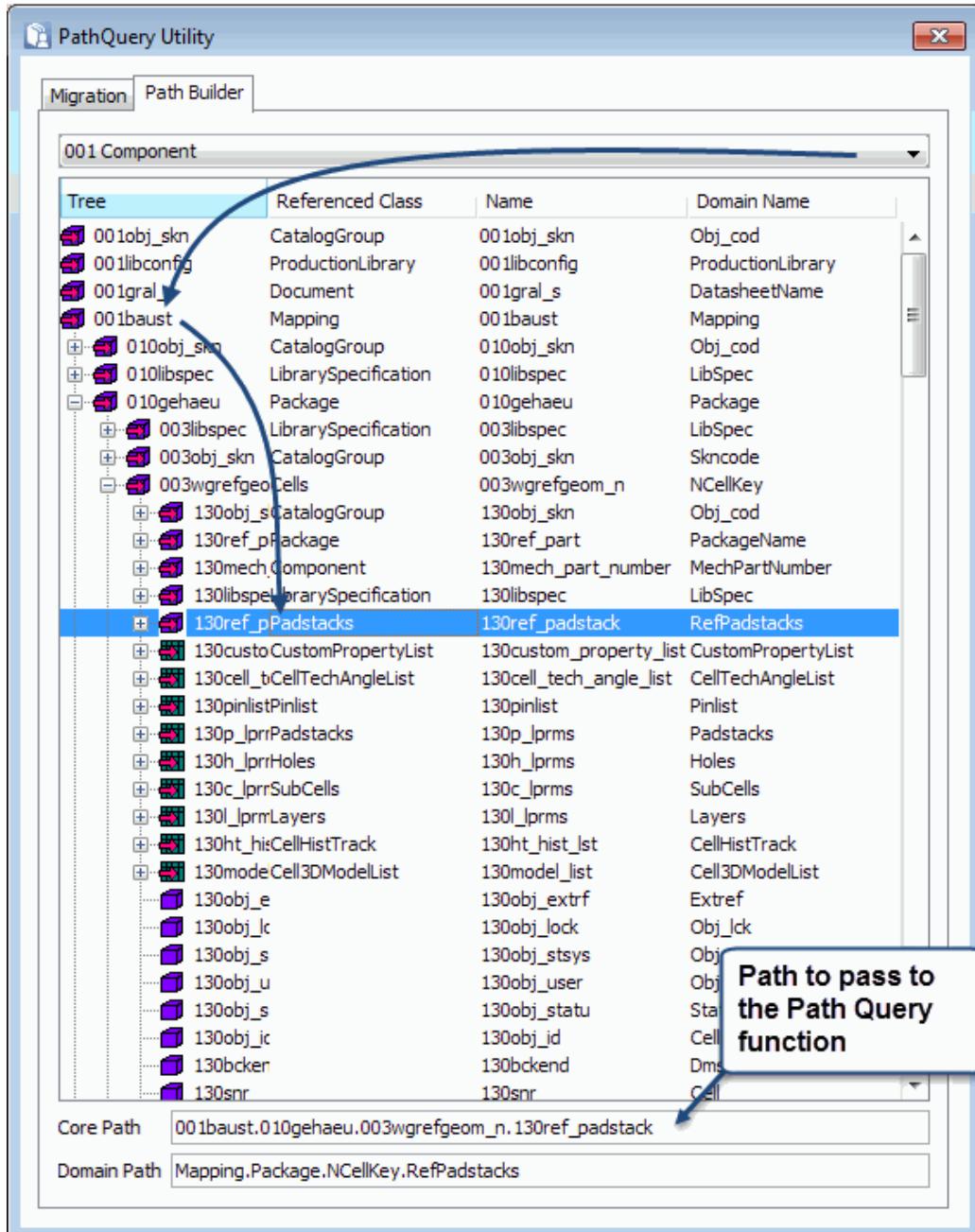
Usage Notes

The example in [Figure 178](#) shows how to use the **Path Builder** tab, assuming an API client wants to use the Path Query function and needs to specify the path from a Component object to the Padstack as an argument.

1. Select your source object class from the dropdown list at the top of the window. For example, class number 1, Component.
2. Navigate to the target object class (for example, Padstacks).
3. Select the characteristic.

The Core Path field now contains the correct path to pass to the Path Query function as a parameter (for example, 001baust.010gehaeu.003wgrefgeom_n.130ref_padstack). The Domain Path shows the same result, but replaces the internal EDM Library characteristic names by the Domain Model names.

Figure 178. The Path Builder Tab Sheet



Related Topics

[PathQuery Dialog Box - Migration Tab](#)

[Creating a Characteristic to View Data From a Foreign Class](#)

The batchadmin Program

The **batchadmin** shell program operates without a graphical user interface and uses command arguments to perform a set of database management tasks.



Note:

Because it is an administrative program, **batchadmin** is only available for invocation from a server software tree

Use the **batchadmin** command to do the following database management tasks:

- Update the EDM Library system tables.
- Load or save a ClassInit file.
- Initialize major-/minor versioning for an object class.
- Restore the default EDM database labels.
- Regenerate the database indexes.
- Find orphan column characteristics that do not have a parent list frame.

Seeing the effect of executing the **batchadmin** program requires restarting EDM Library services and the Xpedition EDM Library Cockpit application.

[About Class Init Files](#)

[Default Class Init Files](#)

[batchadmin Syntax](#)

About Class Init Files

A ClassInit file represents the structure of the database (that is, the tables and relationships), but without any actual data.

Saving a ClassInit file writes a representation of the following database items into a single file:

- Object classes
- Catalog groups
- Static and dynamic characteristics
- Relationships to dynamic characteristics

Using the **batchadmin** program to save a ClassInit file creates a backup file of the database configuration that can be used to initialize or update another database. Class, Catalog Group, and Characteristic objects contain a “from Init file” characteristic that instructs the **batchadmin** program to write the

definitions to the named initialization file, thus enabling you to separate customizations into a different loadable file.

Using **batchadmin** to load a ClassInit file updates existing tables in the database if they already exist, or creates new tables if they do not exist. Tables and columns not in the new ClassInit file remain unchanged and are not overwritten in the database. Loading a ClassInit file differs from importing a database in that a ClassInit file only contains the database structure with no actual data. Initializing a database writes the name of the initialization file into the “from Init file” characteristic on an object, which enables you to tell where the Class, Catalog Group, or Characteristic object originated.

Related Topics

[Default Class Init Files](#)

[batchadmin Syntax](#)

Default Class Init Files

Several pre-configured ClassInit files are located in the `<SDD_HOME>/dms/init` directories. The `<SDD_HOME>/dms/init/current` directory contains all current ClassInit files, while the `<SDD_HOME>/dms/init/diff_<rls>_<rls>` directory contains only ClassInit files that have changed between releases.

Using the EDM Server Cockpit web application to specify EDM Library Service parameters loads the ClassInit files from the current directory to initialize a new database, and the ClassInit files in the `diff_<rls>_<rls>` directories when updating an existing database. The `BaseClasses` file, which is always loaded before any of the other files, initializes the database to provide the Administrator classes you use to create and manage the EDM data model.



Note:

Because the order in which you load the standard initialization files is important, always use the EDM Server Cockpit instead of the **batchadmin** program to load the contents of the standard initialization files. After that, you can then use the **batchadmin** program to initialize the database with any customizations contained within customer-specific initialization files.

Related Topics

[About Class Init Files](#)

[batchadmin Syntax](#)

batchadmin Syntax

To access: In an EBS command window, type **batchadmin**

The **batchadmin** program performs several administrative tasks on tables in the EDM data model including loading the contents of a class init file into an EDM database, generating an init file from an EDM database, creating base system tables, regenerating indexes, and initializing major/minor versioning on an object class. The **batchadmin** program is only available for invocation from a server software tree.

Usage

```
batchadmin -xdm_data_dir data_directory_path
[-systemtables|-load|-restore|-migrate|-load_diff|-cleanup]
[-labels] [-indexes] [-initfile filename] [-class class_number]
[-initfilekey from_Init_File] [-db_type type] [-utf8] [-debug]
```

Arguments

- **-xdm_data_dir *data_directory_path***

The path to the EDM Server data directory. For example:

```
-xdm_data_dir C:\MentorGraphics_Data\EDM_Server-Data
```

- **-systemtables**

Creates a basic set of system tables (if the database is empty) or updates the existing set of system tables in accordance with a new database structure and license type. The **-systemtables** switch updates the following EDM Library system tables and their columns as required for proper operation of the EDM Library system:

```
te_glb_rastw
te_cls_merk
te_lab
te_obj
te_obk
te_person
te_prop_merk
te_skn
te_text
te_regel
te_user
version
tr_merk_lab
tl_obk_gui
tr_obk_label
tr_smt
tr_unit
tw_arc
tw_cir
tw_dimension
```

```
tw_grp
tw_line
tw_vtx
```

Every time you update system tables with the -systemtables switch, the program writes a row into the version table, which logs the update. Deploying EDM Services on an EDM Server automatically updates the system tables.

You must follow the -systemtables switch with the -db_type switch to indicate the EDM Library Services licensing type appropriate for the database.

- **-load/-restore**

Specifies whether to import (-load) or export (-restore) a database initialization file. When exporting, use the -utf8 switch to ensure the init file is written with UTF-8 encoding.

- **-load_diff**

Loads the contents of the init file without removing anything from the data model. The behavior differs from the -load option in that -load_diff does not remove dropdown values from other characteristics loaded from an initialization file of the same name.

- **-migrate**

Implements major-/minor versioning on a non-versioned object class or an object class that has classic versioning. The migration includes updating the data model and the data. You can only use the -migrate switch with the -class switch.

If the corresponding object class did not have versioning implemented, the -migrate option creates all major-/minor versioning characteristics in a **Major-/Minor Versioning** tab and migrates the data. If the corresponding object class already had versioning implemented, the **batchadmin** program only creates the characteristics required for major/minor versioning in a **Major-/Minor Versioning** tab, keeps existing versioning characteristics in their tab sheets, and migrates the data.

For more information about major-/minor versioning, refer to “[Release and Revision Control](#)” on page 129.

- **-cleanup**

Checks for orphan list column characteristics that do not have a parent list frame and, if found, changes their status value from “A” (Approved/Released) to “U” (Under Construction).

For example, there are list column characteristics that specify a List No. value of 1051 on the **Top** tab of their characteristic definition, but there is no parent list frame characteristic in the database with a List No. value of 1051. Using the -cleanup switch changes the status value of the orphan column characteristics to “U” so they are no longer displayed. Optionally, you can then use EDM Library Cockpit either delete those characteristics from the database, or create a new list frame characteristic that uses the orphan column characteristics.

- **-labels**

Loads the labels for menus, buttons, and so on by reading the contents of the <SDD_HOME>/dms/labels/DbeLabelsx.x file into the database. The option is useful if objects in the Labels object class have changed and you want to restore label text to the factory default values. The -labels option can only be used with the -load option (default).

- **-indexes**

Drops and recreates all indexes in the database schema, which can optimize the table storage structure and speed performance. Run the **batchadmin** program using the -indexes switch at regular intervals, especially if the following conditions are true:

- Large volumes of data have been moved within the database (for example, data has been loaded into the database using a EDM Library loader program).
- Users have noticed a performance degradation wherein EDM Library now takes several minutes to return search results that used to take much less time. If the decrease in performance is due to damaged database indexes, running the Modify Table Storage Structure can increase performance.

- **-initfile *filename***

Specifies the name of a database class initialization file to import or export.

- **-initfilekey *from_Init_file***

An extra switch that can only be used for an export. Use this switch in conjunction with -initfile. The -initfilekey switch exports all classes, characteristics (and their option list values) associated with a specific initialization file. The “from Init file” characteristic inside the **Top** tab of a class or Characteristic object, or the **Admin** tab of a Catalog Group object, defines the name of the initialization file.

- **-class *class_number***

Can be used for an export procedure, but only along with the -restore and the -initfile switches. Use -class to export an initialization file for a single object class. The -class switch also exports catalogs for the given class, labels used with those catalogs, dynamic characteristics attached directly or indirectly to catalogs of a given class, and labels, options, and input patterns of those dynamic characteristics.

Omitting -class exports all class definitions of the entire data model.

- **-db_type *type***

When using the -systemtables switch, use -db_type to indicate the EDM Library Services licensing type. Acceptable values for type are:

- **XDM25** — EDM Library Services 25
- **XDM50** — EDM Library Services 50
- **XDM300** — EDM Library Services 200/300
- **EDM** —Integrated EDM Library Services

Specifying EDM as a type consumes an xedmdeveloper license if you use the -class switch to update the Class class.

- **-utf8**

An optional switch that you can use with -restore that specifies you want to use UTF-8 encoding when writing the exported init file to the file system. If you do not specify the -utf8 switch, the **batchadmin** program uses the encoding setting specified by the system.

When loading, the **batchadmin** program expects init files to already be in be in UTF-8 format (UTF-8 format is the default and only permissible encoding). Therefore, the -utf8 switch is not needed.

- **-debug**

An optional switch that turns on verbose diagnostic output.

Examples

The following syntax uses the **batchadmin** program to update the EDM Library system tables:

```
batchadmin -xdm_data_dir data_directory_path -systemtables -db_type type  
[-debug]
```

The following syntax uses the **batchadmin** program to reload the default label text into the database:

```
batchadmin -xdm_data_dir data_directory_path -load -labels [-debug]
```

The following syntax uses the **batchadmin** program to regenerate the database indexes.

```
batchadmin -xdm_data_dir data_directory_path -indexes [-debug]
```

The following syntax uses the **batchadmin** program to load an init file into the database:

```
batchadmin -xdm_data_dir data_directory_path -load -initfile filename  
[-debug]
```

The following syntax uses the **batchadmin** program to export an init file containing the entire EDM Library data model from the database:

```
batchadmin -xdm_data_dir data_directory_path -restore -initfile filename  
[-utf8] [-debug]
```

The following syntax uses the **batchadmin** program to export an init file containing all characteristics and their option list values belonging to a specific object class:

```
batchadmin -xdm_data_dir data_directory_path -restore -initfile filename  
-class class_number  
[-utf8] [-debug]
```

The following syntax uses the **batchadmin** program to export an init file from the database, but only classes, characteristics (and their option list values) that have a specific init filename in their “from Init File” input field. For example, you might want to export all classes and characteristics belonging to the CompClassInit file, which means they have a from_Init_File value equal to CompClassInit.

```
batchadmin -xdm_data_dir data_directory_path -restore -initfile filename  
-initfilekey from_Init_File [-utf8] [-debug]
```

The following syntax uses the **batchadmin** program to migrate class number 110 (Documents) and all its objects to major-/minor versioning:

```
batchadmin -xdm_data_dir data_directory_path -migrate -class 110 [-debug]
```

The following syntax uses the **batchadmin** program to change the status value to “U” for list column characteristics with no parent list frame, thereby removing them from display:

```
batchadmin -xdm_data_dir data_directory_path -cleanup [-debug]
```

Related Topics

[Command Line Syntax Conventions](#)

data_model_checker

To access: In an EBS command window, type **data_model_checker**

Use the **data_model_checker** command to run tests that check for proper data model construction, required elements, and conflicting value settings. Because it is an administrative program, **data_model_checker** is only available for invocation from a server software tree.

Usage

```
data_model_checker -login_config config_name [-run check1, check2, ...  
checkN] | [-skip check1, check2, ... checkN] [-out_file log_file_name]
```

Arguments

- **-login_config config_name**

The name of a login configuration. For information about how to create a login configuration, refer to “Automatic Login Configurations” in the *Xpedition EDM Library Overview*.

- **-run check1, check2, ..., checkN**

An optional switch that specifies specific checks to run. If you omit the -run or -skip switch, then the command executes all checks. Follow the -run switch with a list of check keywords separated by commas (refer to [Table 24](#) on page 370). For example:

```
-run catalog, class, reference
```

- **-skip check1, check2, ..., checkN**

An optional switch that specifies specific checks to skip. If you omit the -skip or -run switch, then the command executes all checks. Follow the -skip switch with a list of check keywords separated by commas (refer to [Table 24](#) on page 370). For example:

```
-skip catalog, class, reference
```

- **-out_file log_file_name**

An optional switch that specifies the path and name of a log file. If you omit the -out_file switch, the command writes a *data_model_check_<date>.log* file in the current directory.

Description

The **data_model_checker** command runs checks on 19 areas of the data model. You can run all checks, or use keywords ([Table 24](#)) following the -run or -skip switch to run only a specific check on all qualifying objects. For example, the access check tests all characteristics requiring an access path, the catalog check tests all catalogs, and so on. You cannot run a check on a subset of objects.

To determine the database with which to connect, use the -login_config switch followed by the name of a login configuration. Typing the command without arguments returns command help.

Include the -run or -skip switches to specify specific checks to include or exclude. Omitting the -run or -skip switches runs all checks.

By default, the command writes errors to a *data_model_check_<date>.log* file in the current directory. To specify a different name or location for the log file, use the *-out_file* switch.

Table 24. Data Model Checks

| Check Keyword | Description |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| access | <p>Check all Characteristic objects requiring an access path for these Access Path value problems:</p> <ul style="list-style-type: none">• Missing access path.• An access path that uses non-existing characteristics.• An access path with characteristics that are multiclass references.• An access path formed with incorrect syntax.• Characteristics in an access path that also need an access path.• A main key characteristic with an access path different than the characteristic name.• A top level characteristic with an access path that has references that do not have the Search Characteristic status bit set.• An access path that contains a “pseudo” reference characteristic (that is, a characteristic with different Class and Ref. Class values, but that does not have the Reference Characteristic status bit set).• A top level characteristic with a reference that would not return a value because the Outer Join status bit is not set.• Columns of lists that have the Input Characteristic bit set with an access path containing nullable reference without the Outer Join bit set. |
| catalog | <p>Check all Catalog Group objects for these problems:</p> <ul style="list-style-type: none">• A catalog group without a title.• A catalog group without a parent name.• A catalog group with non-existing parents. |
| catalog_dname | <p>Check the Domain Model Name of all Catalog Group objects for these problems:</p> <ul style="list-style-type: none">• A catalog group with no Domain Model Name value.• Two or more catalog groups within the parent catalog group with the same Domain Model Name value. <p>The domain_model_checker command can also run this check.</p> |
| charact_dname | <p>Check the Domain Model Name of all Characteristic objects for these problems:</p> <ul style="list-style-type: none">• A characteristic with no Domain Model Name value.• Two or more characteristics within a class, list, or sublist that have the same Domain Model Name value. <p>The domain_model_checker command can also run this check.</p> |

Table 24. Data Model Checks (continued)

| Check Keyword | Description |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| charact_lang | Check for any characteristics that have a language setting but do not have an Information Text label or a Search Text label for that language. |
| class | Check class objects and report these problems: <ul style="list-style-type: none"> • A class object without a Class Name value. • A menu group with several different values for Group Order Number. • Classes with identical sequence numbers inside the same menu group. |
| class_dname | Check the Domain Model Name of all class objects for these problems: <ul style="list-style-type: none"> • A class object with no Domain Model Name value. • Two or more class objects with the same Domain Model Name value. The domain_model_checker command can also run this check. |
| consistency | Check for these reading, writing, and searching consistency problems: <ul style="list-style-type: none"> • Writable ("Input Characteristic") static top level characteristics for which a writable Main Key characteristic does not exist. • Static top level characteristics for which a key characteristic does not exist (for object reading). • Writable ("Input Characteristic") dynamic top level characteristics for which a writable static Main Key characteristic does not exist. • Dynamic top level characteristics for which a static key characteristic does not exist (for object reading). • List column characteristics is stored in different table than the list frame. • Main Key characteristics has a nonempty access path different than the characteristic ID. • Input Characteristic characteristics having nonempty access path different than the characteristic ID. • List column characteristics with a different table than the list frame and having an empty access path (for object reading). • Characteristics with incorrect access path syntax. • List column characteristics for which an access path element has not been found (for object reading). • Characteristics with incorrect access path and the key characteristic is missing for some path element. • Access path elements needing access path themselves. • Characteristics with incorrect access path and an inconsistent table or column. • List column characteristics for which the key characteristic of the first access path element has different table than the list table. |

Table 24. Data Model Checks (continued)

| Check Keyword | Description |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | <ul style="list-style-type: none"> • Static top level characteristics without access path for which a key characteristic does not exist. • Static list columns without access path. • Dynamic top level characteristics without access path for which key characteristic does not exist. • Dynamic list columns without access path. • Static characteristics for which an access path element has not been found (for object searching). • Static characteristics whose dynamic path element is not attached to all catalogs (for object searching). • Dynamic characteristics for which access path element has not been found (for object searching). • Dynamic characteristics whose dynamic path element is not attached to all catalogs (for object searching). |
| default_value_type | <p>Check Characteristic objects for these default value problems:</p> <ul style="list-style-type: none"> • An incorrect value in the Default Value field. • Characteristics of a type that cannot be configured with a unit. • Characteristics with a value type (such as BLOB or date) that have a value in the Default Value field, but should not. |
| dname_old | <p>Check the Domain Model Name of class and Catalog Group objects for these problems:</p> <ul style="list-style-type: none"> • A class object with no Domain Model Name value. • A Catalog Group object with no Domain Model Name value. • Class objects and Catalog Group objects with the same Domain Model Name value. <p>The charact_dname, catalog_dname, and class_dname checks supersede the dname_old check. The data_model_checker command only runs the dname_old check if explicitly specified by the -run switch. The domain_model_checker command can also run this check.</p> |
| dynamic | <p>Check dynamic characteristics for these problems:</p> <ul style="list-style-type: none"> • List column characteristics or a sublist are attached to a catalog.

Only simple characteristics or a list frame characteristic are permitted to be attached to a catalog. In a proper configuration, list columns and sublists are attached to the catalog through the parent list, not directly. • Dynamic characteristics not in a list, and not attached to any catalog (a dynamic characteristic should always be attached to at least one catalog). |
| length_rec | <p>Check for any characteristics that have a Value Length setting that is too short to hold the characteristic value (recursive value length check).</p> |

Table 24. Data Model Checks (continued)

| Check Keyword | Description |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| list | <p>Check all list characteristics for these problems:</p> <ul style="list-style-type: none"> • List frames with no line keys. • List frames with multiple line keys (and having sublists). • List frames too short to hold the obj_id or the parent line key. • List frames (dynamic) too short to hold the obj_id or the parent line key. • List frames without the Main Key status bit set. • Sublist frames with the Main Key status bit set. • List elements with an invalid parent list number. • Lists with different class and reference class values. • Multiple lists having the same list number for a given class. • Sublists and parent lists that are not properly nested. • Sublists that do not have a parent list. • List column characteristics without a list frame. |
| main_key | <p>Check the length of the obj_id characteristic and compare that against any characteristic that has the Main Key status bit set. If any main key characteristic is shorter than the unique identifier length, report a problem.</p> |
| multi_ref | <p>Check characteristics that have the Multiclass Object Reference status bit set for these problems,:</p> <ul style="list-style-type: none"> • A missing class characteristic name for Default Value. • A class number in the Class field that does not exist. • No class number in the class characteristic option list. • A class number in the class characteristic option list that does not exist. • A non-integer value in the class characteristic option list. • A characteristic with a Value Length too short to hold the referenced obj_id. |
| option_list | <p>Check option lists for these problems:</p> <ul style="list-style-type: none"> • Option values that are too long for input characteristics. • Option values that are too long for non-input Characteristics. |
| reference | <p>Check references for these problems:</p> |

Table 24. Data Model Checks (continued)

| Check Keyword | Description |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | <ul style="list-style-type: none"> A characteristic with a Value Length too short to hold the referenced obj_id. A characteristic containing the number of a class in the Ref Class field that does not exist. A characteristic with no value in the Ref Class field. |
| take_over | <p>Check catalog group parents and subcatalog groups for these problems:</p> <ul style="list-style-type: none"> A characteristic whose Value Length is too short to hold the take over value. A characteristic trying to take a value from a characteristic that does not exist. A characteristic with an incorrect format for Default Value. |
| take_over_type | <p>Check catalog group parents and subcatalog groups for these problems:</p> <ul style="list-style-type: none"> A characteristic taking a value from a characteristic of an incompatible type. A characteristic taking a value from a characteristic of an inconsistent type. A characteristic taking a value from a non-existing characteristic. |
| take_over_unit | <p>Check catalog group parents and subcatalog groups for these problems:</p> <ul style="list-style-type: none"> A characteristic with a unit taking a value from a characteristic with an incompatible unit. A characteristic with no unit taking a value from a characteristic with a unit. A characteristic with a unit taking a value from a non-existing characteristic. |
| user | <p>Check characteristics that inherit the user name from the user who is logged in (for example, “created by” and “modified by” characteristics) and report any Set User Name, Checkout User Name (co_user), or Locking User Name (obj_user) characteristics with a length too short to hold the longest user name in the user object class.</p> |
| value_type | <p>Check that characteristics with the same table name and table column have the same value type. If the value types are different, report a problem</p> |

Examples

```
data_model_checker -login_config phils_db
-run access, charact_dname, charact_lang -outfile C:/logs/data_check_2018_0
7_09.txt
```

This example runs the **data_model_checker** command using the phils_db login configuration to connect with the database. The command runs only the access, charact_dname, and charact_lang checks and saves log information to the *C:/logs/data_check_2018_07_09.txt* file.

Related Topics

[Command Line Syntax Conventions](#)

domain_model_checker

To access: In an EBS command window, type **domain_model_checker**

Use the **domain_model_checker** command to run only the checks that test classes, catalogs, and characteristics for correct domain model naming. Because it is an administrative program, **domain_model_checker** is only available for invocation from a server software tree.

Usage

```
domain_model_checker -login_config login_config [-run check1, check2, ...  
checkN ]|[-skip check1, check2, ... checkN][-out_file log_file_name]
```

Arguments



Note:

The arguments are the same ones used by the [data_model_checker](#) command.

Description

domain_model_checker is similar to [data_model_checker](#), but can only run the class_dname, catalog_dname, charact_dname, and dname_old checks (refer to [Table 24](#) on page 370).

The checks test all class objects, catalog objects, or Characteristic objects. You cannot run a check on a subset of objects.

Examples

```
domain_model_checker -login_config my_login  
-run charact_dname -outfile C:/logs/domain_check_2018_07_09.txt
```

This example runs the **domain_model_checker** command using the `my_conn` login configuration to connect with the database. The command runs only the `charact_dname` check and saves log information to the `C:/logs/domain_check_2018_07_09.txt` file.

Related Topics

[Command Line Syntax Conventions](#)

Appendix A ascl2dms Import Manager

The **ascl2dms** data import manager program uses a configuration text (ASCII) file and a data text (ASCII) file to create a catalog structure and objects in an EDM Library database. The program can only load the data from the source file into a database, and has no export capability.

[ascl2dms Import Manager Overview](#)
[ascl2dms Input Files](#)
[Configuration File Structure](#)
[Starting the ascl2dms Program](#)
[ASCII Loader Import Wizard Screens](#)
[ascl2dms Command Line Syntax](#)

ascl2dms Import Manager Overview

You can run the **ascl2dms** loader program with a graphical interface by typing the command name into an EBS command window, or in batch mode by following the command name with the -batch switch.

The graphical user interface displays a series of screens you can use to enter the appropriate parameters, usually from a list of predefined choices. After completing all screens, the program loads the data and creates the objects in the database.

You can save your screen entries into an XML session transcript file, which is useful if you have to run **ascl2dms** multiple times with minimal variations. Rerunning the loader with a session transcript lets you step through the screens and change only the parameters that vary. Alternately, you can edit the transcript file before launching the program so you do not have to make changes in the interface at all.

Batch mode does not prompt for input (you set parameters using command switches). Batch mode also lets you specify a session transcript file as input, but you must make all changes in the file before using it.

ascl2dms Input Files

The **ascl2dms** program requires paths to a configuration file and a load file. Both files are in ASCII format, which you can use a standard text editor to edit.

With the graphical user interface, you specify the file paths in the [ASCII Loader Import - Settings Screen](#). In batch mode, use the -cfgfile and -datafile switches.

The configuration file defines how to read the data contained in the load file and import it into the database. All columns of the configuration file must contain values or an error can occur during the import.



Note:

A load file must not contain comment lines (that is, lines beginning with a pound sign) or problems during the import can occur.

The load file contains the data to load, in text form, formatted in a row/column structure and separated by special column separator characters defined by the configuration file. You cannot use the loader to import nested table data (that is, characteristic data that resides in a table within a table).

When loading data, make sure that you have defined and provided values for any mandatory characteristics required by that object class (for example, the domain model name when creating catalog objects). If mandatory characteristics do not have values, the import manager cannot create and save those objects in a database. If you use the program to create dynamic characteristics, make sure you use names that are unique for each catalog group (for example, do not use "Value" for both resistance and capacitance).

Related Topics

[Configuration File Structure](#)

Configuration File Structure

The configuration and data files are text files. You can assign any name and extension to these files. A common, but not required, convention is to use `<name>.cfg` for the configuration file and `<name>.dat` or `<name>.tab` for the data file.

Example Configuration and Data Files

Scanners and Tokens

.CONFIG_GLOBAL Section

.CONFIG_CONVERT Section

Example Configuration and Data Files

A configuration file must contain keywords that define the beginning and end of the configuration, and the beginning and end of the global and convert sections.

A configuration file begins with the `.SECTION_START CONFIG_SCAN` keyword and ends with the `.CONFIG_END` and `.SECTION_END` keywords. In between the beginning and ending keywords are two required sections:

- A global section that begins with the `.CONFIG_GLOBAL` keyword.
- A convert section that begins with the `.CONFIG_CONVERT` keyword.

The following shows an example configuration file. The file provides definitions to the **ascl2dms** program when you load a data file with capacitor components, a data file with resistor components, or a data file with components that reference manufacturing parts. Refer to this example when reading subsequent topics that describe scanners, tokens, and sections within the configuration file.

```
.SECTION_START CONFIG_SCAN

# Global parameter definitions (required):
# -----
# DefLineEnable [Y] - Specifies if a header definition exists (must be Y)
# DefLineMarker [@] - The character that identifies the header definition
line (@)
# ColSeparator [\t] - The column separator (\t is the tab character)
# SearchEnable [N] - Look for the Data in the config file (Y/N)
# SectionStart [.secstart] - Section that begins data if SearchEnable=Y
# SectionEnd [.sectend] - Section that ends data if SearchEnable=Y
#
.CONFIG_GLOBAL
@ScannerName + DefLineEnable + DefLineMarker + ColSeparator +
SearchEnable + SectionStart + SectionEnd
OBJECT_SCAN   + Y             + @           + \t           + N
               + .secstart     + .sectend

# Token name and characteristic assignment definitions- (required):
# -----
# TokenName - Header in the data file
```

ascl2dms Import Manager

Example Configuration and Data Files

```

# CharactName - EDM Library characteristic key (001obj_id)
# TokenType Definitions:
#   8 - Mandatory object key (only one data column can have the key)
#   1 - Static characteristics - properties that exist on all objects
#         (for example, Description)
#   0 - Dynamic characteristic - properties that exist on objects only in
#       specific catalogs (for example, Resistance)
#   2 - The catalog identifier in the database (for example, AAABCBA)
#   3 - The catalog abbreviation in the database. Abbreviations usually
#       are unique (for example, FixedResistor).
#   101 - Ignore the mapping relationship - this is default behavior if
#         the header name is not found
#
.CONFIG_CONVERT
@ScannerName + TokenName + CharactName + TokenType +
OBJECT_SCAN + Part Number + 001obj_id + 8 +
OBJECT_SCAN + Catalog + + 3 +
OBJECT_SCAN + Status + 001obj_statu + 1 +
OBJECT_SCAN + Description + 001komp_name + 1

#Mapping for the Capacitor Catalog
OBJECT_SCAN + Capacitance + capacitor_value + 1 +
OBJECT_SCAN + Voltage (Cap) + capacitor_voltage + 1 +
OBJECT_SCAN + Tolerance (Cap) + capacitor_tolerance + 1

# Mapping for the Resistor Catalog
OBJECT_SCAN + Resistance + resistors_value + 1 +
OBJECT_SCAN + Wattage + resistors_wattage + 1 +
OBJECT_SCAN + Tolerance (Res) + resistor_tolerance + 1

# Manufacturing Source List
OBJECT_SCAN + MFGPN KEY + 001a26hertyp + 1 +
OBJECT_SCAN + Package Weight + 001packageweight + 1 +
OBJECT_SCAN + Databook + 001lst_kat + 1 +
OBJECT_SCAN + P-Price + 001lst_prs + 1 +
OBJECT_SCAN + Component Source Status + 001lst_statu + 1 +
OBJECT_SCAN + Packing Unit + 001lst_vpe + 1 +
OBJECT_SCAN + Manuf. ID + 001lstal0her + 1 +
OBJECT_SCAN + Supplier ID + 001lstal0sup + 1 +
OBJECT_SCAN + BoS Declaration + 001rohsobj_id + 1

.CONFIG_END
.SECTION-END

```

The following shows the content of a data file with 10 capacitor components to load. Per the configuration file, an @ symbol defines the header line in the load file, and a tab is the column separator.

| @ID | Catalog | Tolerance (Cap) | Capacitance | Voltage (Cap) | Part Number |
|-----|-----------|-------------------------|-------------|---------------|-------------|
| | Status | Description | | | |
| 1 | Capacitor | 0.05 | 18pF | 50V | 2100-0001 |
| D | | CAP CER 18PF 50V C0G 5% | 0402 | | |
| 2 | Capacitor | 0.05 | 30pF | 50V | 2100-0002 |
| D | | CAP CER 30PF 50V 5% | NP0 0402 | | |

| | | | | | |
|----|----------------|--------------------------|---------|------|-----------|
| 3 | Capacitor | 0.05 | 47pF | 50V | 2100-0003 |
| D | CAP CER | 47PF 50V 5% NPO 0402 | | | |
| 4 | Capacitor | 0.1 | 470pF | 50V | 2100-0004 |
| D | CAP CER | 470PF 50V 10% X7R 0402 | | | |
| 5 | Capacitor | | 0.001uF | 16V | 2100-0005 |
| D | CAP CER | 0.001UF 16V X7R 0402 | | | |
| 6 | Capacitor | 0.1 | 0.022uF | 10V | 2100-0006 |
| D | CAP CER | 0.022UF 10V 10% X7R 0402 | | | |
| 7 | Capacitor | 0.1 | 0.1uF | 6.3V | 2100-0007 |
| D | CAP CER | 0.1UF 6.3V 10% X5R 0402 | | | |
| 8 | Capacitor | 0.1 | 1uF | 6.3V | 2100-0008 |
| D | CAP CER | 1UF 6.3V 10% X5R 0402 | | | |
| 9 | Capacitor | | 10uF | 10V | 2100-0010 |
| D | CAP CER | 10UF 10V Y5V 0805 | | | |
| 10 | Capacitor | 0.1 | 10uF | 16V | 2100-0013 |
| D | CAP, CER, SMD, | 7343, 16V, 10UF, 10% | | | |

Related Topics

- [Scanners and Tokens](#)
- [.CONFIG_GLOBAL Section](#)
- [.CONFIG_CONVERT Section](#)

Scanners and Tokens

Scanners and tokens within the configuration file tell the **ascl2dms** program about the type of data to import and the mapping between the load file (source) and the database (target).

Scanners

In both sections of the configuration file, the first column header is @ScannerName. The scanner value identifies the type of data import. The only acceptable value in the @ScannerName column is OBJECT_SCAN, which tells the **ascl2dms** program that the import is for objects and their data content.

Tokens

A token defines the connection between the ASCII file and the EDM Library data structure. The TokenName and TokenType column values assign the column content in the load file to the EDM Library data structure.

- TokenName creates a characteristic name in the configuration file that matches the column name in the load file.
- TokenType defines the type of token (such as the object key characteristic, static or catalog-dependent characteristic, catalog group abbreviation, or the catalog identifier).
[.CONFIG_CONVERT Section](#) on page 383 describes the token types for the OBJECT_SCAN scanner.

Related Topics

[Example Configuration and Data Files](#)
[.CONFIG_GLOBAL Section](#)
[.CONFIG_CONVERT Section](#)

.CONFIG_GLOBAL Section

The **ascl2dms** program requires that the first section of the configuration file start with the .CONFIG_GLOBAL keyword and define global loader parameters.

The following shows an example .CONFIG_GLOBAL section of a configuration file:

```
.CONFIG_GLOBAL
@ScannerName      + DefLineEnable    + DefLineMarker    + ColSeparator    +
SearchEnable      + SectionStart     + SectionEnd      +
OBJECT_SCAN       + Y              + @              + \t             + N
                  + .secstart       + .sectend
```

The first line after the .CONFIG_GLOBAL keyword is the definition line and contains the following column names:

- **ScannerName** — The name of the scanner object. The only permitted scanner object name is OBJECT_SCAN.
- **DefLineEnable** — Identifies whether the loader should evaluate the definition line of the load file. The value must be Y.

The .CONFIG_CONVERT section then defines the token types. Token names are taken from the line of the load file that is marked by the DefLineMarker symbol (usually @).

- **DefLineMarker** — The character that marks the definition line (usually @) in the configuration and load files. Any character except a pound sign (#), which marks a comment, is permitted.
- **ColSeparator** — The character used by the parser to recognize a new column. Can be any character except for the ones shown in the table.

Table 25. ColSeparator Characters Not Allowed

| Character | Definition |
|-----------|------------------------------|
| # | Pound sign |
| \ | Forward slash |
| { } | Left or right curly braces |
| < > | Left or right angle brackets |
| - | Hyphen |

| Character | Definition |
|-----------|-------------|
| = | Equals sign |
| > | Caret |
| \$ | Dollar sign |

If a tab character is the column separator, use \t to denote the tab character in the configuration file, but use a plain tab in the data file. A tab character setting displays a comma to show column header separation in the [ASCII Loader Import - Select Catalog Files Screen](#), but this does not affect the data import.

- **SearchEnable** — Defines whether the scanner should find the import data inside the configuration file:
 - **Y** — The configuration file contains the import data.
 - **N** — The load file contains the import data (the default).
- **SectionStart** — A key word to mark the beginning of the import data area in the configuration file (only works if you specify SearchEnable=Y). Can be any string without a space or pound sign (#).
- **SectionEnd** — The end of the data being integrated into the configuration file (SearchEnable=Y). Can be any string without a space or a pound sign (#).

Related Topics

[Example Configuration and Data Files](#)

[Scanners and Tokens](#)

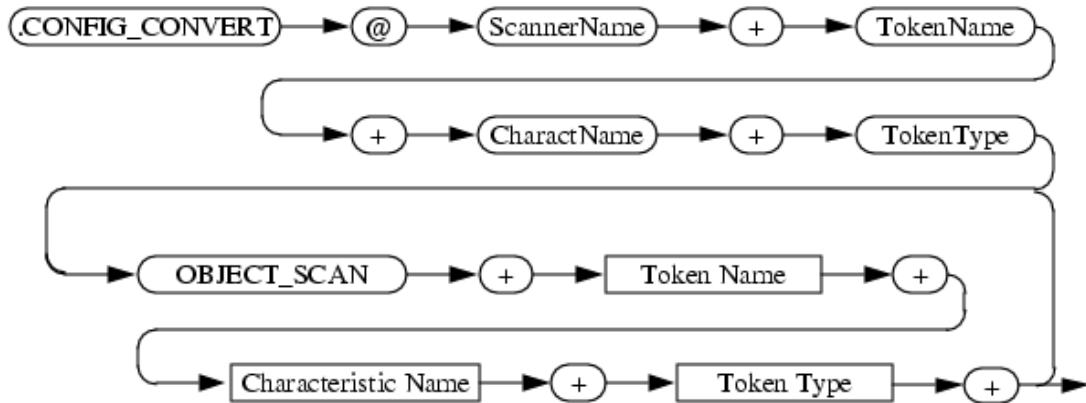
[.CONFIG_CONVERT Section](#)

.CONFIG_CONVERT Section

The **ascl2dms** program requires that the second section of the configuration file start with the .CONFIG_CONVERT keyword and specify how the loader assigns the token names from the definition line to EDM Library characteristics.

The .CONFIG_CONVERT section follows the format shown in [Figure 179](#).

Figure 179. Structure Syntax of the .CONFIG_CONVERT Section



The following shows an example .CONF_CONVERT section.

```
.CONFIG_CONVERT
@ScannerName + TokenName + CharactName + TokenType +
OBJECT_SCAN + Part Number + 001obj_id + 8 +
OBJECT_SCAN + Catalog + + 3 +
OBJECT_SCAN + Status + 001obj_statu + 1 +
OBJECT_SCAN + Description + 001komp_name + 1
```

The columns of the .CONFIG_CONVERT section are:



Note:

You can specify more columns than necessary in the .CONFIG-CONVERT section. The columns do not have to be in the load file. You can use a single configuration file for several load files.

- **ScannerName** — The name of the scanner object. The only permitted scanner object name is OBJECT_SCAN.
- **TokenName** — The names of the tokens in the definition line of the load file. Specify a string without the pound sign (#).
- **CharactName** — The database characteristic that corresponds to the TokenName value in that row.
- **TokenType** — The kind of token. Valid numeric TokenType values are:
 - 0 — A dynamic characteristic. The CharactName column contains the name of the dynamic characteristic.
 - 1 — A static, or standard, characteristic. The CharactName column contains the name of the standard characteristic.
 - 2 — A catalog identifier such as AAAABB where a loaded object is written.

- 3 — The abbreviation of the catalog where a loaded object is written.
- 8 — The mandatory unique identifier of the object. Object identifier characteristics usually have a name in the format <class_no>obj_id.
- 101 — The token is an entry without a corresponding characteristic, which causes the loader to skip the information in the data column when loading.

Related Topics

[Example Configuration and Data Files](#)

[Scanners and Tokens](#)

[.CONFIG_GLOBAL Section](#)

Starting the ascl2dms Program

You can start the **ascl2dms** program with a graphical user interface, or in batch mode. Batch mode enables you to specify all arguments on the command line by using switches and flags.

Prerequisites

- Your login account has the following rights to access database objects:
 - **Importing an object** — You need the right to add new objects in the target class. For example, if importing Component objects, you must have rights to create a new Component object.
 - **Catalog group creation** — Sometimes new catalog groups are needed to categorize the imported data. When this is a requirement, you need rights to create new catalog groups for the object class in which you are performing the import.
Importing data into a fixed catalog structure requires using an EDM Library account with catalog creation rights.
 - **Characteristic creation** — Importing objects can also require the ability to create new characteristics to store data from source properties.

An error message can occur if you do not have sufficient rights within the database to perform a specific task. If you find your login account does not grant sufficient rights, consult your administrator.

Procedure

This procedure describes how to invoke the **ascl2dms** program with either the graphical user interface or in batch mode.

- **To invoke the ascl2dms program with the wizard-like GUI** — Type only the program name in an EBS command window.

ascl2dms

You can also use the **Tools > Import/Export > Import Manager** pulldown menu in Xpedition EDM Library Cockpit.

- **To launch the ascl2dms program in batch mode** — Follow the name of the program with the -batch flag and any additional arguments (refer to “[ascl2dms Command Line Syntax](#)” on page 402). To see the full list of command arguments, use the -help flag.

After you execute the command line, the data import program then runs without further interaction.

Related Topics

[ASCII Loader Import Wizard Screens](#)

[ascl2dms Command Line Syntax](#)

ASCII Loader Import Wizard Screens

This section describes the ASCII loader import wizard screens in the order in which they display.

- [ASCII Loader Import - Introduction Screen](#)
- [ASCII Loader Import - Select Session Transcript File Screen](#)
- [ASCII Loader Import - Settings Screen](#)
- [ASCII Loader Import - Select Catalog Files Screen](#)
- [ASCII Loader Import - Select Key Property Screen](#)
- [ASCII Loader Import - Create Catalog Groups Screen](#)
- [ASCII Loader Import - Map Characteristics Screen](#)
- [ASCII Loader Import - Verify Source Data Screen](#)
- [ASCII Loader Import - Import Settings Screen](#)
- [ASCII Loader Import - Save Session Transcript File Screen](#)
- [ASCII Loader Import - Creating Catalog Groups and Characteristics Screen](#)
- [ASCII Loader Import - Importing Data Screen](#)
- [ASCII Loader Import - Final Summary Screen](#)

ASCII Loader Import - Introduction Screen

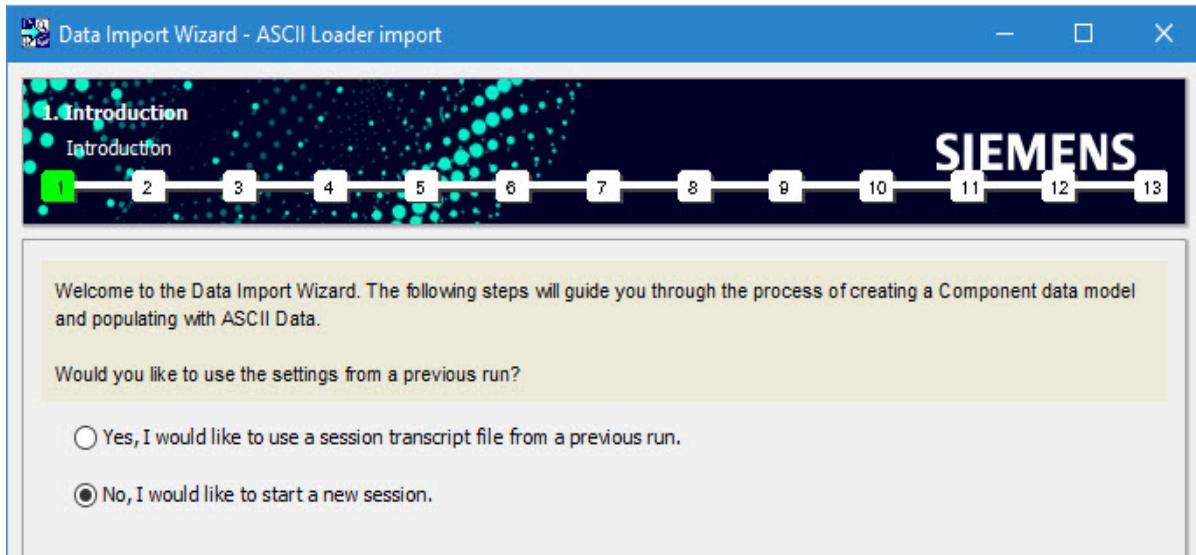
To access: Refer to “[Starting the ascl2dms Program](#)” on page 385

The Introduction screen displays after you invoke the **ascl2dms** program with the graphical user interface.

Previous screen: None

Next screen: [ASCII Loader Import - Select Session Transcript File Screen](#) or [ASCII Loader Import - Settings Screen](#) (depending on whether choosing to use a session transcript file)

Figure 180. .ASCII Loader Import Introduction (Screen 1)



Usage Notes

From the introduction screen, you must select one of the following:

- **Yes** — Lets you navigate to a session transcript from a previous run of the ASCII Loader import wizard, which pre-fills subsequent screens with the values from that session. You then only need to step through the screens to change values as appropriate.

Clicking **Next** asks for your login information, then displays a screen with which to navigate to the location of the transcript file (step 2 on the progress indicator). Giving the location of a transcript file and clicking **Next** then displays the [ASCII Loader Import - Settings Screen](#).

- **No** — Does not use values from a session transcript file, which means you must supply all fields in subsequent screens with new information and choices.

Clicking **Next** asks for your login information, and then goes directly to the [ASCII Loader Import - Settings Screen](#).

When prompted, complete the EDM Library login dialog box to connect to an EDM Server. If you are unsure of how to enter login information, refer to “Invoking EDM Library Cockpit” in the *Xpedition EDM Library Overview*.

To avoid problems later when the wizard actually starts importing data, you must log in using an account that has rights to create Catalog objects, Characteristic objects, and new data objects in the target object class.

ASCII Loader Import - Select Session Transcript File Screen

To access: Refer to “[Starting the ascl2dms Program](#)” on page 385

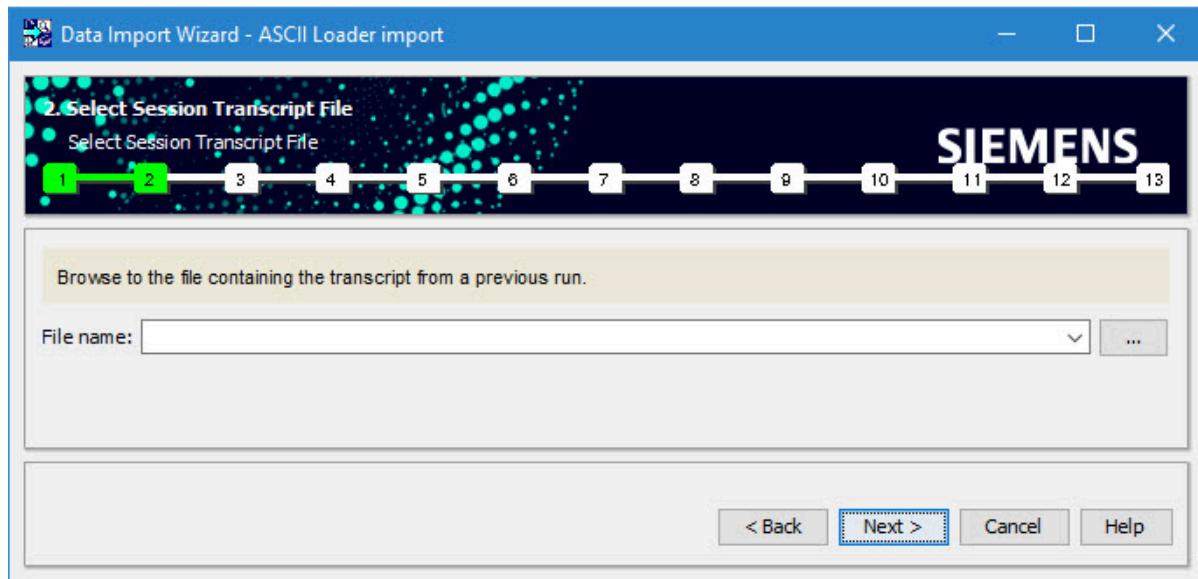
The ASCII Loader Import Select Session Transcript File screen provides a field with which to navigate to the location of a transcript file saved from a previous session of the ASCII import loader.

The screen only displays if you selected “Yes, I would like to use a session transcript file from a previous run” in the [ASCII Loader Import - Introduction Screen](#).

Previous screen: [ASCII Loader Import - Introduction Screen](#)

Next screen: [ASCII Loader Import - Settings Screen](#)

Figure 181. ASCII Loader Import Select Session Transcript File (Screen 2)



Usage Notes

A load file should not contain comment lines (that is, lines beginning with a pound sign) as this can cause problems during the import.

ASCII Loader Import - Settings Screen

To access: Refer to [“Starting the ascl2dms Program”](#) on page 385

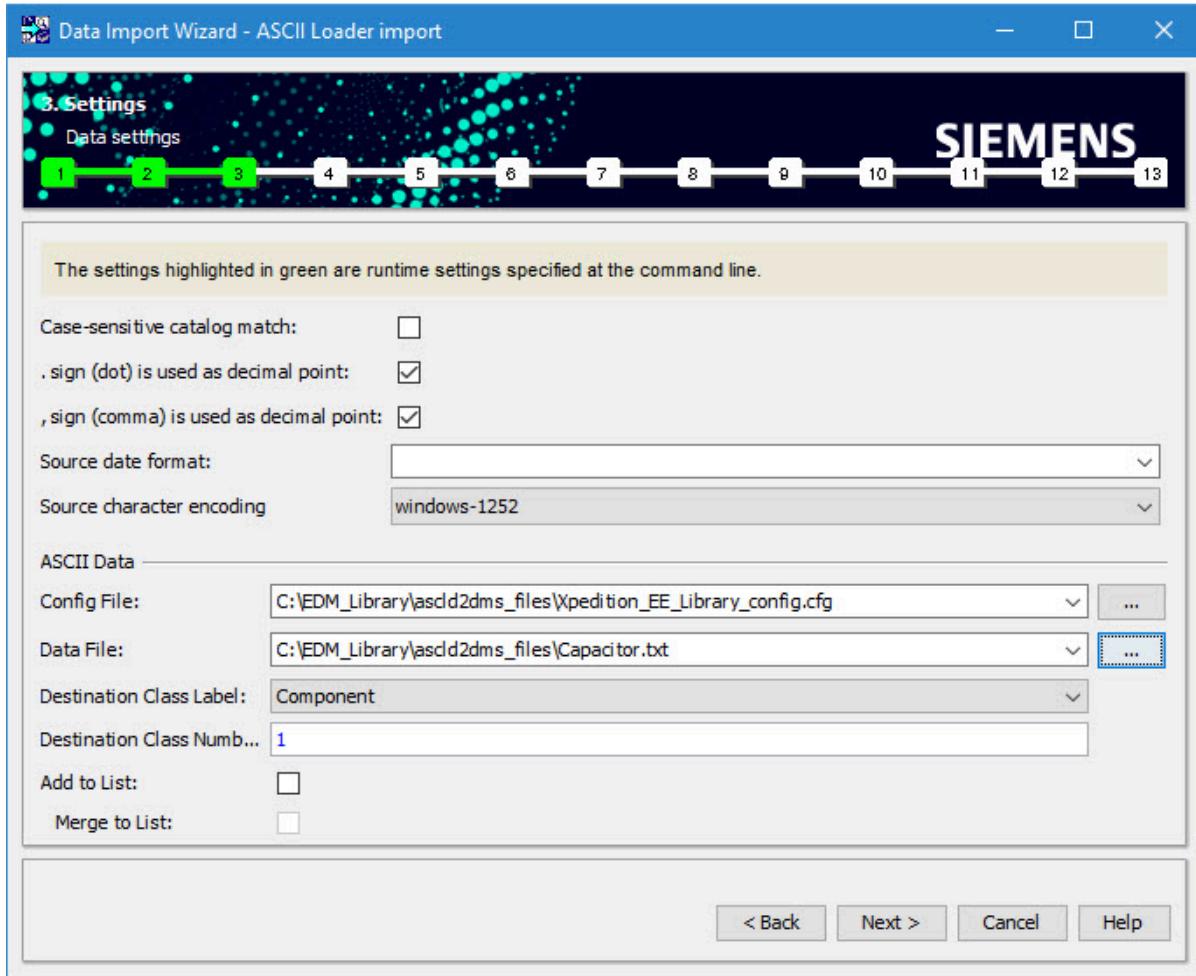
The ASCII Loader Import Settings screen gathers the parameters necessary to interpret the ASCII data correctly when performing the import.

As shown in [Figure 182](#), the import settings screen is divided into two halves. The top half specifies generic settings to use when determining how to recognize data when importing it into the database. The bottom half contains settings unique to the ASCII import wizard.

Previous screen: [ASCII Loader Import - Introduction Screen](#)

Next screen: [ASCII Loader Import - Select Catalog Files Screen](#)

Figure 182. ASCII Loader Import Settings (Screen 3)



Usage Info

Complete the Settings screen as follows:

- Check the “Case-sensitive catalog match” checkbox to turn on case-sensitive recognition when reading the ASCII configuration data that specifies the EDM Library catalogs in which to import data. If not checked, do not use case-sensitivity when matching the EDM Library catalog group name with the catalog information from the ASCII configuration file.
- Check the next two check boxes to recognize a period and a comma as a numeric decimal point instead of as a string character for any values located within the ASCII data file. You must check one or both boxes to be able to store a value from the data file containing a period or comma as double data types in EDM Library, provided that you have also specified that you want that property stored as a double data type later in the [ASCII Loader Import - Map Characteristics Screen](#).
- Optionally, enter a format used to recognize date properties in the source data. When importing, the import wizard then stores any value that matches this pattern as a date data type in EDM

Library, provided that you have also specified that you want values stored as a date data type later in the [ASCII Loader Import - Map Characteristics Screen](#).

If you receive an error related to the date format later on the Verify Source Data screen, you may need to use a Java expression in the Source date format field instead of relying on the Default Value specified in the characteristic definition. For example, when loading a date value such as 10-Oct-2017 12:12, you would not want to use the default dd-mon-yyyy hh24:mi format. Instead, enter a Java expression of dd-MM-yyyy HH:mm into the Source date format field, which matches date format of the input data. For more information, refer to “[Data and Time Patterns](#)” in the Oracle Java API documentation.

If you do not specify a date pattern (that is, the field is empty), the import wizard stores the data as a string data type in EDM Library.

- Choose a source character encoding value appropriate for your location from the list of dropdown choices. The default value is ISO-8859-1. The values listed are those supported by the Java Runtime environment on the client system.
- Enter a path to an ASCII configuration file that specifies how to interpret the data contained in the data file (also called the ASCII catalog file). The ASCII configuration file must adhere to the construction rules described in [“Configuration File Structure”](#) on page 379.
- Enter a path to ASCII catalog data file containing the ASCII data to import.
- Specify the class in which to import the data. Selecting a class label from the dropdown list of available classes automatically fills in the corresponding class number.
- Check the Add to List check box to add new lines in a list frame for imported column characteristics if they are directed to the same object, without deleting existing lines. If not checked (the default), replace existing lines in a list with the new value.

If Add to List is checked, also checking Merge to Lists overwrites (merges) the same lines in a list frame. You must provide the line key characteristic value for each line. If Add to List is checked, but Merge to Lists is not, trying to overwrite existing lines results in an error.

ASCII Loader Import - Select Catalog Files Screen

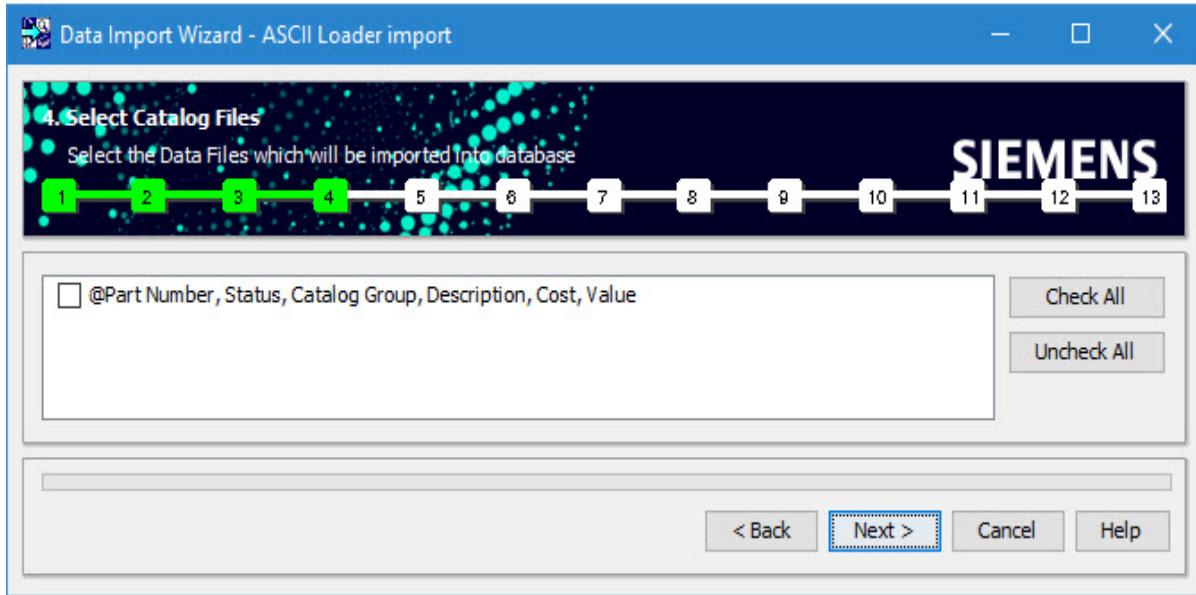
To access: Refer to [“Starting the ascl2dms Program”](#) on page 385

The ASCII Loader Import Select Catalog Data Files screen lists the column headers from the data file you specified in the Settings screen. You must verify the information to continue.

Previous screen: [ASCII Loader Import - Settings Screen](#)

Next screen: [ASCII Loader Import - Select Key Property Screen](#)

Figure 183. ASCII Loader Import Select Catalog Data Files (Screen 4)



ASCII Loader Import - Select Key Property Screen

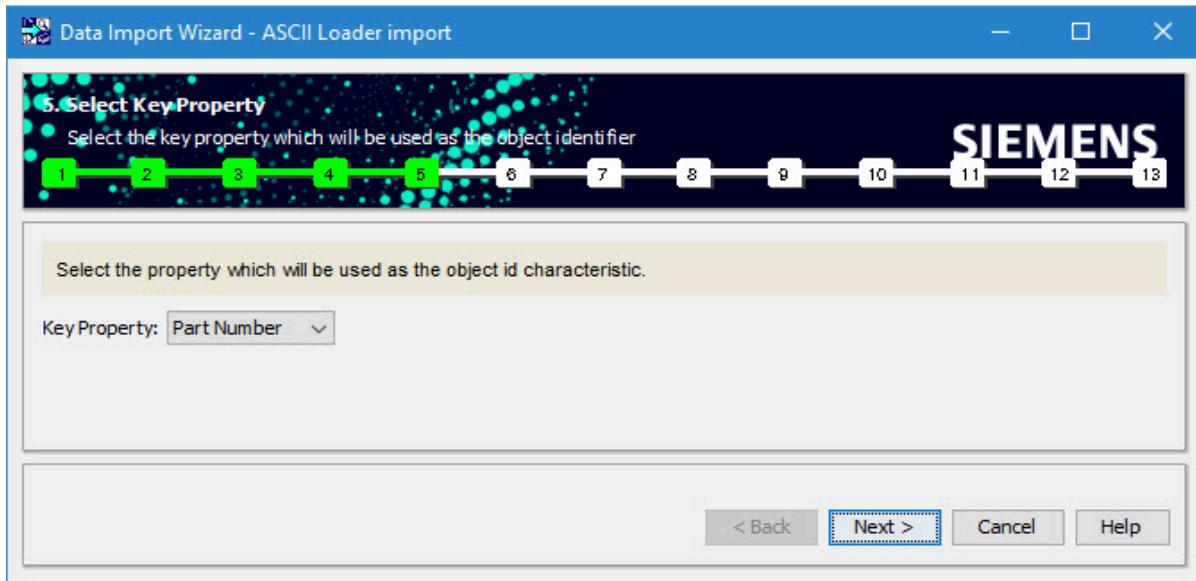
To access: Refer to “[Starting the ascl2dms Program](#)” on page 385

The ASCII Loader Import Select Key Property screen asks which column from the previous screen to use as the key value for new objects the wizard creates in the database. The property you select to use as the object key must have a unique value for each data object you load.

Previous screen: [ASCII Loader Import - Select Catalog Files Screen](#)

Next screen: [ASCII Loader Import - Create Catalog Groups Screen](#)

Figure 184. ASCII Loader Import Select Key Property (Screen 5)



ASCII Loader Import - Create Catalog Groups Screen

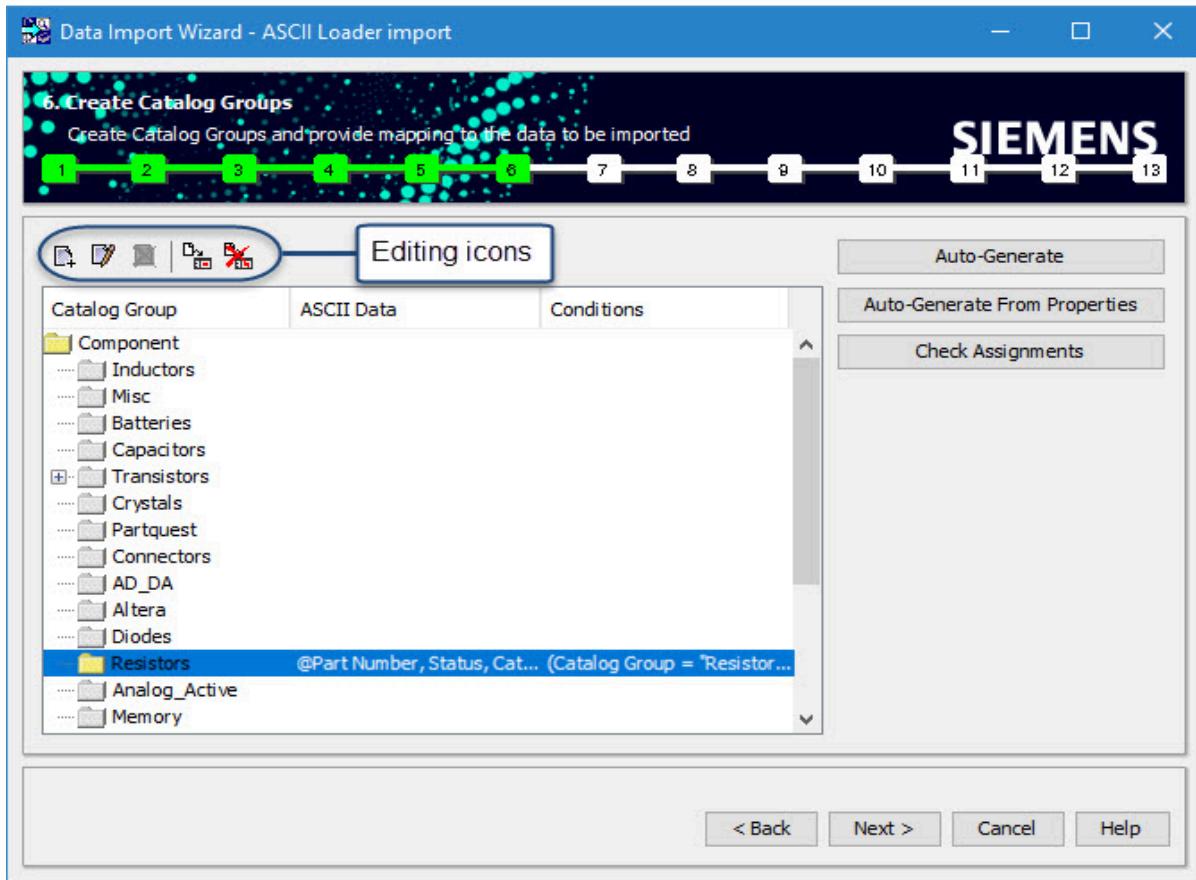
To access: Refer to “[Starting the ascl2dms Program](#)” on page 385

The ASCII Loader Import Create Catalog Group screen reflects the catalog grouping for the destination class specified earlier in the Settings screen.

Previous screen: [ASCII Loader Import - Select Key Property Screen](#)

Next screen: [ASCII Loader Import - Map Characteristics Screen](#)

Figure 185. ASCII Loader Import Create Catalog Group (Screen 6)



Usage Info

Use the Create Catalog Groups screen to specify the catalog groups where you would like to import the ASCII catalog data.

If a catalog hierarchy already exists in the database, that hierarchy displays here. You can use the editing icons to add, edit, or delete EDM Library catalog groups, and assign or clear the mapping of your ASCII catalog data to EDM Library catalogs.

If there is no catalog hierarchy shown, click **Auto-Generate** or **Auto-Generate From Properties** to automatically create a catalog hierarchy in the database.

- Click **Auto-Generate** to generate a flat catalog group structure that reflects the flat ASCII data structure. The software creates a matching EDM Library catalog group for any ASCII catalog that does not match any existing EDM Library catalog group.
- Click **Auto-Generate From Properties** to create an EDM Library catalog structure based on properties contained in the ASCII data. When choosing **Auto-Generate From Properties**, a dialog box displays letting you choose the properties to use for the different catalog levels.

The **Check Assignments** button displays a message that indicates if any ASCII catalogs are not yet assigned to a EDM Library catalog group.

ASCII Loader Import - Map Characteristics Screen

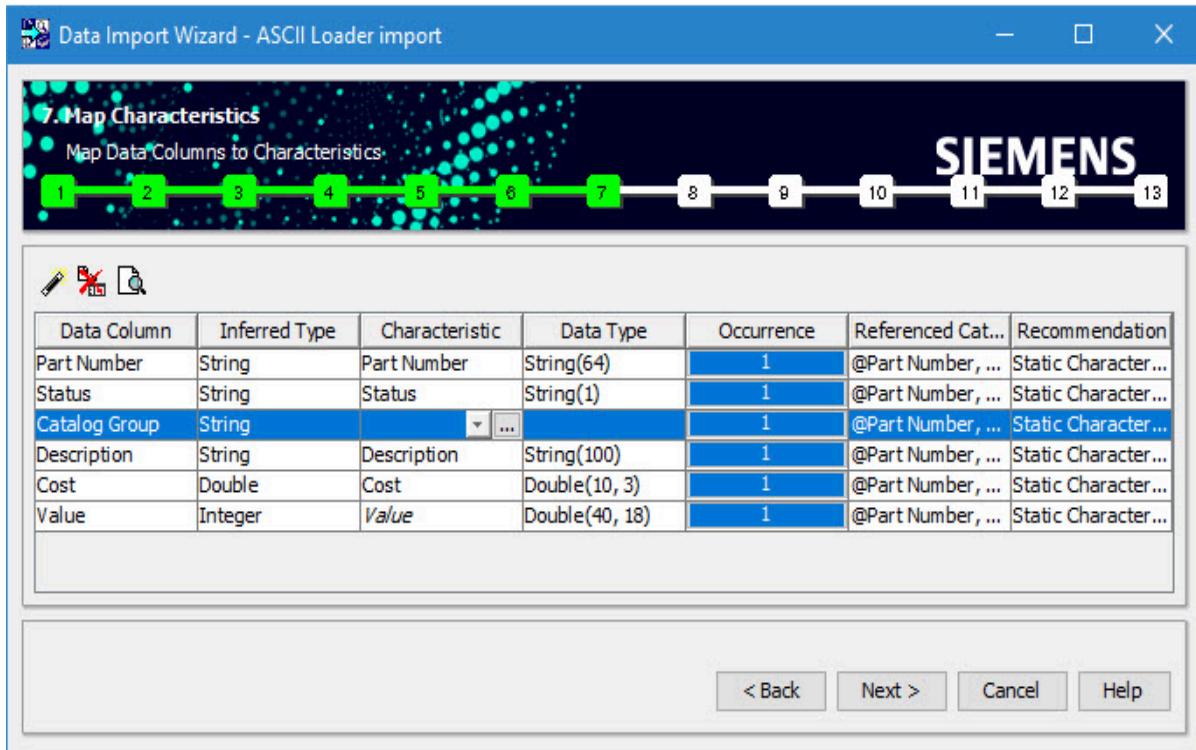
To access: Refer to “[Starting the ascl2dms Program](#)” on page 385

Use the ASCII Loader Import Map Characteristics screen to specify how to map the column names and values within ASCII catalogs to EDM Library characteristics and data types to use within EDM Library when importing the data.

Previous screen: [ASCII Loader Import - Create Catalog Groups Screen](#)

Next screen: [ASCII Loader Import - Verify Source Data Screen](#)

Figure 186. ASCII Loader Import Map Characteristics (Screen 7)



Usage Info

If there is no value in the Characteristic and Data Type column, select a row and use the pulldown menu to choose an existing static or dynamic EDM Library characteristic to map to, or click the **New Characteristic** button ().

When creating a new characteristic, you must choose a valid data type. A string is the most generic data type, while the date data type is most specific. Any source value that EDM Library attempts to store as a date data type must match the source data pattern specified on the Settings screen (refer to [Figure 182](#) on page 390).

The three buttons at the top perform the following actions when selecting one or more table rows (you can also right-click in the table to display a popup menu with the same functions):

- — Automatically create a EDM Library characteristic with the same name as the data column in the ASCII catalog.
- — Clear any previous EDM Library characteristic assignment(s).
- — Display values from the data file associated with a particular data column.

ASCII Loader Import - Verify Source Data Screen

To access: Refer to “[Starting the ascl2dms Program](#)” on page 385

The ASCII Loader Import Verify Source Data screen enables you to verify the source data before continuing.



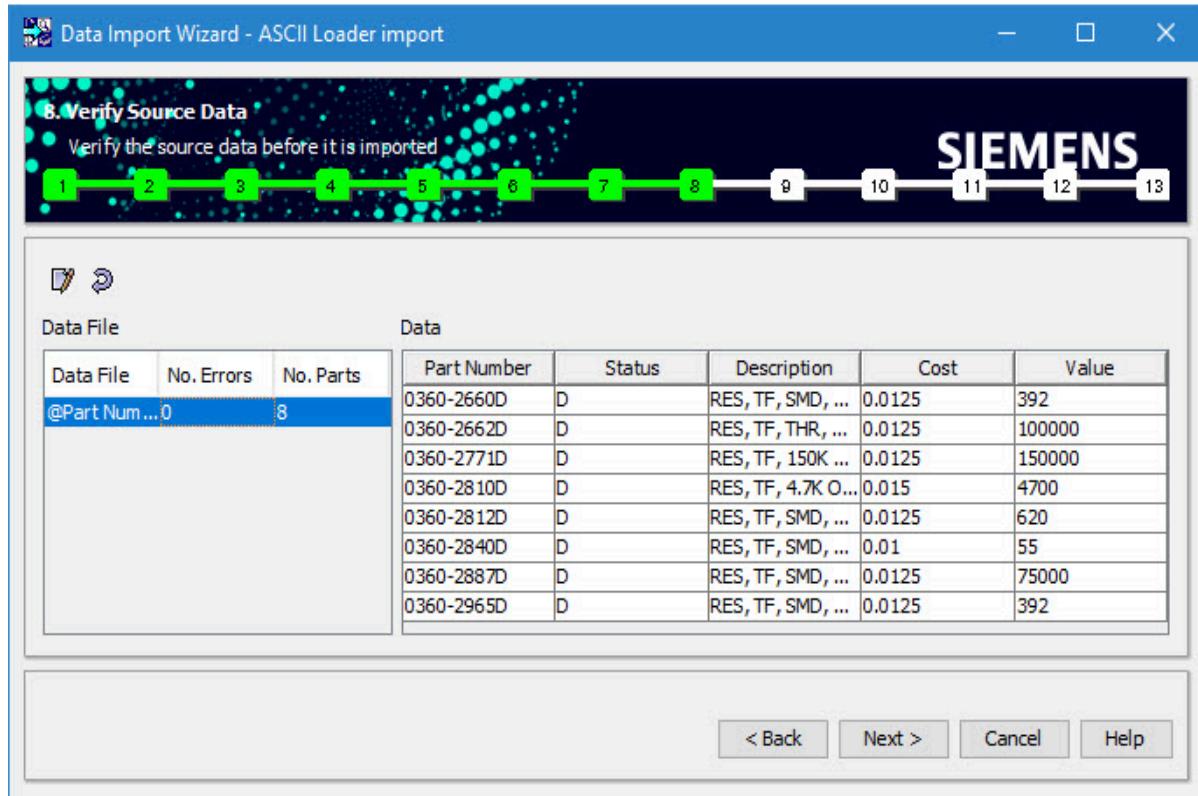
Note:

The Verify Source Data screen contains one column for each column in each ASCII data file. If the amount of data is very large, it could be difficult to see at one time. To see all the data in the Verify the Source Data screen, you might need to expand the screen, and then use the mouse cursor to widen the various columns.

Previous screen: [ASCII Loader Import - Map Characteristics Screen](#)

Next screen: [ASCII Loader Import - Import Settings Screen](#)

Figure 187. ASCII Loader Import Verify Source Data (Screen 8)



Usage Info

To change any value, select that cell and use the **Override Selected Value** button (). To revert back to the original value, use the **Revert** button (). Fields highlighted in pink are mandatory fields that do not currently have a value. You must supply a value in these fields before continuing.

String values in source data that are too long are in red, as are other input data errors. Proceeding with the import reports an error and does not import the object. Changing a value that is too long to something shorter enables a successful import. To sort the data for error values, click a column head. You can select and correct multiple values at once.



Note:

If you receive an error related to the date format, you might need to use a Java expression in the source date format field on the Settings screen instead of relying on the default value. For more information, refer to “[ASCII Loader Import - Settings Screen](#)” on page 389

ASCII Loader Import - Import Settings Screen

To access: Refer to “[Starting the ascl2dms Program](#)” on page 385

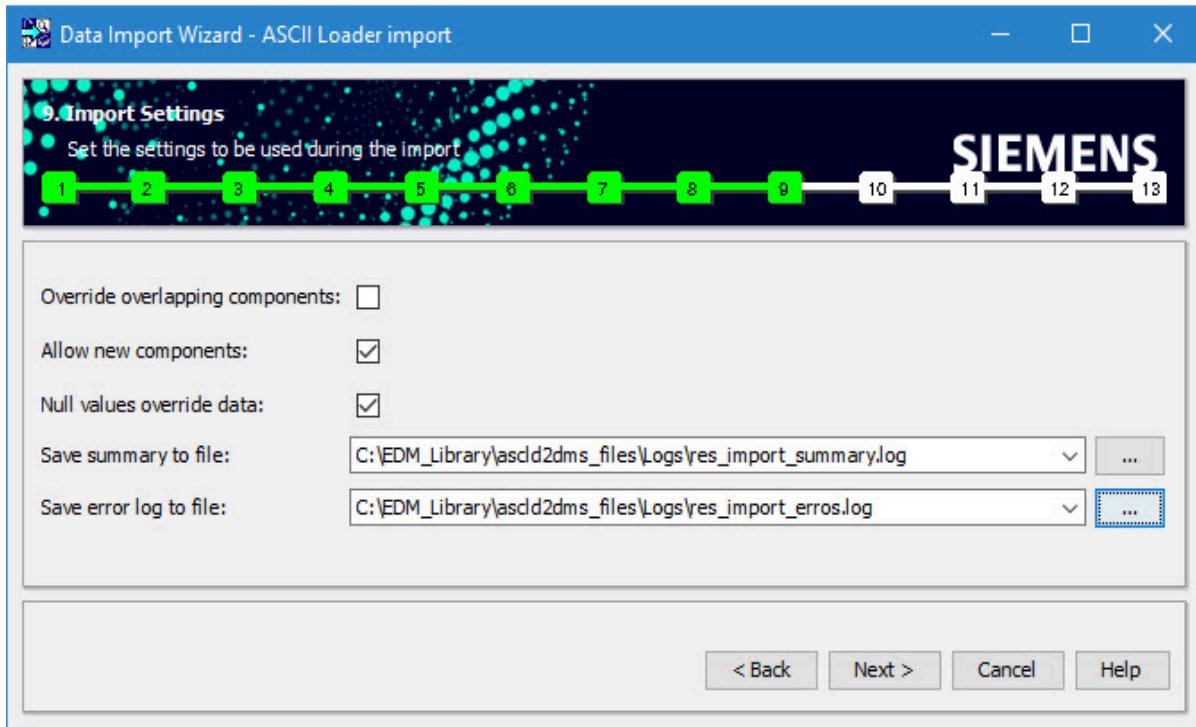
The Import Settings screen lets you verify whether or not to overwrite (that is, update) the values of objects that might already exist in the EDM database, whether to permit the creation of new objects

when they do not already exist, whether null (empty) values in the imported data should override existing values, and the locations to save a summary of the import and an error log.

Previous screen: [ASCII Loader Import - Verify Source Data Screen](#)

Next screen: [ASCII Loader Import - Save Session Transcript File Screen](#)

Figure 188. ASCII Loader Import Settings (Screen 9)



Usage Info

You cannot use the import manager to replace objects in the database with new objects that have different object key values (that is, search and replace functionality). To replace objects with different objects, delete the old object in the database, and then use the import manager to create the new object.

The summary log contains information about the general import process, such as the number of imported objects, the number of overridden objects, and so on. The error log lists any error messages that might have occurred during import.

ASCII Loader Import - Save Session Transcript File Screen

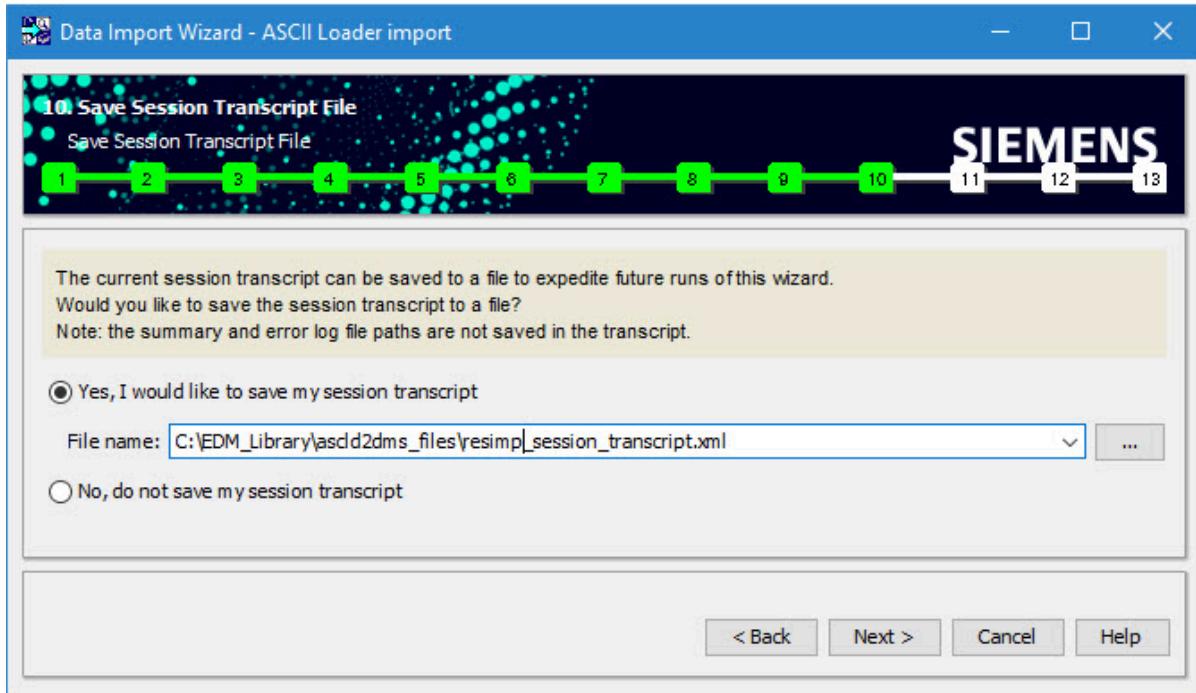
To access: Refer to [“Starting the ascl2dms Program”](#) on page 385

The Save Session Transcript file screen lets you save the transcript of the session for future execution.

Previous screen: [ASCII Loader Import - Import Settings Screen](#)

Next screen: [ASCII Loader Import - Creating Catalog Groups and Characteristics Screen](#)

Figure 189. ASCII Loader Import Save Session Transcript File (Screen 10)



Usage Info

Choose **Yes** to supply a path and name of an XML file, or **No** to continue without saving a session transcript. A session transcript enables you to load all the information you have entered in this session and then only change values that require modification.

ASCII Loader Import - Creating Catalog Groups and Characteristics Screen

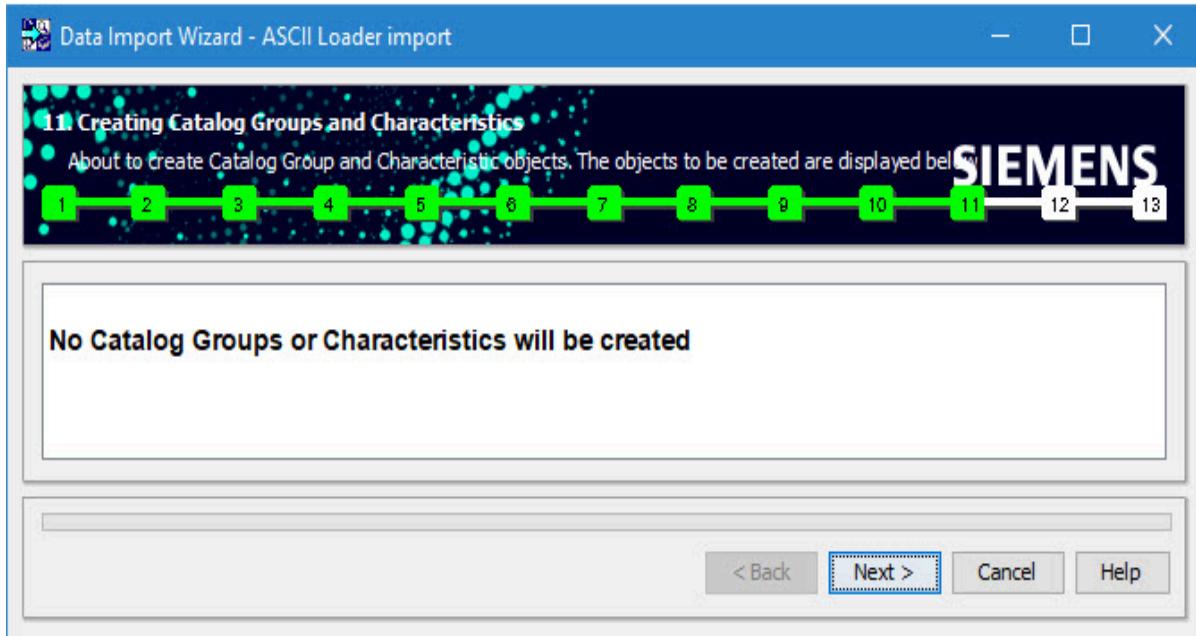
To access: Refer to “[Starting the ascl2dms Program](#)” on page 385

The ASCII Loader Import Creating Catalog Groups and Characteristics screen displays a list of catalogs, static characteristics, and dynamic characteristics to create in the EDM database.

Previous screen: [ASCII Loader Import - Save Session Transcript File Screen](#)

Next screen: [ASCII Loader Import - Importing Data Screen](#)

Figure 190. ASCII Loader Import Create EDM Library Objects (Screen 11)



Usage Info

Click **Next** to create the catalogs, static characteristics, and dynamic characteristics in the database and to display the next screen, which lists the EDM Library catalogs into which data will be imported.

Otherwise, click **Cancel** to exit the wizard or the **Back** button to go to a previous screen and change information.



Note:

If you are logged into an EDM database with an account that does not have rights to create Characteristic objects, an error message occurs.

ASCII Loader Import - Importing Data Screen

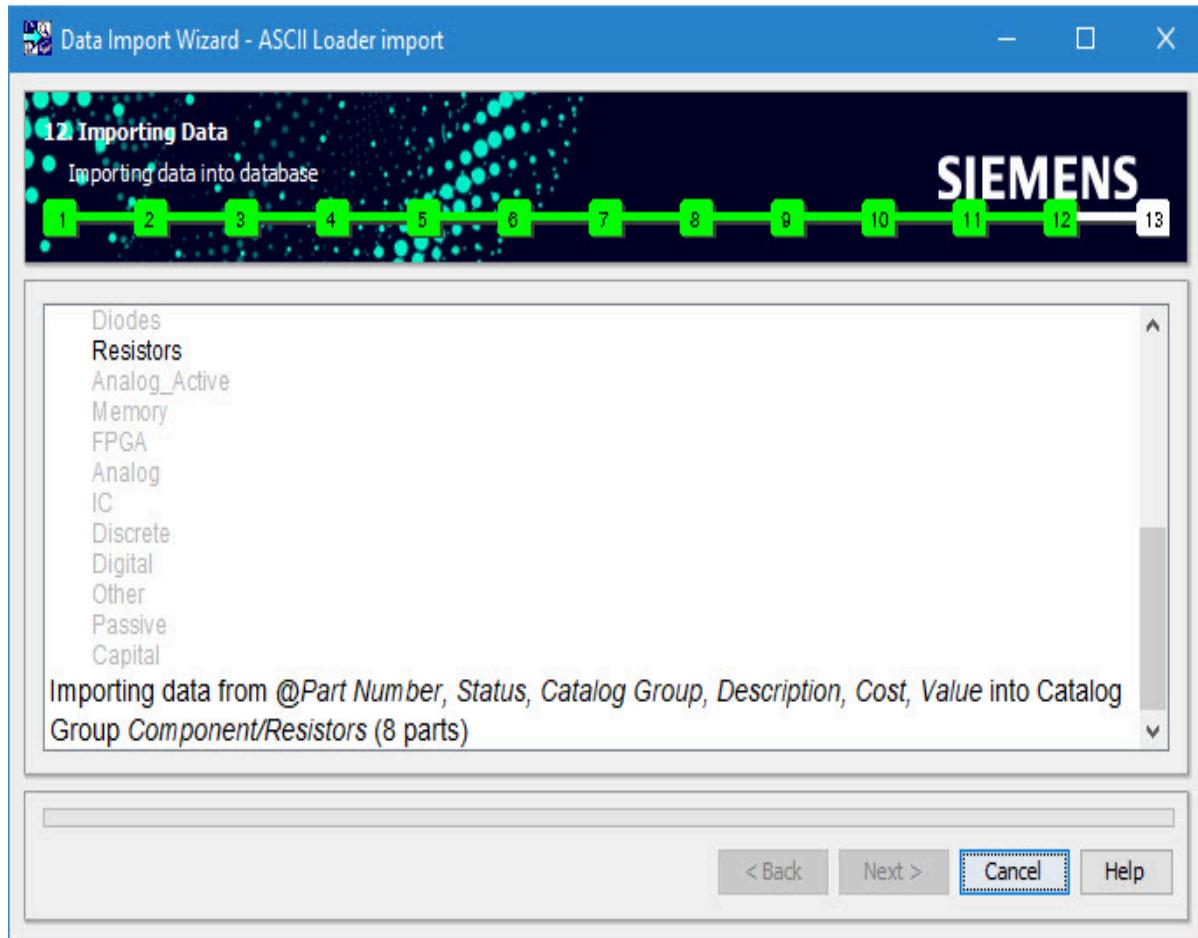
To access: Refer to “[Starting the ascl2dms Program](#)” on page 385

The ASCII Loader Import Importing Data screen lists the new EDM Library data as it is imported.

Previous screen: [ASCII Loader Import - Creating Catalog Groups and Characteristics Screen](#)

Next screen: [ASCII Loader Import - Final Summary Screen](#)

Figure 191. ASCII Loader Import Importing Data (Screen 12)



Usage Info

After all data imports, click **Next** to display a results summary.

ASCII Loader Import - Final Summary Screen

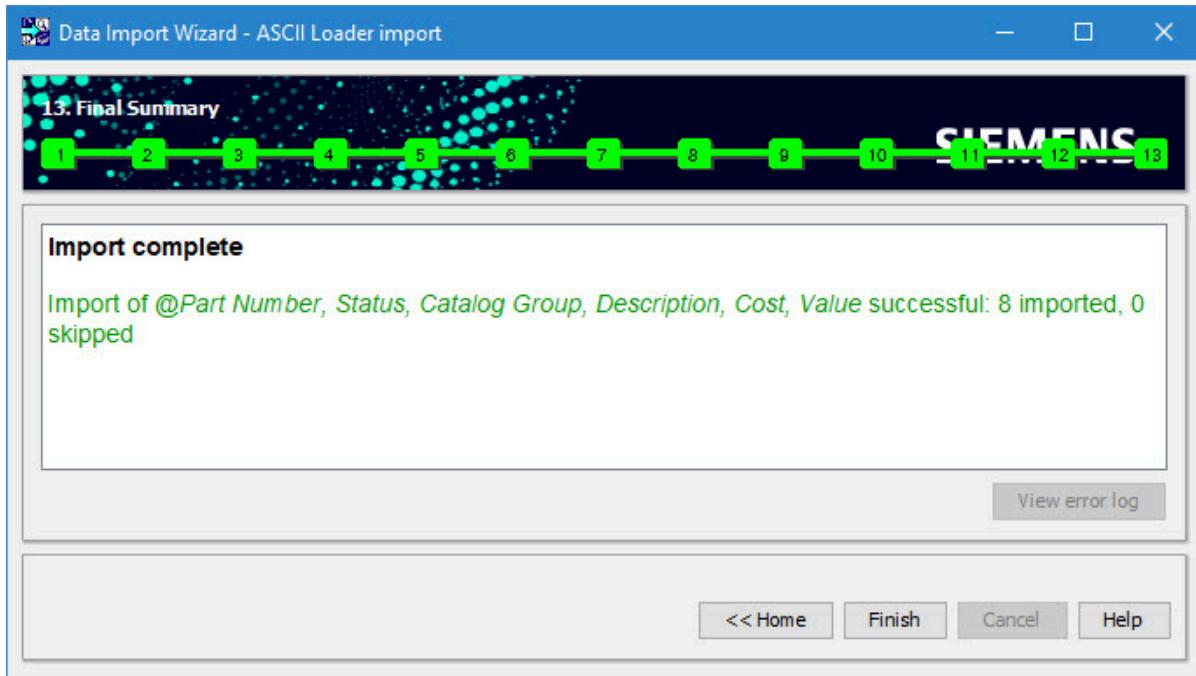
To access: Refer to “[Starting the ascl2dms Program](#)” on page 385

The ASCII Loader Import Final Summary screen displays a summary of the import.

Previous screen: [ASCII Loader Import - Importing Data Screen](#)

Next screen: None

Figure 192. ASCII Loader Import Final Summary (Screen 13)



ascl2dms Command Line Syntax

To access: In an EBS command window, type **ascl2dms**

You can use the **ascl2dms** command with the -batch switch to use the data import manager without the graphical user interface.

Syntax

```
ascl2dms [-version|-help|-batch]
          -dmsloginconfig config_name
          [-session transcript_file]
          [configuration_settings]
          [logging_options]
          mandatory_settings
          [optional_settings]
```

Arguments

Arguments of the **ascl2dms** command are grouped into the categories shown in the command syntax.

-version|-help|-batch

The -version, -help, or -batch switch must follow the **ascl2dms** command or the program defaults to running the graphical user interface.

- **-version**

Display the program version and exit. Specifying -version causes the program to ignore any other switches or flags.

- **-help**

Display command usage information and exit. Specifying -help causes the program to ignore any other switches or flags.

- **-batch**

Run the program in batch mode without further user interaction. The program gathers information from subsequent switch or flag settings.

-dmsloginconfig *config_name*

To use a stored login configuration with the **ascl2dms** command to establish a database connection, follow the -batch flag with the -dmsloginconfig <*config_name*> switch.

A login configuration locally stores complete user-specific connection information (username, password, server name, license selection, and production library name) that you would otherwise have to type into login dialog boxes. To create a login configuration, use the EDM Batch Login Setup program (refer to Automatic Login Configurations in the *Xpedition EDM Library Overview*).

For example, the following syntax tells the data import program to use the login configuration named "strom":

```
ascl2dms -batch -dmsloginconfig strom
```

-session *transcript_file*

The -session switch is optional. If used, follow the switch with the path to a transcript file. For example:

```
-session C:\imports\transcripts\ascload_xscpt_02082018.txt
```

configuration_settings

Configuration settings are optional. In some cases, configuration settings can also come from the session transcript file (if specified). Any configuration settings specified using a command line flag or switch override the value stored in a session file.

The following optional flags and switches specify configuration settings:

- **-casesensitive|-no-casesensitive**

Specifies case-sensitivity when matching source catalog names to EDM Library catalog groups. If not specified, the default is to not use case sensitive matching.

- **-dotdecimal|-no-dotdecimal**

Using -dotdecimal recognizes a period ('.') as a decimal point in source data (the default), while using -no-dotdecimal interprets a period as a string character rather than a decimal point.

- **-commadecimal|-no-commadecimal**

Using -commadecimal recognizes a comma (',') as a decimal point in source data (the default), while using -no-commadecimal interprets a comma as a character rather than a decimal point.

- **-srcdateformat *date_format***

Instructs the data import program to recognize any source data that follows the given date format as a date data type when importing. The format supplied as the <*date_format*> must be compatible with the format used in the java.text.SimpleDateFormat class in Java. If the -

ascl2dms Import Manager

ascl2dms Command Line Syntax

srcdateformat switch is missing from the command line, the data import program does not perform date recognition.

- **-srcencoding *encoding***

Specifies the source data encoding. Any value for *encoding* must be a subset supported by the JVM in use or names registered in IANA Charset Registry. If not supplied, the default encoding is iso-8859-1.

- **-override|no-override**

Use -override to merge imported values with existing EDM Library objects that have the same unique key identifier (the default). Use the -no-override flag to avoid modification of existing EDM Library objects.

- **-allownew|no-allownew**

Use -allownew to enable the creation of new EDM Library objects (the default). Specifying -no-allownew only permits the modification of existing EDM Library objects and does not permit creation of new EDM Library objects. You can use the -allownew|no-allownew switches in combination with the -override|no-override switches to support the following use cases:

- To modify only existing EDM Library objects and ignore new ones, use the -override and -no-allownew switches together.
- To import new EDM Library objects and leave existing EDM Library objects intact, use the -no-override and -allownew switches together.
- To import new EDM Library objects and modify existing EDM Library objects, use the -override and -allownew switches together.

Note that using the -no-override and -no-allownew switches together is not a valid combination because this always skips all imported data entries.

- **-nulloVERRIDE|no-nulloVERRIDE**

Using -nulloVERRIDE causes empty (null) values in imported data to replace existing characteristic values (the default) in the EDM database. Using -no-nulloVERRIDE ignores empty (null) values in imported data.

logging_options

Logging options are optional. If used, the following two switches specify where to store summary and error log information:

- **-summarylog *summary_path***

Specifies the path to the log file containing summary information.

- **-errorlog *error_log_path***

Specifies the path to the log file containing error information.

mandatory_settings

Mandatory switches have no default values.

- **-cfgfile** *config_path*

The path to the configuration file to use when loading ASCII data.

- **-dstclass** *class_no*

The number of the EDM Library class where the **ascl2dms** program should load objects. The default value for *class_no* is 1.

optional_settings

The following driver-specific switch and flags, which are also saved as settings in the session transcript file, are optional when you run the **ascl2dms** data import program in batch mode:

- **-datafile** *data_path*

The path to the data file to use. You can omit the -datafile switch if the data is in part of the configuration file.

- **-addtolist|no-addtolist**

Use the -addtolist flag to make the import program add new lines in list frames for column characteristics being imported if they are directed to the same object, without deleting existing lines. The default is -no-addtolist, which replaces existing lines in a list.

The -addtolist flag only has an effect if you also specify -override.

- **-mergetolist|no-mergetolist**

In addition to adding new lines, use -mergetolist to also overwrite (merge) the same lines in a list frame. You must provide the line key characteristic value for each line. If you specify -addtolist but not -mergetolist, trying to overwrite existing lines results in an error.

The -mergetolist flag only has an effect if you also specify -addtolist and -override. To express a negative value, use -no-mergetolist.

Appendix B

XML Console

Use the **xml-console.exe** command to export query result data from the database to an XML data file, or to import objects contained in an XML data file into the database.

Exporting data requires an XML query file and the name of a data file in which to store the query results. You can export data from any non-administrative object class that has a status value of A (Approved/Released), but can only export data from the following core administrative classes:

- Exported in a non-encrypted format: Toolbox (036), ProcessFlow (082), Units (086), and Labels (087) classes.
 - Exported in an encrypted format: MailGate Events (035), User (052), Search Preset (080), and Input Pattern Check (085) classes.
-



Note:

Do not use the **xml-console** command to create or modify EDA library objects in the following object classes: Mapping (10), Interface (70), Symbol (71), Package (3), Cell (130), Padstack (120), Pad (122), and Hole (123). Do not use **xml-console** to load EDA library object BLOBS. As an alternate to **xml-console**, use EDX Export and EDX Import for EDA library objects.

Importing data requires only the information contained in the data file. To avoid data files that are too large, a command switch enables you to split the output into multiple data files when exporting, or load from multiple data files when importing. You must also have edit rights when importing, which provides the ability to create objects in the target object classes and catalogs.

If data objects contain BLOBs, you can use a switch specify a separate source or destination directory for the BLOB data.

[xml-console Command](#)

[XML Query File Format](#)

[XML Data File Format](#)

[Bulk Loading Documents With xml-console](#)

xml-console Command

To access: In an EBS command window, type **xml-console**

The **xml-console** command can export object data from an EDM database to an XML data file, or import XML data file content into an EDM database. Exporting requires specifying the path to an XML query file in the command line, which describes the search criteria to return a set of data objects from the database for export.

Usage

To display command usage information:

```
xml-console -help
```

To import data into an EDM database from an XML file:

```
xml-console -configname login_config -import -importfile import_file
[-pack] [-blobdir blob_directory] [-verbose] [-complete] [-transaction]
```

To export data from the EDM database to an XML file:

```
xml-console -configname login_config -export -queryfile query_file
-outfile export_file [-dateformat UTC|LOCAL|LEGACY]
[-blobdir blob_directory] [-pack [-packsize n]] [-verbose]
```

Arguments

- **-help**

Display command help information. Using the -help switch ignores all other switch or flag settings.

- **-configname *login_config***

A required switch used to specify the name of an EDM Library login configuration.



Note:

When using **xml-console** to import data, an error can occur if a user does not have access to the object class, or does not have permission to edit characteristic values.

A login configuration is simply a way to locally store complete user-specific connection information (username, password, EDM Server name, license selection, and production library name) that you would otherwise have to enter manually (usually through a series of login dialog boxes) to connect with a database. Using the -configname switch means that no further login information is required to establish a database connection.

For information about how to create a login configuration, refer to “Automatic Login Configurations” in the *Xpedition EDM Library Overview*. For information about how to log into an EDM Server in general, refer to “Invoking EDM Library Cockpit” in the *Xpedition EDM Library Overview*.

- **-import**

Import data from an XML data file into the database. The -import flag must be followed by the -importfile switch.

- **-importfile *import_file***

The name of the XML data file to use as an import source (refer to “[XML Data File Format](#)” on page 421).

- **-export**

Export data from the database to an XML data file. The -export flag must be followed by the -output and -queryfile switches.

- **-outfile *export_file***

The name of the XML data file that is the output destination.

- **-queryfile *query_file***

The path to an XML query file (refer to “Examples” and “[XML Query File Format](#)” on page 415).

- **-dateformat UTC|LOCAL|LEGACY**

An optional parameter available only in export mode that specifies the timestamp value format for objects in the exported file. If not used, the default is to write timestamp values in UTC format. If used, -dateformat must have one of the following as a value:

- UTC — Coordinated universal time (UTC). Represent 24 hours using 0 through 23. For example, 2019-12-31 00:59:59 UTC
- LOCAL — Local time zone. Represent 24 hours using 0 through 23. For example, 2019-12-31 00:59:59 PST
- LEGACY — For compatibility with 7.9.x releases, represent 24 hours using 1 through 24 in the local time zone. For example, 2019-12-31 24:59:59 PST

- **-pack**

An optional flag that, when used, splits the output file into smaller XML data files or reads import information from smaller XML data files.

Using the -pack flag with the -output *export_file* switch names the first output file *<export_file>.xml*, the second output file *<export_file>.001*, the third *<export_file>.002*, and so on. Omitting the -packsize switch splits each *export_file* when it reaches a default of 5000 objects.

Using the -pack flag with the -import *import_file* switch loads *<import_file>.xml*, but then also looks for and loads *<import_file>.001*, *<import_file>.002*, and so on if found.

- **-packsize *n***

An optional switch used with the -output *export_file* switch and -pack flag to define the number of objects *n* to include in the export file before splitting the file. If the -packsize switch is not used, then the default is 5000 objects per output file.

- **-blobdir *blob_directory***

An optional switch specifying the input or output directory containing BLOB data for import or export.

BLOBS (Binary Large OBjects) represent files directly attached to objects. For example, an object might have an attached datasheet (.pdf file), a specification in Word (.docx) format, or

XML Console

xml-console Command

a .zip file representing multiple documents. You must specify the -blobdir switch to have these attachments included in the export or import.

For example, when specified with the -export switch, -blobdir C:\documents\ exports any files attached to objects into the C:\documents\ directory. When specified with the -import switch, -blobdir C:\documents\ imports files from the C:\documents\ directory and attaches them to the appropriate objects. If a referenced file is missing from the BLOB directory, **xml-console** imports the object without the attachment.

- -transaction

When importing, turns on transaction mode. If not used, each object is committed as it is imported.

- -verbose

An optional switch used to display detailed import or export progress information.

- -complete

When importing data, the default behavior of the xml-console command is to skip over the following characteristics and not overwrite values:

- Check-Out Status (co_status) and Checked Out By (co_user)
- Created By (ersteller) and Created At (erst_date)
- Last Modified By (bearbeit) and Last Modification Date (obj_datum)
- Sync Exclude (sync_exclude) and Last Synchronization Date (sync_date)
- Timestamp List (tlbx_ts_list)

Use the optional -complete switch to overwrite the values with time stamp data from the XML file.

Description

To determine the database with which to connect, the **xml-console** command requires the -configname switch followed by the name of an EDM Library login configuration. To determine whether the operation is an XML import or an XML export, the **xml-console** command line must contain either the -import or -export flag. The -importfile switch or -output switch specifies the path to the import source file or the path to the output destination file.

When using the **xml-console** command to export data, the command line must have the -queryfile switch to specify the location of a query file. The **xml-console** command runs the query file to determine the matching data within the EDM database to export. “[XML Query File Format](#)” on page 415 contains full information about the query file format. “[XML Data File Format](#)” on page 421 contains information about the format of the XML data file that is output by the **xml-console** command.

Using the query file, the **xml-console** command can export data from any non-administrative class that has a status value of A (approved/released), but restricts the export (and subsequent import) of data to the following set of EDM Library core object classes:

- Exported in a non-encrypted format: Toolbox (036), ProcessFlow (082), Units (086), and Labels (087) classes.
- Exported in an encrypted format: MailGate Events (035), User (052), Search Preset (080), and Input Pattern Check (085) classes.

Except for the ProcessFlow class, EDM Library core object classes have a status value of S (System).

Exporting or importing BLOB data requires using the -blobdir switch followed by the path to the directory that is either the export destination or the import source for the BLOB data.

Examples

This section contains the following **xml-console** command examples:

- A [Simple Export](#) example.
- A [Simple Import](#) example.
- An example of how to construct a query file and use the **xml-console** command when [Exporting Only Selected Characteristic Columns From a List Frame](#).
- [A Query File to Export a Complete Design \(MBOM\) and Referenced Variant BOMS](#).

Simple Export

To export data using **xml-console**, a query file must exist that returns a listing of the data to export (refer to [“XML Query File Format”](#) on page 415). The following example shows the content of a query file named *export.xml*, which the **xml-console** command uses to export all object data contained in the Component object class (class number 001):

```
<?xml version="1.0" encoding="UTF-8"?>
<export>
  <object class="001" catalog="AA" dynamic="true" unit="true" >
    </object>
  </export>
```

A class value of 001 and a catalog value of AA denotes that the export should include all Component objects from the component root class downward. A dynamic value of true exports all dynamic characteristics as well as fixed characteristics in the Component object class. A unit value of true exports double characteristics with the unit range name.

The following modifies the example *export.xml* query file to add a search restriction that only exports a selected set of components where the value of the part number characteristic (001obj_id) starts with “PN-”.

```
<?xml version="1.0" encoding="UTF-8"?>
<export>
  <object class="001" catalog="AA" dynamic="true" unit="true" >
    <restrictions>
      <restriction path="001obj_id" value="PN-*" />
    </restrictions>
  </object>
</export>
```

Now that the *export.xml* query file exists, the following **xml-console** command could be used to export the component data to the C:\mydata\Components.xml file from the database specified by the myconfig database configuration.

```
xml-console -configname myconfig -export -queryfile
C:\queryfiles\export.xml -outfile C:\mydata\Components.xml
```

Simple Import

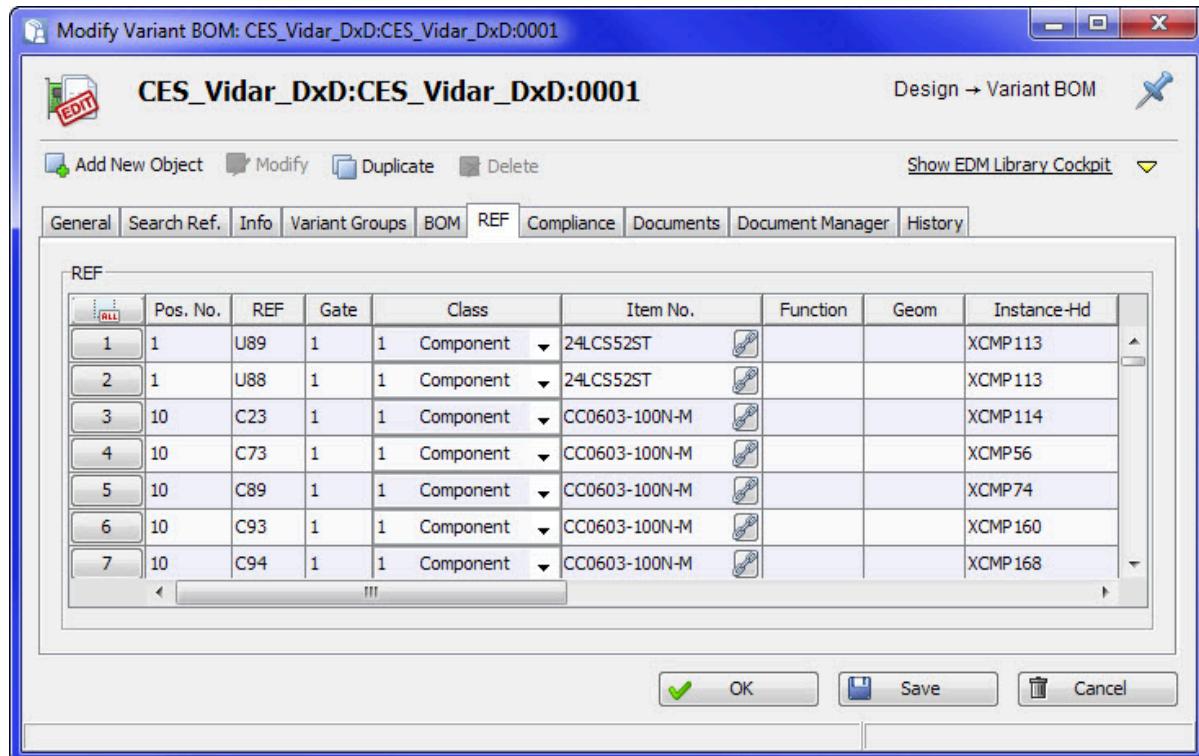
Given that exported data now exists in the *Components.xml* file, the following **xml-console** command could be used to re-import the data into a different database, as specified by the otherconfig EDM Library login configuration:

```
xml-console -configname otherconfig -import -importfile C:\mydata\Components.xml
```

Exporting Only Selected Characteristic Columns From a List Frame

Figure 193 shows the contents of the **REF** tab of a variant BOM object named CES_Vidar_DxD, whose data is derived from a Design (MBOM) object of the same name. All data to export is on the **REF** tab sheet and contained within the REF list frame. As the tab name implies, all data on the **REF** tab sheet is reference data from the parent Design (MBOM) object.

Figure 193. Ref Tab of a Variant BOM Object



The **xml-console** command uses the following example query, saved to a file named *vbom_query.xml*, to export only certain specified columns from the **REF** tab of the CES_Vidar_DxD variant BOM object.

```
<?xml version="1.0" encoding="UTF-8"?>
<export xmlns:ns="noNamespaceSchemaLocation="XmlExportCommand.xsd">
<!-- Variant BOM -->
<object class="021" emptyfields="true" noninputfields="true"
defaultvalues="true" unit="true">
<restrictions>
```

```
<restriction path="021ref_sum" value="CES_Vidar_DxD_Alt" />
</restrictions>
<only>
  <field id ="021bmkliste" />
  <field id ="021bmkliste.021b_cadref" />
  <field id ="021bmkliste.021b_gate" />
  <field id ="021bmkliste.021b_idnrk" />
  <field id ="021bmkliste.021b_geom" />
</only>
</object>
</export>
```

In the above query file, a noninputfields value of true enables the export of characteristics whose Input Characteristic bit status value is unchecked (that is, EDM Library determines the characteristic values by reference rather than user input). If noninputfields is not set (or set to false), the query fails and returns an empty file because it cannot export the characteristic values.

**Note:**

Exporting data from non-input characteristics means that, because the characteristics do not permit data input, the exported characteristic values cannot be imported back into the database.

The section bracketed by the `<restrictions>` and `</restrictions>` markers create a search restriction that returns the CES_Vidar_DxD variant BOM object.

The query then uses the section bracketed by the `<only>` and `</only>` markers to search for the columns from within the REF list frame (021bmkliste) to export. To identify which columns to include in the export, the field id value uses the format *list_frame_name.column_name*. The query file example exports the data contained in the following four columns of the REF list frame:

- REF (021bmkliste.021b_cadref)
- Gate (021bmkliste.021b_gate)
- Item No. (021bmkliste.021b_idnrk)
- Geom (021bmkliste.021b_geom)

The section bracketed by the `<only>` and `</only>` markers must come after the search restriction.

Now that the `vbom_query.xml` query file exists, the following `xml-console` command is used to export the four columns of data from the CES_Vidar_DxD variant BOM object to the `C:\mydata\CES_Vidar_REF.xml` file from the database specified by the `my_config` database configuration.

```
xml-console -configname myconfig -export -queryfile C:\queryfiles\vbom_query.xml -outfile C:\mydata\CES_Vidar_REF.xml
```

A Query File to Export a Complete Design (MBOM) and Referenced Variant BOMS

By performing multiple queries using multiple `<object>` tags, you can export multiple objects into a single data file. The following query file example exports data from a design (MBOM) object named “NextWave” along with data from all referenced variant BOMs.

```
<?xml version="1.0" encoding="UTF-8"?>
<export xmlns:noNamespaceSchemaLocation="XmlExportCommand.xsd">
<!-- Master BOM -->
<object class="020" unit="true">
<restrictions>
  <restriction path="020obj_id" value="NextWave" />
</restrictions>
<ignore>
  <field id = "020posliste" />
</ignore>
</object>

<!-- Variant BOM -->
<object class="021" unit="true" >
<restrictions>
  <restriction path="021ref_sum" value="NextWave" />
</restrictions>
</object>
</export>
```



Note:

The section bracketed by the `<ignore>` and `</ignore>` markers instruct the query to ignore the field associated with the `020posliste` characteristic, which is a legacy field applicable to the older EDM Library classic interface but that is no longer used. If you do not include a statement ignoring the `020posliste` characteristic, doubled entries can occur upon import into the EDM database.

Related Topics

[Command Line Syntax Conventions](#)

[XML Query File Format](#)

[XML Data File Format](#)

XML Query File Format

An XML query file defines how the **xml-console** command should query the database so as to return only objects with matching characteristic values as output to an XML data file. You can also use the query file to restrict the set of object characteristics to include in the export. Constructing a query schema file requires that you follow the format described in this section.

“Examples” in “[xml-console Command](#)” on page 408 contains additional examples of query files used by the **xml-console** command.

[Required XML Query File Tags](#)

[Search Restrictions in the XML Query File](#)

[Field Identifiers in the XML Query File](#)

[Example XML Query Files](#)

Required XML Query File Tags

An query file must always have a set of required tags. Attributes can exist between the `<object>` and `</object>` tags that form a query to return the data for export.

The following shows the query file format:

```
<?xml version="1.0" encoding="UTF-8"?>
<export>
  <object class="value" attribute1="value" attribute2="value" ...
          attributeN="value">
  </object>
</export>
```

Required Tags

The following tags must always exist in the XML query file:

- `<?xml version="1.0" encoding="UTF-8"?>` — Declares the XML version and encoding. The only acceptable value are a version value of “1.0” and an encoding value of “UTF-8”.
- `<export>` and `</export>` — The beginning and end of the query file export information.
- `<object>` and `</object>` — The beginning and end of the section to identify the class from which to export one or more objects. Contained within the `<object>` tag are key attribute declarations and values using the format `attribute="name"`.

When exporting data from several classes or from different catalog groups in the same class, a query file might contain several sections bracketed by `<object>` and `</object>` tags.

Key Attributes for the `<Object>` Tag (Query File)

The following are key attributes that can be included as part of the `<object>` tag in a query file. The class attribute must occur first and is always required, but other attributes are optional.

- **class="num"** — Identifies the EDM Library class containing data to export. The value must be the three digit number that uniquely identifies the class within the EDM database (for example, class="101"). The **xml-console** command can export data from any non-administrative class with a status value of A (Approved Released), but can only export data from the following core classes (that is, classes in the Admin module in the class hierarchy pane of Xpedition EDM Library Cockpit):
 - Exported in a non-encrypted format: Toolbox (036), ProcessFlow (082), Units (086), and Labels (087) classes.
 - Exported in an encrypted format: MailGate Events (035), User (052), Search Preset (080), and Input Pattern Check (085) classes.

The **xml-console** command cannot export data from the core object classes class (099) or the characteristics class (056). Except for the ProcessFlow class, EDM Library core object classes have a status value of S (System).

- **catalog="catalog_id"** — Indicates the EDM Library catalog within the class containing data to export. The value must be the catalog group identifier (for example, catalog="GGAAAD").
- **dynamic="true|false"** — A "true" or "false" value that indicates whether to export dynamic characteristics (for example, dynamic="true").
- **emptyfields="true|false"** — A "true" or "false" value that indicates whether to export a characteristic without a value (for example, an empty string) as empty tags.
- **noninputfields="true|false"** — A "true" or "false" value that indicates whether to export non input characteristics.
- **defaultvalues="true|false"** — A "true" or "false" value that indicates whether to export values from "take-over value" characteristics. Take-over value characteristics have the "take-over value" status flag set and are composed with values from other characteristics (for example, ^libspec^:^snr^). The **xml-console** command can export such values to an XML file but are ignored during an import, similar to "noninputfields" option.
- **unit="true|false"** — A "true" or "false" value that indicates whether to export double characteristics with units.
- **graphic="true|false"** — A "true" or "false" value that specifies whether to export the EDM Library CAD graphic (if it exists for the data object).
- **numwithnull="true|false"** — A "true" or "false" value that indicates whether to mark empty fields with null=true.
- **comments="true|false"** — A "true" or "false" value that indicates whether to place XML comments into the exported XML data file.
- **broken_ref="true|false"** — For an import, a broken_ref attribute value of "true" loads objects with broken references while a value of "false" excludes objects with broken references. For an export, a value of "true" exports all objects returned by the query result (regardless if those objects have broken references), while a value of "false" exports all objects except those with broken references.

The following shows an <object> tag with attributes and values:

```
<object class="021" emptyfields="true" noninputfields="true"  
defaultvalues="true" unit="true">
```

Related Topics

[Search Restrictions in the XML Query File](#)

[Field Identifiers in the XML Query File](#)

Search Restrictions in the XML Query File

You can specify additional search restrictions in the query file by adding <restrictions> and </restrictions> tags, which identify the beginning and ending of the search restriction section for a specific object class. Within the section are one or more <restriction> declarations. Each <restriction> identifies a specific characteristic restriction to use when matching an object for export.

The following shows the format of the query file with additional search restrictions:

```
<?xml version="1.0" encoding="UTF-8"?>  
<export>  
  <object element1="value" element2="value" ... elementN="value">  
    <restrictions>  
      <restriction path="char_ID" value="char_value" >  
      ...  
      ...  
      <restriction path="char_ID" value="char_value" >  
    </restrictions>  
  </object>  
</export>
```

Key Attributes for the <Restriction> Tag

The following key attributes can be included as part of the <restriction> tag.

- **path="char_ID"** — Indicates the identifier of the characteristic to restrict (for example, path="101library_part").
- **value="char_value"** — Indicates the restriction value for this characteristic (for example, value="PN-*"). Values can be formatted using the same set of special characters accepted by the search window in Xpedition EDM Library Cockpit.

The following shows a restriction section within a query file:

```
<restrictions>  
  <restriction path="001obj_id" value="PN-*" />  
</restrictions>
```

Related Topics

[Required XML Query File Tags](#)

[Field Identifiers in the XML Query File](#)

Field Identifiers in the XML Query File

Field identifiers enable inclusion or exclusion of certain characteristics. To use field identifiers to only export a certain set of characteristics contained within a class, create a section bracketed by the `<only>` and `</only>` tags. Similarly, create a section bracketed by `<ignore>` and `</ignore>` tags in the query file to specify characteristics to exclude.

An inclusion or exclusion section in your query file is especially useful if you are exporting tabular data and only want to export a subset of columns.

If the query file does not include a section using `<only>` and `</only>` tags or `<ignore>` and `</ignore>` tags, the **xml-console** command exports all characteristics for that class. Within the inclusion or exclusion section are one or more `<field id="name">` declarations (for example, `<field id ="021bmkliste" />`). The following shows the format of the query file with included characteristics:



Note:

If you have an `<only>` section, it must follow any restrictions, as shown in the following example. If you place the `<only>` section before the `<restrictions>` section, the **xml-console** command fails with a parsing error.

```
<?xml version="1.0" encoding="UTF-8"?>
<export>
  <object element1="value" element2="value" ... elementN="value">
    <restrictions>
      <restriction path="char_ID" value="char_value" >
        ...
        ...
        <restriction path="char_ID" value="char_value" >
      </restrictions>
      <only>
        <field id = "value" />
        ...
        ...
        <field id = "value" />
      </only>
    </object>
  </export>
```

The following shows example field identifiers within the inclusion (only) section of a query file:

```
<only>
  <field id = "021bmkliste" />
  <field id = "021bmkliste.021b_cadref" />
  <field id = "021bmkliste.021b_gate" />
```

```
<field id ="021bmkliste.021b_idnrk" />
<field id ="021bmkliste.021b_geom" />
</only>
```

The following shows an example field identifier within the exclusion (ignore) section of a query file:

```
<ignore>
<field id ="020posliste" />
</ignore>
```

Related Topics

[Required XML Query File Tags](#)

[Search Restrictions in the XML Query File](#)

Example XML Query Files

You can create XML query files for use with the **xml-console** command that act like queries you run in the search pane of the Xpedition EDM Library Cockpit tool.

Refer to the “Examples” subsection of [“xml-console Command”](#) on page 408 for additional query file examples.

query_comp_filter_catalog.xml

This example returns all Component objects contained in the Capacitor catalog.

```
<?xml version="1.0" encoding="UTF-8"?>
<export>
<object class="001" dynamic="true" noninputfields="false"
       comments="true" broken_ref="true">
<restrictions>
<restriction path="001obj_txkg" value="Capacitor"/>
</restrictions>
</object>
</export>
```

query_comp_filter_status.xml

This example returns all Component objects with a status of D (In Development).

```
<?xml version="1.0" encoding="UTF-8"?>
<export>
<object class="001" dynamic="true" noninputfields="false"
       comments="true" broken_ref="true">
<restrictions>
<restriction path="001obj_statu" value="D"/>
</restrictions>
</object></export>
```

query_toolboxes.xml

The toolbox class (036) contains multiple configuration objects. This example shows how to return all toolboxes with a Call name that starts with "Tutorial".

```
<export>
  <object class="036" dynamic="true" noninputfields="false"
         comments="true" broken_ref="true">
    <restrictions>
      <restriction path="036snr" value="Tutorial*" />
    </restrictions>
  </object>
</export>
```

Related Topics

[xml-console Command](#)

XML Data File Format

This section describes the XML data file format that occurs as a result of using the **xml-console** command to export data from an EDM database. The **xml-console** command can also use an XML data file as a source when importing data from the file system into an EDM database.

- [General XML Data File Structure](#)
- [Key Attributes for the <object> Tag \(Data File\)](#)
- [The <delete> Tag](#)
- [Editing Characteristic Values in the XML Data File](#)

General XML Data File Structure

An exported XML data file must follow a specific structure. Understanding the structure enables you to parse the exported data or create a data file to use for import.

Although you can manually create an XML data file, the best process is to create an export file using a query and then manually modify the file (which ensures proper XML format).

```
<?xml version="1.0" encoding="UTF-8"?>
<data>
    <date format="format"/>
    <object objectid="value" element1="value" element2="value" ...
        elementN="value" class="value" catalog="value">
        <!--Label-->
        <field id="char_name">char_value</field>
        <!--Label-->
        <field id="char_name">char_value</field>
        .
        .
        .
        <!--Label-->
        <field id="char_name">char_value</field>
        <list id="char_name" clear="true">
            <!--Label-->
            <field id="frame_name">char_value</field>
            .
            .
            .
            <!--Label-->
            <field id="char_name">char_value</field>
        </list>
    </object>
</data>
```

The following tags must always be in the XML data file:

- **<?xml version="1.0" encoding="UTF-8"?>** — Declares the XML version and encoding. The only acceptable value are a version value of "1.0" and an encoding value of "UTF-8".

- **<data> and </data>** — Specify the beginning and end of the data in the XML data file.

- **<date>** — Specifies the format of date timestamps for objects that follow in the data file.

If you do not use the -dateformat switch during an export, the default is to use the coordinated universal time (UTC) format of "yyyy-MM-dd HH:mm:ss z" for all timestamps. For more information about date formats, refer to Preferences Dialog Box - General Tab in the *Xpedition EDM Library Overview*.

- **<object> and </object>** — Specify the beginning and end of the section to identify the characteristics and values for a particular EDM database object. Contained within the <object> tag are key attribute declarations and values using the format attribute="name".

- **<field id>** — Specify the unique identifier of a characteristic existing on the object and its value.

- **<list>** — Denotes a EDM Library list frame, where the *id* attribute is the name of the list frame characteristic. The *clear* attribute indicates if this is first element of a list and whether to clear the list before import. Failing to clear the list can generate an import error if the list item already exists.

The *list* tag contains nested <field id> tags.

Related Topics

[Key Attributes for the <object> Tag \(Data File\)](#)

[The <delete> Tag](#)

[Editing Characteristic Values in the XML Data File](#)

Key Attributes for the <object> Tag (Data File)

Like an XML query file, the <object> tag in an XML data file can have similar key attributes:

- **objectid** — The unique identifier of the data object.
- **noninputfields** — Indicates whether non-input characteristic were exported.
- **defaultvalues** — Indicates whether default values were exported.
- **class** — Indicates the EDM Library class for this object.
- **catalog** — Indicates the EDM Library catalog for this object.
- **graphic** — A string value that is the name of the BLOB containing graphics.

- **field** — Represents a single exported characteristic, where *id* is the name of the characteristic. The field tag value represents the EDM Library object value for the characteristic. Each field is commented with `<!-- EDM Library characteristic label -->`.
- **multirefclass** — Indicates the given reference class.
- **broken_ref="true|false"** — A broken_ref attribute value of “true” loads objects with broken references while a value of “false” excludes objects with broken references.

For example, to load a Production Library object where some referenced Component objects are not yet in the database, set the broken_ref attribute to true. Similarly, set the broken_ref attribute to true to load Component objects before populating the database with referenced Document objects. In each of these cases, setting the broken_ref attribute to false (or not including the broken_ref attribute at all) would not load an object if it has a broken reference.

Related Topics

[General XML Data File Structure](#)

[Editing Characteristic Values in the XML Data File](#)

[The <delete> Tag](#)

The <delete> Tag

When importing, the **xml-console** command enables you to delete objects already residing in an EDM database by including a <delete> tag in the XML data file.

For example, the following declaration in the XML data file deletes the Res1 object from the AAAB catalog of the Components class (class 001).

```
<data>
  <delete objectid="Res1" class="001" catalog="AAAB" />
</data>
```

Related Topics

[General XML Data File Structure](#)

[Key Attributes for the <object> Tag \(Data File\)](#)

[Editing Characteristic Values in the XML Data File](#)

Editing Characteristic Values in the XML Data File

Once you understand the data structure of an XML data file, you can edit the <field id> lines in the file to change the names and location of objects, or other characteristic values when importing.

**Note:**

Do not edit the exported XML data files so as to create new objects or modify characteristic values on EDA library objects in the any following object classes: Mapping (10), Interface (70), Symbol (71), Package (3), Cell (130), Padstack (120), Pad (122), and Hole (123). Do not use **xml-console** to load EDA library object BLOBs. For EDA library objects, use EDX Export and EDX Import as an alternate to **xml-console**.

Example XML Data File With Several Characteristic Definitions

The following example shows an XML data file that specifies the parameters for a resistor component object named Res1 that resides in catalog AAAB. The object has the following characteristic values, as defined by the <field id> lines:

- 000resistance is a single text dynamic characteristic.
- 001audit_valid_to is a characteristic without a value (that is, an empty string).
- 001erst_date is a date characteristic.
- 001pow_max is a double field with a unit value.
- 021ref_idnrk is a multiclass reference.

```
<?xml version="1.0" encoding="UTF-8"?>
<data>
    <date format="yyyy-MM-dd HH:mm:ss z" />
    <object objectid="Res1" noninputfields="true" defaultvalues="true"
    class="001"
    catalog="AAAB">
        <!--Resistance-->
        <field id="000resistance">RE1</field>
        <!--Model-->
        <field id="001model_param">MODEL</field>
        <!--001obj_cod-->
        <field id="001obj_skn">AAAB</field>
        <!--Status sort-->
        <field id="001statusort">0</field>
        <!--Audit valid to-->
        <field id="001audit_valid_to" />
        <!--Creation Date-->
        <field id="001erst_date">2008-07-29 11:41:03.0</field>
        <!--Item No.-->
        <field id="021ref_idnrk" multirefclass="001">Res1</field>
        <!--Max Power Dissipation-->
        <field id="001pow_max">50.0kW</field>
        <list id="001testframe" clear="true">
            <!--Test-->
            <field id="001testval">ABC</field>
        </list>
        <!--Test Frame-->
    </object>
</data>
```

```

<list id="001testframe">
    <!--Test-->
    <field id="001testval">123</field>
</list>
<!--Test Frame-->
<list id="001testframe">
    <!--Test-->
    <field id="001testval">XYZ</field>
</list>
</object>
</data>

```

Moving Objects to a New Location

You can change the <field id> line that defines the catalog location to import the object into a different catalog. For example, to move object ABC123 from the AAAB catalog to the AAAC catalog, you would change the value of the 001obj_skn (Catalog ID) characteristic (shown in bold).

```

<?xml version="1.0" encoding="UTF-8"?>
<data>
    <object objectid="ABC123" class="001" >
        <field id="001obj_skn">AAAC</field>
    </object>
</data>

```

Renaming an Object

You can change the <field id> line that defines the value of the key identifier characteristic to rename the object. This example changes the value of the 001obj_id (Part Number) characteristic of the AMP531134_240FV object to 8800-0001 (shown in bold). An error occurs if a Component object named 8800-0001 already exists.

```

<?xml version="1.0" encoding="UTF-8"?>
<data>
    <date format="yyyy-MM-dd HH:mm:ss 'UTC'" />
    <object objectid="AMP531134_240FV" class="001">
        <!--Part Number-->
        <field id="001obj_id">8800-0001</field>
    </object>
</data>

```

Common Editing Issues

When editing the XML data file, you need to be careful not to make changes that can cause problems during the import. The following are issues to be aware of when editing the data file:

- Characteristics in the XML data file that do not exist in the data model.
- Objects reference other objects that are not in the database.

- The data file is missing mandatory characteristic values for an object.
- A characteristic value does not have the correct format, unit, or input pattern.

Related Topics

[General XML Data File Structure](#)

[Key Attributes for the <object> Tag \(Data File\)](#)

[The <delete> Tag](#)

Bulk Loading Documents With xml-console

You can create an XML data file and then use the **xml-console** command with the -blobdir switch to quickly bulk load datasheets or other documents from the filesystem into the Documents object class. After you load the documents, you can run the **xml-console** command with a second data file to associate them to Component objects.

[Loading Documents in Bulk](#)
[Associating Documents to Components](#)

Loading Documents in Bulk

Bulk loading with the **xml-console** command provides a quick way to get documents into the database as Document objects, where they can then be associated with Component objects.

Procedure

1. Place all documents to load into a single directory on the file system (for example, C:\edm_documents).

The path to this directory follows the -blobdir option of the **xml-console** command line in Step 3.

2. Create an XML data file (refer to Examples) with one <object></object> section describing each Document object to create.

Within each <object></object> section, follow these rules to define characteristic values that are on the **General** tab of each new document object:

- Use the value of the Document Name characteristic (110snr) as the objectid value, with “:1:1” appended. For example, if the 110snr value is “datasheet_example.pdf”, then the objectid value is:

```
<object objectid="datasheet_example.pdf:1:1">
```

Alternately, set all of the objectid values to a placeholder value such as “dummy_id” or “”, and then permit the import to change the name based on the Document Name (110snr) key characteristic value.

- Define a unique value for the Document Name key characteristic (110dokname). For example:

```
<!--Document Name-->
<field id="110snr">datasheet_example.pdf</field>
```

- Define the key value of the Catalog Group characteristic (110obj_skn) that denotes where to place the imported document. For example:

```
<!--Catalog Group Key -->
<field id="110obj_skn">NNDM</field>
```

- Optionally, define a value for the Title of Document characteristic (110dokname). For example:

```
<!--Title of Document-->
<field id="110dokname">XMLIO datasheet_example.pdf</field>
```

- Optionally, define a value for the Long Description characteristic (110dokbemerek). For example:

```
<!--Long Description-->
<field id="110dokbemerk">Loaded by XML utility</field>
```

Within the `<list></list>` subsection, follow these rules to identify the attached loadable files and define the characteristic column values in the File Information list box (110doc_lst) on the **Attachments** tab of the new documents object:



Note:

If you are attaching a second file to the same Document object, ensure that the `<list id="110doc_lst">` tag does not contain the `'clear="true"` attribute.

- Set the Index column characteristic (110doc_idx) value to 0.
- Optionally, set the File Type column characteristic (110filetype) value to the filename extension (for example, “pdf”).
- Optionally, set the File Path column characteristic (110d_blob_p) value to the source path used to load the file (for example, `C:\edm_documents\datasheet_example.pdf`)
- Define an Object characteristic (110d_blob) value that is the leaf name from the file path (for example, `datasheet_example.pdf`).

3. Open an EBS command window and type the following command:



Note:

By default, the **xml-console** command commits each object as it imports. If you are loading a large number of documents (for example, over a hundred) consider including the `-transaction` switch to speed the import time.

```
xml-console -configname login_config -import -importfile import_file.xml -blobdir
document_directory -verbose
```

Results

The files are loaded as Document objects in the database. To verify that the new objects exist, choose **File > Refresh Data Model**, and then search the Documents class.

Examples

The following XML data file loads two files named `datasheet_example1.pdf` and `datasheet_example2.pdf` from the `C:\edm_documents` folder into the Document Manager catalog (NNDM) of the Documents object class (110).

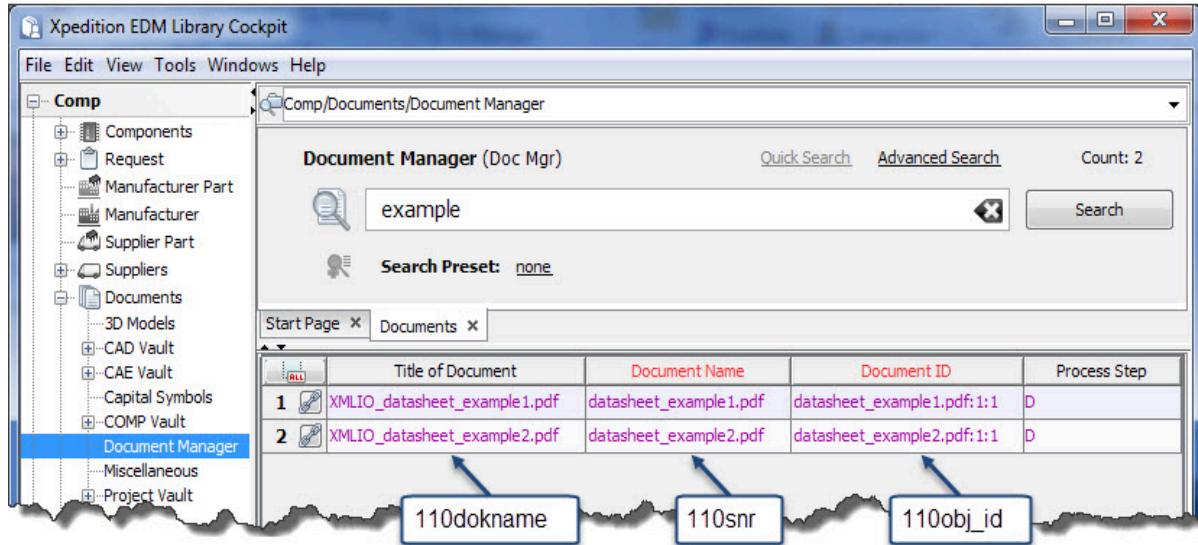
```

<?xml version="1.0" encoding="UTF-8"?>
<data>
  <date format="yyyy-MM-dd HH:mm:ss z" />
  <object objectid="" class="110" broken_ref="true">
    <!--Document Name-->
    <field id="110snr">datasheet_example1.pdf</field>
    <!--Catalog Group Key-->
    <field id="110obj_skn">NNDM</field>
    <!--Title of Document-->
    <field id="110dokname">XMLIO_datasheet_example1.pdf</field>
    <!--Long Description-->
    <field id="110dokbemerk">Loaded by XML utility</field>
    <!--File Info-->
    <list id="110doc_lst" clear="true">
      <!--Index-->
      <field id="110doc_idx">0</field>
      <!--File Type-->
      <field id="110filetype">pdf</field>
      <!--File Path-->
      <field
        id="110d_blob_p">C:\edm_documents\datasheet_example1.pdf</field>
      <!--Object-->
      <field id="110d_blob">datasheet_example1.pdf</field>
    </list>
  </object>
  <object objectid="" class="110" broken_ref="true">
    <!--Document Name-->
    <field id="110snr">datasheet_example2.pdf</field>
    <!--Catalog Group Key-->
    <field id="110obj_skn">NNDM</field>
    <!--Title of Document-->
    <field id="110dokname">XMLIO_datasheet_example2.pdf</field>
    <!--Long Description-->
    <field id="110dokbemerk">Loaded by XML utility</field>
    <!--File Info-->
    <list id="110doc_lst" clear="true">
      <!--Index-->
      <field id="110doc_idx">0</field>
      <!--File Type-->
      <field id="110filetype">pdf</field>
      <!--File Path-->
      <field
        id="110d_blob_p">C:\edm_documents\datasheet_example2.pdf</field>
      <!--Object-->
      <field id="110d_blob">datasheet_example2.pdf</field>
    </list>
  </object>
</data>

```

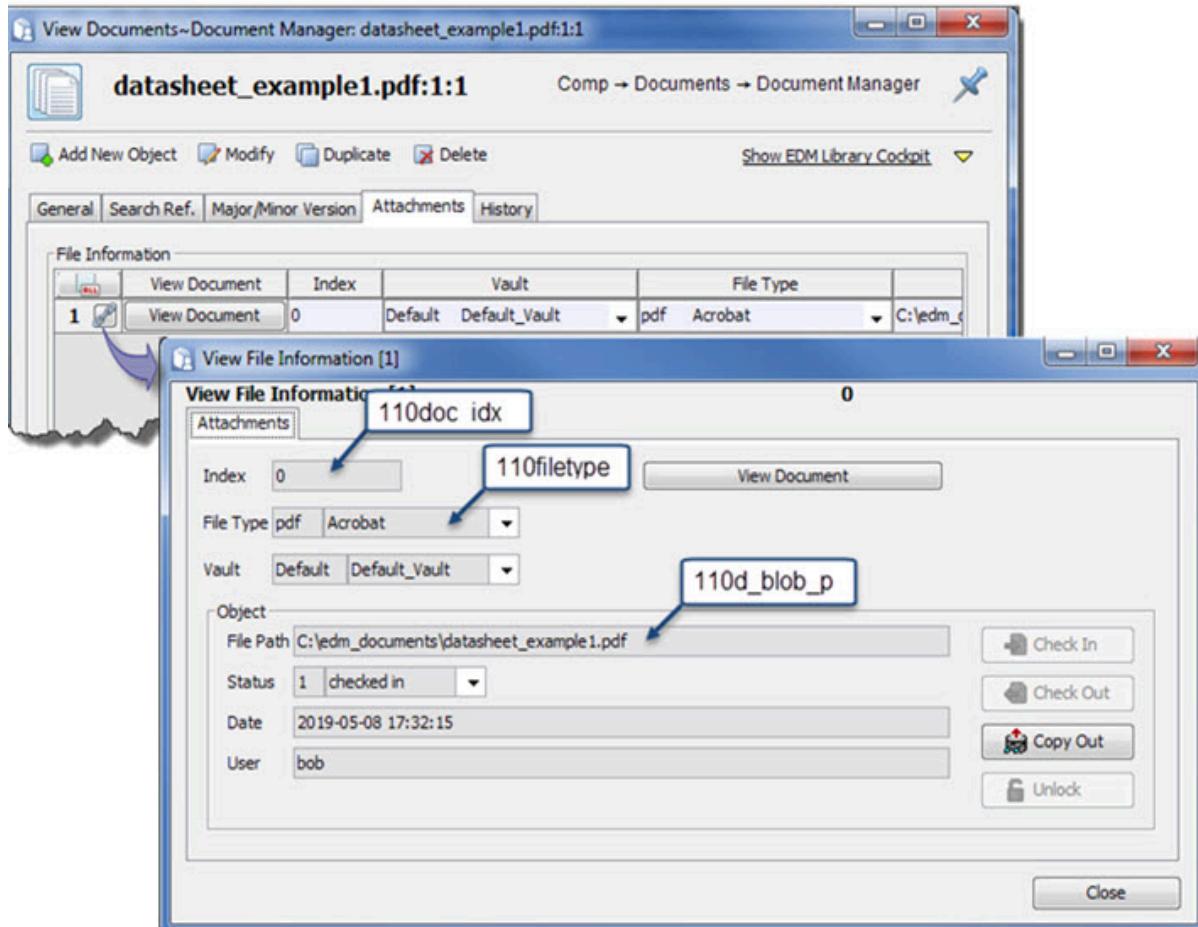
A search of the Document Manager catalog in the Documents class shows the two new objects in the search results list.

Figure 194. Search Results List Containing Imported Document Objects



Open the first document in the list and view the File Information on the **Attachments** tab to show the result of the <list></list> subsection definition for datasheet_example1.pdf in the XML data file.

Figure 195. Attachments Tab of the First Document Object



Related Topics

[XML Data File Format](#)

[xml-console Command](#)

[Associating Documents to Components](#)

Associating Documents to Components

You can use the **xml-console** command with an XML data file to associate a pre-existing document to a Component object.



Note:

Although this task shows how to create a Document object reference from a Component object, you can create a document reference from any class that has a **Documents** tab (for example, the Manufacturer Part, Variant BOM, or Audit class).

Prerequisites

- When building the XML data file, both the Component object and Document object must exist in the database, and you must know the key values of each object, including version information for the Document object.

Procedure

- Create an XML data file that describes the document references (refer to Examples).
- Open an EBS command window and type the following command:

```
xml-console -configname login_config -import -importfile import_file.xml -verbose
```

Results

The referenced Document object now displays in **Documents** tab of the Component object.

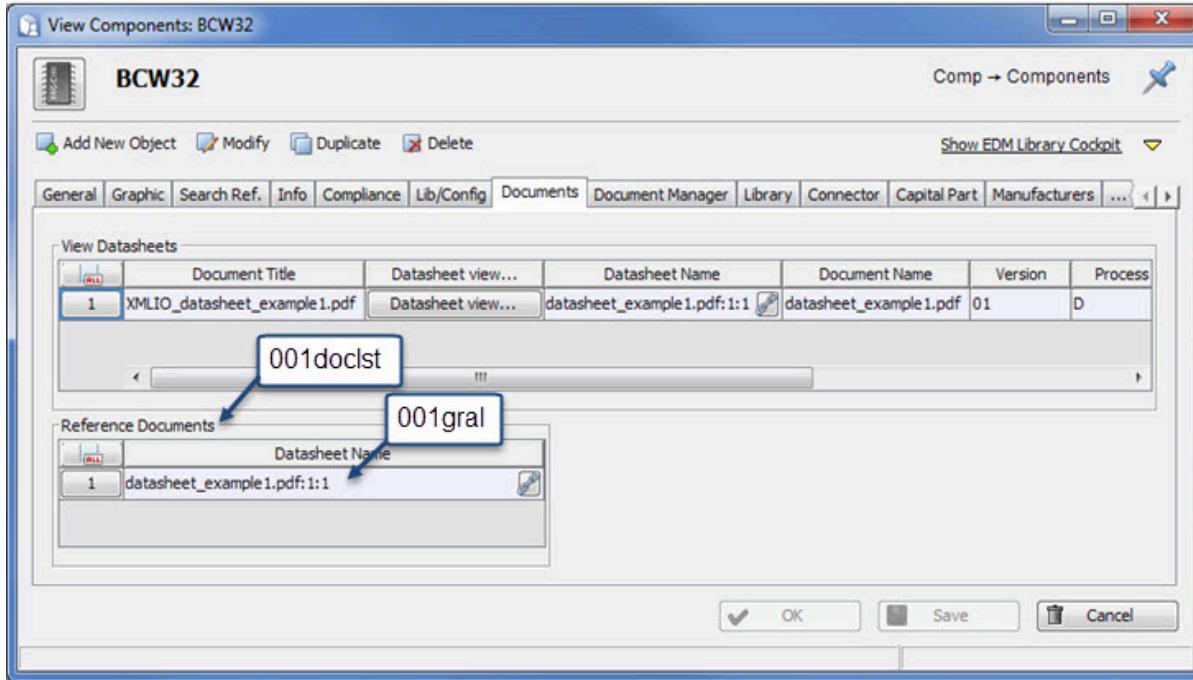
Examples

The following XML data file associates the *datasheet_example1.pdf* Document object to the BCW32 Component object.

```
<data>
<date format="yyyy-MM-dd HH:mm:ss z" />
<object objectid="BCW32" class="001" broken_ref="true">
    <!--Reference Documents-->
    <list id="001doclst" clear="true">
        <!--Datasheet Name-->
        <field id="001gral">datasheet_example1.pdf:1:1</field>
    </list>
</object>
</data>
```

[Figure 196](#) shows the resulting Document object reference created on the **Documents** tab of the BCW32 Component object.

Figure 196. Referenced Document on the Documents Tab of a Component Object



Related Topics

- [XML Data File Format](#)
- [xml-console Command](#)
- [Loading Documents in Bulk](#)

