System Administrator’s Guide
First Edition (January 2007)
This edition applies to version 6, release 2, modification 1 of IBM Maximo Enterprise Adapter and to all subsequent releases and modifications until otherwise indicated in new editions.

Third-Party Technology: Certain Maximo products contain technology provided under license from third parties, as noted in the following table:

<table>
<thead>
<tr>
<th>IBM Product</th>
<th>Third-Party Information</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Portions © 2005 BEA Systems, Inc. BEA WebLogic Server® provided by BEA Systems, Inc.</td>
</tr>
<tr>
<td>Maximo Discovery</td>
<td>MSDE Copyright © Microsoft® Corporation.</td>
</tr>
<tr>
<td>IBM Maximo Mobile</td>
<td>Portions © 2005 DataMirror, Inc.</td>
</tr>
</tbody>
</table>


© Copyright International Business Machines Corporation 2007. All rights reserved.

US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
# Contents

**About This Guide** .............................................. xi
  Audience ............................................................ xi
  Related Documentation ........................................... xi
  Support ............................................................. xii

**Section I: Using the Maximo Integration**

**Chapter 1: What is the Maximo Enterprise Adapter?** ............................................. 1-1

**Chapter 2: Architecture** ................................................. 2-1
  Overview ............................................................ 2-2
  Integration Object Layer ........................................... 2-5
    Integration Objects ............................................... 2-5
    Integration Points ............................................... 2-7
      Integration Point Properties .................................. 2-7
      Query and Response Integration Points ....................... 2-7
      Integration Point Processing Class .......................... 2-8
    Real Time Integration Events .................................... 2-8
  Interface Layer .................................................... 2-9
    Adapters .......................................................... 2-9
      Internal Type Adapters ......................................... 2-9
      External Type Adapters ....................................... 2-10
    Interfaces ......................................................... 2-10
      Interface Operation Property .................................. 2-12
      Query and Response Type Interfaces .......................... 2-12
      Interface Processing Class and User Exit Class .............. 2-12
    Data Formats ..................................................... 2-13
      Interface Data Format Conversion .............................. 2-13
    Interface Processing Rules ..................................... 2-14
    Interface Controls .............................................. 2-14
  External System Layer ............................................. 2-15
    External System Definition and Configuration .................. 2-15
    Interface Control Values ...................................... 2-16
    End Points ....................................................... 2-16
    Inbound and Outbound Queues ................................... 2-17
  Inbound and Outbound Communication ............................... 2-18
    Inbound Gateway Communication ................................ 2-18
    Outbound Handler Communication ................................ 2-19

**Chapter 3: Outbound and Inbound Processing** ......................................................... 3-1
  Outbound Integration Processing .................................. 3-2
    Initiating the Outbound Integration Process .................... 3-3
      Real Time Integration Processing ................................ 3-3
      Batch Integration Processing .................................. 3-4
    Building the Integration Object ................................ 3-5
    Applying Integration Point Processing .......................... 3-7
    Duplicating the Integration Object ................................ 3-8
    Applying Interface Processing Rules .............................. 3-10
    Applying User Exit Preprocessing ................................ 3-11

© Copyright IBM Corp. 2007
Chapter 4: Maximo XML and Schema ................................................................. 4-1
  Structure of Maximo XML ................................................................. 4-2
  Interface Element .................................................. 4-3
    Header Element .................................................. 4-4
      Header Attributes ............................................. 4-4
      Header Fields ............................................. 4-5
    Content Element ................................................. 4-6
      Standard Integration Objects .................................. 4-6
      Merged Integration Objects ................................... 4-7
    Action Attribute ..................................................... 4-8
      Add Action ..................................................... 4-9
      Delete Action ..................................................... 4-9
      Update Actions ..................................................... 4-9
      Comparison of the Change, Replace, and AddChange Actions ..................................................... 4-11
      Valid Action Attribute Combinations ..................................................... 4-14
    Field Attributes ..................................................... 4-15
      Changed Attribute ..................................................... 4-15
      Glorder Attribute ..................................................... 4-16
      Langenabled Attribute ..................................................... 4-17
      Maxvalue Attribute ..................................................... 4-17
    Additional Considerations ..................................................... 4-18
  XML Schemas for Maximo Interfaces ..................................................... 4-19
    Schema Content ..................................................... 4-19
    Key Fields ..................................................... 4-19
    Validation ..................................................... 4-19
    Schema Attribute ..................................................... 4-19
    Schema Directories ..................................................... 4-20
    Regeneration of the Schema Definition ..................................................... 4-20

Chapter 5: Interface Tables ........................................................................ 5-1
  Use of Interface Tables ................................................................. 5-2
  Location of Interface Tables ......................................................... 5-2
  Naming Convention for Interface Tables ..................................................... 5-2
# IBM Maximo Enterprise Adapter: System Administrator's Guide

## Chapter 7: Error Management
- Queue Error Handling .......................................................... 7-2
- Interface Table Error Handling ............................................ 7-3
- Error Management Configuration ........................................ 7-4
  - Integration Administration Setup Properties .................. 7-4
  - External Systems Configuration Properties .................. 7-5
  - Maximo.properties Configuration ................................ 7-5
- Error Notification .............................................................. 7-6
- Error Management Folders .................................................. 7-7
- Error Correction ............................................................... 7-9
  - Critical Errors ............................................................. 7-9
  - Common Causes of Errors .............................................. 7-9
  - Researching Errors ...................................................... 7-10
  - Updating Message Data ................................................. 7-12
  - Deleting Errors .......................................................... 7-13

## Section II: Advanced Topics

### Chapter 8: Integration Gateway
- Gateway Framework ........................................................... 8-1
  - Interpretation Layer ....................................................... 8-2
- Writing Your Own Interpreter ........................................... 8-3
- Bypassing Interpretation .................................................. 8-4
- Sending Transactions via EJB ............................................ 8-5
  - Security ........................................................................ 8-5
  - Response ...................................................................... 8-5
- Sending Transactions via HTTP/HTTPS .................................. 8-6
  - Security ........................................................................ 8-6
  - Response ...................................................................... 8-6
- Sending Transactions via the Gateway Web Service .............. 8-7
  - Security ........................................................................ 8-7
  - Response ...................................................................... 8-7

### Chapter 9: Router
- End Points ........................................................................ 9-1
  - Handlers ....................................................................... 9-2
    - EJB Handler ............................................................... 9-4
    - FLATFILE Handler ..................................................... 9-6
      - Flat File Naming Convention ................................ 9-6
      - Flat File Formatting ............................................... 9-7
      - Flat File Properties ................................................ 9-7
    - HTTP Handler ........................................................... 9-8
    - IFACETABLE Handler .................................................. 9-9
    - JMS Handler ............................................................. 9-10
    - WEBSERVICE Handler ............................................... 9-12
    - XMLFILE Handler ..................................................... 9-14
  - Writing Custom Handlers .................................................. 9-15

### Chapter 10: Advanced Interface Table Polling
- Cron Tasks ....................................................................... 10-1
  - Selectors ..................................................................... 10-2
  - Queue Tables ................................................................ 10-3

### Chapter 11: JMS Queue Configuration
- Sequential Queues ............................................................ 11-1
  - Continuous Queue ......................................................... 11-2
    - Considerations .......................................................... 11-3
    - Number of Beans ...................................................... 11-4
Chapter 12: Security ................................................................. 12-1
  Securing Integration Queues ............................................. 12-2
  Securing the Integration Gateway ..................................... 12-4
  Securing the Gateway Web Service ................................. 12-6
  Securing the Servlet (HTTP) ............................................. 12-7
  Securing Maximo Web Services ....................................... 12-9
  Securing the Remote Integration APIs (MicService) ............. 12-12
  Securing Interface Tables .............................................. 12-12
  Outbound Security ......................................................... 12-12

Chapter 13: Cluster Configuration ........................................ 13-1
  Cron Task Configuration .................................................. 13-2
  JMS Queues .................................................................. 13-2
  Global Directory ............................................................. 13-3
  Integration Gateway ......................................................... 13-3
  EJB ............................................................................. 13-3
  Servlet (HTTP) ............................................................... 13-3
  Gateway Web Service ....................................................... 13-3
  Maximo Web Services ....................................................... 13-4

Chapter 14: Customization with Processing Rules .................. 14-1
  MBOs and Sub-records .................................................... 14-2
  Defining a Processing Rule .............................................. 14-3
    Specifying the Sub-record or MBO ................................. 14-3
    Specifying when to Trigger the Processing Rule .............. 14-4
    Specifying the Processing Rule Action ............................ 14-4
      Message Processing Actions ....................................... 14-4
      Field Transformation Actions ...................................... 14-5
    Specifying the Processing Sequence ............................. 14-10
  Defining Conditions and Evaluations ............................... 14-10
    Specifying a Condition .............................................. 14-10
    Specifying the Evaluation Category ............................. 14-11
    Specifying the Field to Evaluate .................................. 14-13
    Specifying the Type of Evaluation ................................ 14-14
    Specifying when to Evaluate the Field .......................... 14-15
    Specifying the Comparison Field .................................. 14-16
  Interface Controls .......................................................... 14-18
    Control Levels ............................................................ 14-18
    Control Types ............................................................. 14-19
      Boolean Type Controls ............................................. 14-19
      List Type Controls .................................................... 14-19
      Value Type Controls .................................................. 14-19
      Cross-Reference Type Controls ................................... 14-19
    Creating New Controls ................................................ 14-20

Chapter 15: Customization with User Exits .......................... 15-1
  Outbound Customization ................................................ 15-2
  Integration Point Processing ........................................... 15-4
  Interface Processing Rules ............................................. 15-4
  User Exit Preprocessing ................................................ 15-4
  Interface Processing Class ............................................. 15-5
About This Guide

This section explains how this guide can help you to use IBM® Maximo®. It also provides information about other IBM Corporation resources available to you, such as additional documentation and support.

Audience

The guide is intended for the following people:

- Developers
- Implementation analysts
- Support personnel
- System administrators

Related Documentation

For more information about Maximo, refer to the following documentation:

<table>
<thead>
<tr>
<th>Document</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM Maximo Finance Manager's Guide</td>
<td>Describes how IBM Maximo completes financial transactions and how to set up general ledger accounts.</td>
</tr>
<tr>
<td>IBM Maximo Installation Guide</td>
<td>Describes how to install and configure the following software:</td>
</tr>
<tr>
<td></td>
<td>▼ Application server</td>
</tr>
<tr>
<td></td>
<td>▼ IBM Maximo</td>
</tr>
<tr>
<td></td>
<td>▼ Actuate®</td>
</tr>
<tr>
<td>IBM Maximo Multisite Administrator's Guide</td>
<td>Describes how to configure IBM Maximo for a Multisite implementation.</td>
</tr>
<tr>
<td>IBM Maximo Online Help</td>
<td>Provides step-by-step procedures for IBM Maximo applications.</td>
</tr>
<tr>
<td>IBM Maximo Reconciliation Module Implementation Guide</td>
<td>Describes how to use the IBM Maximo Reconciliation module to reconcile the two types of information that IBM Maximo maintains about information technology (IT) assets: IT asset data and deployed asset data.</td>
</tr>
<tr>
<td>Document</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><em>IBM Maximo Report Administration and Development Guide</em></td>
<td>Describes how to use Actuate to design and administer IBM Maximo reports.</td>
</tr>
<tr>
<td><em>IBM Maximo System Administrator's Guide</em></td>
<td>Describes database configuration, security, and other administrative level applications and tasks.</td>
</tr>
<tr>
<td><em>IBM Maximo User's Guide</em></td>
<td>Provides an overview of the Maximo end user applications. It also describes how the IBM Maximo applications interact with each other.</td>
</tr>
<tr>
<td><em>IBM Maximo Workflow Implementation Guide</em></td>
<td>Provides information about how to use IBM Maximo to plan, design, build, test, implement, and manage Workflow processes.</td>
</tr>
</tbody>
</table>

**Support**

IBM Maximo users with a valid Annual Customer Support Plan (ACSP) can obtain product support online at Support Online: support.mro.com.

Support Online includes information about product releases, software patches, and documentation updates. To find the most current version of a document, refer to the Knowledge Base on this site.
What is the Maximo Enterprise Adapter?

The Maximo Enterprise Adapter is a set of applications and predefined integration points that help you to integrate Maximo with your enterprise applications and create business flows between Maximo and your other enterprise applications.

Key Features of the Maximo Enterprise Adapter

The key features of the Maximo Enterprise Adapter are:

- Applications to manage integration processing and to create new integration points
- An application to create processing rules for customizing interfaces
- Support for multiple integration models using HTTP, messaging, and database interface tables and flat files
- Real time, batch, and user-initiated processing of outbound and inbound interfaces
- Support for customization of the predefined integration objects
- Provision for data transformation using XSL
- Load and performance scalability using multiple queues and/or multiple queue consumers
- Support for clustered environments that enable reduced downtime and increased availability and performance

The Enterprise Adapter includes the Maximo adapter, which contains a comprehensive set of predefined Maximo interfaces to enable integration with Maximo applications. All Maximo adapter interfaces are Web service ready by default. Users can easily create new interfaces and enable them for Web service without writing any code.

Key Features of the Maximo Adapter

The key features of the Maximo adapter are:

- A comprehensive set of predefined outbound and inbound integration points and interfaces
- Support for query and data synchronization interfaces
- Bulk export of all interfaces, with the ability to select data through a user-defined query
- The ability to create XML or flat files for outbound interfaces
- The ability to perform bulk loading of XML or flat files for inbound interfaces

- Dynamic XML schema generation for all integration objects and Maximo adapter interfaces

- Dynamic generation of WS-I compliant Web services for all Maximo adapter interfaces

Additional adapters can be quickly configured and deployed for enterprise connectivity with various systems. Each adapter can have its own interfaces and delivery mode. Preconfigured adapters for Oracle® and SAP® are available as add-ons.
This chapter describes the architecture of integration processing in Maximo.

Anyone involved in the implementation or day-to-day administration of the Maximo Enterprise Adapter applications should read this chapter. Familiarity with the concepts in this chapter is essential for understanding the remaining documentation and using the application.

This chapter contains the following sections:

- Overview
- Integration Object Layer
- Interface Layer
- External System Layer
- Inbound and Outbound Communication
Maximo Integration facilitates data exchange between Maximo and external applications or systems in a real time or batch mode. Data is exchanged via interfaces, each of which acts as a communication channel between the external system and one or more integration points (points of data exchange) in Maximo.

**Maximo Integration Overview**
Maximo Integration processing can be represented in three layers, as described in the following table:

**Integration Processing Layers**

<table>
<thead>
<tr>
<th>Integration Layer</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integration Object layer</td>
<td>Create and manage integration objects and integration points. Each integration object is built from one or more MBOs that provide the content needed for a specific integration point. Integration points provide a framework for accessing the MBOs in the integration object and any methods defined on the MBOs.</td>
</tr>
<tr>
<td>Interface layer</td>
<td>Create and manage interfaces, including business rules processing and data transformations. Interfaces implement one or more integration points in either direction (outbound or inbound) and each interface-integration point combination can have a processing class, a user exit class, and processing rules associated with it.</td>
</tr>
<tr>
<td>External System layer</td>
<td>Create and manage external systems and their interfaces. This includes defining external systems that are exchanging data with Maximo, identifying the specific interfaces applicable to each external system in either direction (outbound or inbound), setting up interface control values where applicable for each interface, and identifying queue parameters for the system and the communication method used for sending data to the system.</td>
</tr>
</tbody>
</table>

In addition to the three layers, the integration includes specific entities for outbound (Maximo to external system) and inbound (external system to Maximo) communication. These appear in the Inbound Gateway Communication and Outbound Handler Communication sections of the following diagram.
Overview

Maximo Integration Architecture

Inbound Gateway Communication
- EJB
- HTTP
- JMS
- Interface Tables
- XML Files
- Flat Files
- Web Service

Outbound Handler Communication

External System Layer
- External System Definition and Configuration
- Interface Control Values
- End Points
- Inbound and Outbound Queues

Interface Layer
- Query/Response Interfaces
- Data Synchronization Interfaces
- MAXIMO Adapter
- Oracle
- SAP
- Interface Processing Class
- User Exit Class
- Interface Processing Rules
- Interface Controls

Integration Object Layer
- Integration Objects
- Integration Points
- Integration Point Processing Class
- Real Time Integration Events

MAXIMO Business Objects
- MAXIMO Database Tables
Integration Object Layer

The integration object layer of Maximo interacts with the Maximo Business Objects and facilitates their creation and maintenance.

Maximo Enterprise Adapter Integration Object Layer

Integration Objects

An integration object consists of one or more sub-records that derive their content from a particular Maximo Business Object (MBO, or “may-bo”). An MBO is a functional unit of the Maximo application server that defines a set of fields and business rules and may update one or more Maximo database tables.

Each sub-record contains fields from a specific MBO. When you create an integration object, you specify the MBO(s) whose fields make up the object. You can then modify the integration object by excluding unneeded fields and adding user-defined fields. The name of the sub-record is the same as that of the corresponding MBO.

MBOs and sub-records are two distinct entities and it is important to distinguish between them.

MBOs and Sub-records

<table>
<thead>
<tr>
<th>Entity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MBO</td>
<td>The Maximo Business Objects that Maximo uses to directly update the database. During inbound integration processing, MBOs are created from the integration object sub-records. During outbound integration processing, integration object sub-records are created from MBOs.</td>
</tr>
<tr>
<td>Sub-record</td>
<td>A copy of a Maximo MBO that is included in an integration object. It may or may not include all the persistent and non-persistent fields in the original MBO. During outbound processing, Maximo populates the sub-record fields from the corresponding fields in the original MBO. An integration object consists of one or more sub-records.</td>
</tr>
</tbody>
</table>
Example

The predefined purchase order integration object, MXPO, contains sub-records that correspond to the MBOs in the following diagram.

**MXPO Integration Object**

If multiple sub-records are defined for an integration object, valid parent-child relationships must exist between the corresponding MBOs. These relationships are defined within the framework of the MBOs and are used to navigate between the various MBOs involved in defining a specific business entity.

Each MBO has persistent and non-persistent fields in its definition. A persistent field is one that maps directly to a database column in Maximo, and a non-persistent field is one that is included in the MBO definition but not stored in the database. By default, all persistent fields are included in the corresponding sub-record in an integration object, and all non-persistent fields are excluded. Users can optionally exclude persistent fields and include non-persistent ones. Users can also add user-defined fields to support data requirements beyond what is available from the MBOs.
Integration Points

An integration point provides access to an integration object in a given direction (inbound or outbound). Outbound integration points retrieve data from the MBOs to build the integration object, while inbound integration points create, update, or query MBOs, depending on the operation associated with the point. An integration object can have multiple integration points associated with it, in either direction.

Maximo provides predefined integration points for its predefined integration objects. Most integration objects have inbound and outbound integration points; some, such as the chart of accounts, general ledger components, and journal entries, have only one or the other.

Integration Point Properties

Integration points have two primary properties, direction and operation.

The direction property indicates the origin and destination of transactions that use the point, as follows:

**Outbound and Inbound Transactions**

<table>
<thead>
<tr>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outbound</td>
<td>The data transaction originates in Maximo and is sent to an external system or application.</td>
</tr>
<tr>
<td>Inbound</td>
<td>The data transaction originates in an external system or application and is sent to Maximo.</td>
</tr>
</tbody>
</table>

The operation property indicates the purpose of the integration point. Some operations are permitted only in specific directions:

**Operation Properties**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Purpose</th>
<th>Direction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Notify</td>
<td>The integration point performs data synchronization.</td>
<td>Outbound, Inbound</td>
</tr>
<tr>
<td>Query</td>
<td>The integration point processes queries.</td>
<td>Inbound only</td>
</tr>
<tr>
<td>Response</td>
<td>The integration point provides responses to queries.</td>
<td>Outbound only</td>
</tr>
</tbody>
</table>

Query and Response Integration Points

Maximo provides a framework for external systems or applications to query Maximo using integration objects and query type integration points; and to return the response to the query using response type integration points. Query integration points can be defined only in the inbound direction and response integration points can be defined only in the outbound direction.
Integration Object Layer

Integration Point Processing Class

An integration point processing class is a Sun™ Java™ class that provides access to the corresponding integration object and the MBOs associated with that object. Processing classes act in conjunction with the MBOs to facilitate the transfer of data to and from Maximo. Integration point processing classes are optional; an integration point can have one or no processing classes associated with it. Users can create custom integration point processing classes.

Inbound integration point processing classes are more common than outbound ones. They often filter data; invoke specific methods on the MBOs or otherwise preprocess data; and convert integration objects back into MBOs that can be processed by Maximo.

The predefined outbound integration points typically do not use a processing class. However, in cases where Maximo uses the same MBO for different business processes or applications, an outbound integration point processing class filters the data from the MBOs to ensure that only the relevant transactions are sent out via the integration point.

Example

The MXISSUE integration object uses the MATUSETRANS MBO, which handles issues, returns, direct receipts, and invoice variance transactions. Since the MXISSUEOUT integration point sends only issue and return transactions, it uses a processing class to filter out direct receipts and invoice variance transactions.

Real Time Integration Events

During the creation of an outbound integration point, an integration event listener is automatically registered on the primary MBO of the integration object. When the listener is enabled, it monitors Maximo for activity on the corresponding MBO. Whenever any instance of that MBO is created or updated, outbound integration processing is initiated for all interfaces tied to that point.
Interface Layer

The interface layer of the Maximo Integration exposes integration points to external systems and applications. Interfaces provide the ability for multiple external systems and applications to access inbound integration points, and outbound integration points to send data to multiple external systems and applications.

Maximo Enterprise Adapter Interface Layer

Adapters

All interfaces are defined within an adapter, which is a set of related interfaces, programs, mappings, and controls.

By default, all interfaces provided with Maximo are defined within the Maximo adapter. Predefined adapters for integration with major application providers like Oracle and SAP are available from IBM Corporation. Users can add new interfaces to an existing adapter and create new adapters, if necessary.

Adapters can be of an internal or external type, depending on the data format of the interfaces within the adapter.

Internal Type Adapters

Interfaces defined within internal type adapters derive their content definition directly from the corresponding integration objects; that is, there is no difference between the integration object data format and the interface data format.

Since the formats of the interface and integration object are the same, internal type adapters allow the use of interface tables (database tables) as well as XML messages to process interfaces defined within these adapters. Interface tables are one of the mechanisms available to Maximo users for processing inbound and outbound data. Their format mirrors that of corresponding interfaces and they are available only for interfaces that are defined within internal type adapters. For more information about interface tables, see Chapter 5, "Interface Tables," on page 5-1.

The Maximo adapter provided with Maximo Integration is an internal type adapter. By default, new adapters are created as internal type adapters.

NOTE

Maximo Web services are available only for interfaces defined within the Maximo adapter.
External Type Adapters

Interfaces defined within external type adapters use a data format different from that used by Maximo. Mapping between the external format and Maximo format is done using Java code or XSL style sheets. Interfaces defined for external type adapters cannot use interface tables to process data between Maximo and an external system.

Interfaces

Maximo uses interfaces to transform data from Maximo format to an external format, and vice versa; and to apply additional business rules to the data beyond the rules implemented in the inbound or outbound integration point associated with the interface.

For example, Maximo might need to send accounting transactions to a general ledger application, purchasing requisitions to a central purchasing application, and so on. Each external system or application uses different data formats and business rules. Application- and external system-specific interfaces process the transactions according to the formatting and processing requirements of each external system.

Maximo provides predefined application-specific interfaces and system interfaces. The latter are primarily for updating metadata in Maximo. For more information about system interfaces, see "System Interfaces" on page B-51.

If an integration object is specified for an interface, it restricts the integration points that can be associated with the interface, as follows:

An integration object must be defined for an interface within an internal type adapter. These interfaces use the content definition of the integration object that is associated with the interface.

Integration objects cannot be specified for interfaces within an external type adapter. The format of the interface is not based on an integration object. A Java class or XSL style sheet must be used to create the necessary format.
**Interfaces in Internal Type Adapters**

<table>
<thead>
<tr>
<th>Direction</th>
<th>Restrictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outbound</td>
<td>Only one integration point can be specified for the interface. The integration point must be defined on the integration object associated with the interface.</td>
</tr>
<tr>
<td>Inbound</td>
<td>Multiple inbound integration points can be specified for the interface, and they do not need to be defined for the integration object associated with the interface. If multiple inbound integration points are specified, they must belong to different integration objects; that is, for a given interface, there cannot be two inbound integration points specified for the same integration object.</td>
</tr>
</tbody>
</table>

**Interfaces in External Type Adapters**

<table>
<thead>
<tr>
<th>Direction</th>
<th>Restrictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outbound</td>
<td>Multiple integration points can be specified for an interface. If multiple points are specified, they must belong to different integration objects; that is, for a given interface, there cannot be two inbound integration points specified for the same integration object.</td>
</tr>
<tr>
<td>Inbound</td>
<td>Same as outbound.</td>
</tr>
</tbody>
</table>

**Maximo Enterprise Adapter Purchase Order Components**

![Diagram of Maximo Enterprise Adapter Purchase Order Components]
Interface Layer

**Interface Operation Property**

Like integration points, every interface has an operation property (Notify, Query, or Response), which indicates the purpose of the interface. An operation must be specified for an interface before integration points are associated with the interface.

The operation specified for the interface restricts the integration points that can be associated with the interface. Only integration points with the same operation property can be associated with an interface; that is, notify type interfaces can be associated only with notify type integration points; query type interfaces with query type integration points; and response type interfaces with response type integration points.

**Query and Response Type Interfaces**

Query and response operations are available only for interfaces defined within the Maximo adapter. They expose the Query and response type integration points defined on an integration object and allow a user to query Maximo for data using XML documents. The Query and response type interfaces are available only via Maximo Web services.

**Interface Processing Class and User Exit Class**

Like integration point processing classes, interface processing and user exit classes are Java classes that contain processing logic. An interface maps to one or more integration points in either direction; for each interface-integration point combination, you can optionally specify one interface processing class and one user exit class.

The Maximo adapter does not implement any interface processing classes. The ERP adapters provided by IBM Corporation reserve the interface processing classes for predefined, interface-specific processing that includes data format conversion and the application of business rules.

Users who wish to customize predefined interfaces can use the user exit class for that purpose. A user exit class executes before and after any interface processing class, thereby overriding or supplementing the logic in the corresponding interface processing class.

**NOTE** Since the Maximo adapter does not use the interface processing classes, users can identify a custom class as an interface processing class. For the sake of standardization, however, you might prefer to identify all custom classes as user exit classes.

For more information about customizing and determining whether to add custom code as an integration processing class, an interface processing class, or a user exit class, see Chapter 15, "Customization with User Exits," on page 15-1.
Interface Layer

Data Formats

Maximo and external systems can exchange data transactions via XML messages, interface tables, and flat files. Maximo accepts XML and interface table transactions for the real-time exchange of data and flat files for the bulk import and export of data.

Maximo XML

Maximo XML is the XML representation that the MBOs recognize. Maximo writes all outbound XML messages in this format and requires that all inbound non-Maximo XML messages be converted to this format. If an external system uses another XML format, you must provide Java code or XSL style sheets to convert the data.

Interface Tables

Interface tables are relational database tables that you can use in place of XML messages to transfer data between Maximo and external systems. Each table contains the same data fields as the corresponding Maximo XML interface, in a flat, non-hierarchical format.

NOTE

The use of interface tables is available only with internal type adapters.

For more information, see Chapter 5, "Interface Tables," on page 5-1.

Flat Files

A flat file is a non-hierarchical, non-relational representation of the data columns in a Maximo interface or interface table. You can use flat files to load initial master data into Maximo, and to perform the periodic import and export of non-system data to and from Maximo.

Interface Data Format Conversion

In predefined interfaces, data format conversion occurs in the integration point processing class associated with the interface. Users can override or supplement this code by writing a user exit class or an XSL style sheet and associating it with the interface-integration point combination.

All interfaces defined within external type adapters use a data format different from that used by Maximo, thereby requiring a user exit class or XSL style sheet to perform the conversion between the Maximo format (as defined in the integration objects) and the external format.
Interface Processing Rules

Interface processing rules let users change the behavior of any interface without having to write Java classes. Processing rules can access and evaluate values in XML and MBO fields, MBO sets, and controls; and they can change the values in XML and MBO fields, or stop or skip processing of all or part of a transaction.

Processing rules can apply to any interface, regardless of the adapter in which the interface is created. However, data referenced by a rule must be in Maximo format, so users must first transform any data in inbound interfaces associated with an external type adapter.

Inbound and outbound processing of the same interface require separate sets of processing rules. Like interface processing classes and user exit classes, processing rules are defined for an integration point-interface combination. In many cases, you can use either a processing rule or a processing class to achieve the same result.

For more information, see Chapter 14, "Customization with Processing Rules," on page 14-1.

Interface Controls

Interface controls give users the ability to override the behavior of certain predefined interface processing and to configure interfaces according to the requirements of individual organizations and sites. Processing rules can access interface controls, as can Java classes.

Each adapter has its own set of controls, which are associated with interfaces within that adapter. Each external system that uses the adapter has a configurable copy of the controls. Two external systems that process the same interface can share the same processing logic, class files and processing rules, yet process the data differently due to different control settings. For example, two systems using the same purchase requisition (PR) outbound interface may use different control settings that direct one system to send the PR out on status APPR (approved) and the other on status CLOSE.

Maximo provides four types of interface controls.

Interface Control Types

<table>
<thead>
<tr>
<th>Control Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Specifies a value of 0 (false) or 1 (true)</td>
</tr>
<tr>
<td>Cross-reference</td>
<td>Replaces one value with another</td>
</tr>
<tr>
<td>List</td>
<td>Specifies a list of values</td>
</tr>
<tr>
<td>Value</td>
<td>Specifies a single value</td>
</tr>
</tbody>
</table>

For more information, see Chapter 14, "Customization with Processing Rules," on page 14-1.
External System Layer

Any business application that sends data to Maximo or receives data from Maximo is considered to be an external system. The external system layer enables the flow of data between Maximo and external systems by defining the location and characteristics of external systems, and identifying the adapter and interfaces that each external system uses.

*Maximo Enterprise Adapter External System Layer*

External System Definition and Configuration

An external system interacts with Maximo, either as an end point (location) to which Maximo sends outbound data, or as a source from which Maximo receives inbound data. An external system can process inbound interfaces, outbound interfaces, or a combination of both. Maximo can be integrated with any number of external systems. Each external system uses a single adapter to exchange messages with Maximo in the format identified by the adapter, and a single end point to process outbound messages.

*Maximo Enterprise Adapter External System Structure*

The name given to the external system is the name by which Maximo recognizes the external system. Inbound XML messages and interface table transactions must provide this name in the SenderID field, and Maximo writes this name in the RecipientID field of outbound XML and interface tables.
In outbound transactions, the SenderID is the value of MAXVARS.MXSYSID.

The Maximo adapter provides a predefined external system, EXTSYS1, which you can duplicate and modify based on your needs.

**CAUTION** Do not modify external system EXTSYS1 in any way.

### Interface Control Values

When you specify an adapter for an external system, Maximo includes in that system all the interface controls defined for the corresponding adapter. The external system inherits any default, system-level values for the controls.

You can change the default value of the controls or add new values if applicable. If the definition of a control allows organization or site overrides, you also can specify values for specific organizations or sites.

Configure only those controls that will be used by the external system. This will depend on the interfaces, processing rules, and Java classes that the system will process.

### End Points

An end point is a location to which an outbound queue sends data. End points are independent of adapters and external systems, although two systems that use different adapters generally have different end points. An end point typically has an application component that processes the data sent from Maximo.

**NOTE** With the exception of interface tables, end points apply only to outbound transactions.

External systems can be associated with end points in multiple ways; exact usage will depend on your implementation. Possible scenarios include, but are not limited to, the following:

- **A single end point per external system**
  
  This is the common point-to-point scenario where each individual external system has a different end point.

- **A single end point supporting multiple external systems**
  
  Multiple external system use the same set of interface tables, or multiple external systems share a single queue. In this case, you configure Maximo to have multiple systems that use the same end point, and the end point will contain multiple instances of any outbound message (one per external system).

Each end point is associated with a handler, which is a processing class that defines how, and in what format, Maximo sends data from the outbound queue to an end point. For more information, see "Inbound and Outbound Communication," on page 2-18.
Maximo provides the following predefined end points. You can create additional ones if necessary.

**Predefined End Points**

<table>
<thead>
<tr>
<th>End Point</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MXFLATFILE</td>
<td>Writes flat file data, in the form of rows and columns, to a prespecified directory location.</td>
</tr>
<tr>
<td>MXIFACETABLE</td>
<td>Writes data to an interface table (in a relational database) in a prespecified directory location.</td>
</tr>
<tr>
<td>MXXMLFILE</td>
<td>Writes data in XML file format.</td>
</tr>
</tbody>
</table>

**Inbound and Outbound Queues**

A queue is a Java™ Message Service (JMS) queue, which Maximo uses as a staging area during the exchange of messages between Maximo and external systems. JMS queue functionality is available within the BEA® WebLogic® and IBM WebSphere® application servers.

Maximo uses one queue for outbound processing and two for inbound processing. The inbound queues differ in the sequence in which they process transactions, as follows:

- The sequential inbound queue processes transactions in strict FIFO order, and stops processing when it encounters an error in a transaction. Use this queue to process interfaces that are dependent upon the successful processing of previous interfaces.

- The continuous inbound queue does not process in FIFO order and continues processing transactions after it encounters an error in a transaction. Use this queue to load data that is not dependent upon the successful processing of other interfaces.

Within an external system, you can use the sequential queue for some interfaces and the continuous queue for others, or you can use one queue for all inbound interfaces.

Maximo provides the following predefined queues. You can create additional ones if necessary.

**Predefined Queues**

<table>
<thead>
<tr>
<th>Queue</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cqn</td>
<td>Continuous inbound</td>
</tr>
<tr>
<td>sqin</td>
<td>Sequential inbound</td>
</tr>
<tr>
<td>sqout</td>
<td>Sequential outbound</td>
</tr>
</tbody>
</table>

For more information, see Chapter 11, "JMS Queue Configuration," on page 11-1.
Inbound and Outbound Communication

This section describes the ways in which the Integration can send transactions to, and receive transactions from, external systems.

Maximo Enterprise Adapter Inbound and Outbound Communication

<table>
<thead>
<tr>
<th>Inbound Gateway Communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>EJB</td>
</tr>
</tbody>
</table>

| Outbound Handler Communication |

Inbound Gateway Communication

The Maximo integration gateway provides the framework for receiving and queuing XML transactions from an external system. The gateway processes transactions received via Enterprise JavaBeans™ (EJB), HTTP/HTTPS, and the gateway Web service. It determines if Maximo should process the message, writes the message to the inbound queue, and notifies the external system when the message is successfully processed. For more information, see Chapter 8, "Integration Gateway," on page 8-1.

Maximo can process inbound transactions via flat files, XML files, and interface tables.
Outbound Handler Communication

Maximo sends data from the outbound queue to an end point via a handler, which is a processing class that defines how, and in what format, the data is to be delivered. Most handlers use a set of properties, such as a specific URL, a user name and password, or a specific directory location. The values of these properties depends on the end point associated with the handler.

Maximo provides the following predefined handlers. You can create additional ones if necessary.

**Predefined handlers**

<table>
<thead>
<tr>
<th>Handler</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EJB</td>
<td>Delivers outbound data to an Enterprise JavaBean (EJB) executing in the local application server or a remote application server</td>
</tr>
<tr>
<td>FLATFILE</td>
<td>Delivers outbound data into a flat file whose location is configurable</td>
</tr>
<tr>
<td>HTTP</td>
<td>Delivers outbound data as an XML document to a URL over the HTTP or HTTPS protocols</td>
</tr>
<tr>
<td>IFACETABLE</td>
<td>Delivers outbound data into interface tables in a relational database</td>
</tr>
<tr>
<td>JMS</td>
<td>Delivers outbound data into a queuing system that has been enabled through Java Message Service (JMS)</td>
</tr>
<tr>
<td>WEBSERVICE</td>
<td>Delivers outbound data to a Maximo Web services component using SOAP over HTTP</td>
</tr>
<tr>
<td>XMLFILE</td>
<td>Delivers outbound data in XML format to a file in the local machine or a shared network folder</td>
</tr>
</tbody>
</table>

For more information, see Chapter 9, "Router," on page 9-1.
This chapter describes the processing of outbound and inbound transactions. Anyone involved in the implementation or customization of the integration with Maximo should read this chapter.

This chapter contains the following sections:

- Outbound Integration Processing
- Inbound Integration Processing
Maximo performs two types of outbound integration processing:

- Real time, initiated via data entry in Maximo
- Batch, initiated via the Data Export feature of Maximo

The following list is an overview of the outbound processing activities. Not every activity applies to every outbound transaction. The diagrams on the following pages show the same activities in a visual format.

- Initiating the Outbound Integration Process
- Building the Integration Object
- Applying Integration Point Processing
- Duplicating the Integration Object
- Applying Interface Processing Rules
- Applying User Exit Preprocessing
- Applying Interface Class Processing
- Applying User Exit Postprocessing
- Applying XSL Mapping
- Sending the Interface to the External System

The following prerequisites apply to all outbound integrations:

- All applicable integration objects, integration points, interfaces, and external systems must be completely defined.
- The external system(s) must be configured with an end point.
- The following entities must be enabled:
  - External system(s)
  - Applicable interfaces
  - Applicable outbound event listeners
  - The cron task that polls the outbound queue

For information about performing the prerequisite activities, see Chapter 6, "Basic Configuration," on page 6-1.
Initiating the Outbound Integration Process

Summary

A Maximo user initiates the outbound integration process. This can be a background activity that is initiated when a user completes a transaction in Maximo (real time processing) or the initiation of the Data Export feature from the External Systems application (batch processing).

The Data Export feature lets you export the result set of a query from Maximo to an external system. The data export process typically uses different interfaces than those used for real time integration. Some processing in real-time interfaces involves changing field values or stopping the user from performing an activity. Neither of these activities applies in a data export mode.

*Maximo Enterprise Adapter Outbound Processing Activities*

Real Time Integration Processing

1. The user completes a transaction in Maximo. This is the only step in the outbound integration process that is visible to the user.

2. The primary MBO associated with the transaction identifies the related integration point(s) that have the listener enabled. The outbound integration process is initiated for all these integration points with enabled listeners.

**Tip** To find out if the event listener is enabled for a particular integration point, go to Enable/Disable Integration Events on the Select Action menu in the External Systems application.
Outbound Integration Processing

Batch Integration Processing

1. In the External Systems application, the user selects the external system to which data is to be sent.

2. On the Outbound Interfaces tab, the user selects an interface, then selects the Data Export feature.

3. The user selects the records to be exported by creating a query using the Maximo query-by-example (QBE) mechanism. The query must act on the primary MBO in the integration object.

   **Tip** To find out the name of the primary MBO in an integration object, look on the Integration Object tab in the Integration Objects application.

   If the Merged Object? check box in the top right of the window is not selected, the first MBO in the Source MBOs table window is the primary MBO.

   If the Merged Object? check box is selected, you cannot export this interface.

4. If the query returns a valid result set, the data export process continues.
Building the Integration Object

Summary

The integration framework builds an integration object from the MBO(s).

Maximo Enterprise Adapter Outbound Processing Activities

1. The integration object associated with each integration point is determined.

2. The integration object identifies its component MBO(s).

3. The integration framework constructs the integration object from the MBOs.

4. The integration framework adds the following header information to the integration object:

<table>
<thead>
<tr>
<th>Tag</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Integration Object Name&gt;</td>
<td>The name of the integration point (temporarily)</td>
</tr>
<tr>
<td>&lt;SenderID&gt;</td>
<td>The identifier of the Maximo application server (the value of MAXVARS.MXSYSID)</td>
</tr>
<tr>
<td>&lt;MessageID&gt;</td>
<td>Null (temporarily)</td>
</tr>
<tr>
<td>&lt;CreationDateTime&gt;</td>
<td>System DateTime</td>
</tr>
<tr>
<td>&lt;RecipientID&gt;</td>
<td>Null (temporarily)</td>
</tr>
</tbody>
</table>
Outbound Integration Processing

Output

The output of this activity is an integration object with the following format and header data:

```xml
<Integration Object Name>
  <Header>
    <SenderID>Value of MAXVAR.MXSYSID</SenderID>
    <MessageID></MessageID>
    <CreationDateTime>System DateTime</CreationDateTime>
    <RecipientID></RecipientID>
  </Header>
  <Content>
    <IntegrationObject>
      .
      .
      .
    </IntegrationObject>
  </Content>
</Integration Object Name>
```
Applying Integration Point Processing

Summary

The integration framework applies predefined processing logic (if specified) to the integration object.

*Maximo Enterprise Adapter Outbound Processing Activities*

The predefined integration point processing classes verify and filter integration objects, to ensure that their content meets the definition of the integration point. These predefined classes do not change or otherwise edit the data in an integration object. Not all predefined integration points have a processing class.

**Tip**

To find out if an outbound integration point processing class is associated with the integration point, look on the Integration Point tab in the Integration Objects application.

Output

The possible outcomes of this stage are as follows.

*Processing Outcome*

<table>
<thead>
<tr>
<th>If:</th>
<th>Output is . . .</th>
</tr>
</thead>
<tbody>
<tr>
<td>No processing class exists</td>
<td>The existing integration object, unchanged</td>
</tr>
<tr>
<td>The processing class skips the record (due to non-applicable data)</td>
<td>No output, and processing ends</td>
</tr>
<tr>
<td>The processing class stops the record (due to an error)</td>
<td>No output, processing ends, and an exception displays on the user interface screen</td>
</tr>
<tr>
<td>The processing class completes successfully</td>
<td>The existing integration object, possibly updated; in the case of predefined outbound integration points, the user-defined columns are sometimes updated</td>
</tr>
</tbody>
</table>
Duplicating the Integration Object

Summary

If the integration point is associated with multiple interfaces or external systems, the integration framework creates copies of the integration object.

Maximo Enterprise Adapter Outbound Processing Activities

1. The integration framework creates a copy of the integration object for every combination of interface and external system associated with the integration point.

2. The integration framework updates the header information in each integration object, as follows

Processing Outcome.

<table>
<thead>
<tr>
<th>Tag</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;MessageID&gt;</td>
<td>Unique identifier assigned by Maximo for each combination of interface and external system</td>
</tr>
<tr>
<td>&lt;RecipientID&gt;</td>
<td>The name of the external system that will receive the transaction</td>
</tr>
</tbody>
</table>
Output

The outcome of this activity is one integration object per combination of interface and external system, with the following format and header data:

```xml
<Integration Object Name>
  <Header>
    <SenderID>Value of MAXVAR.MXSYSID</SenderID>
    <MessageID>MessageID per external system - outbound interface</MessageID>
    <CreationDateTime>System DateTime</CreationDateTime>
    <RecipientID>External System Name</RecipientID>
  </Header>
  <Content>
    <IntegrationObject>
      .
      .
      .
    </IntegrationObject>
  </Content>
</Integration Object Name>
```

From this point on, the actions listed in each stage apply to each copy of the integration object that is created, in the preceding step, for a specific external system-interface combination.

If the outbound processing for an external system-interface determines that a duplicated integration object should be skipped due to inapplicable data, the skip action applies only to that copy of the integration object; that is, to that specific external system-interface. If the processing determines that a duplicated integration object should be stopped due to an error in the data, the stop action applies to all the copies of the integration object created in this step.
Applying Interface Processing Rules

Summary

The integration framework applies the processing rules (if specified) to the integration object.

Maximo Enterprise Adapter Outbound Processing Activities

Interface processing rules define the conditions under which Maximo can skip or stop a transaction, or change the data in an integration object. If any exist, the integration framework applies them to the integration object in the order specified by the processing rule sequence number.

Tip

To find out if processing rules exist for the integration object, look on the Outbound Processing Rules tab in the Integration Interfaces application.

Output

The possible outcomes of this activity are as follows

<table>
<thead>
<tr>
<th>Processing Outcome</th>
<th>Output is . . .</th>
</tr>
</thead>
<tbody>
<tr>
<td>No processing rules exist</td>
<td>The existing integration object, unchanged</td>
</tr>
<tr>
<td>A processing rule skips the record</td>
<td>No output, and processing ends</td>
</tr>
<tr>
<td>A processing rule stops the record</td>
<td>No output, processing ends, and an exception displays on the user interface screen</td>
</tr>
<tr>
<td>The processing rules complete successfully</td>
<td>The existing integration object, possibly updated</td>
</tr>
</tbody>
</table>
Applying User Exit Preprocessing

Summary

The integration framework applies the custom processing logic in the user exit class (if specified) to the integration object.

**Maximo Enterprise Adapter Outbound Processing Activities**

A preprocessing method in a user exit class allows manipulation of the integration object before any predefined processing takes place on the interface. Users typically use this functionality to customize predefined interfaces.

The predefined adapters do not provide any user exit classes.

**Tip**

To find out if preprocessing customization exists, look at the Interface tab in the Integration Interfaces application. If the Outbound Integration Points subtab shows a user exit class, check that class file for a preprocessing method.

**Note**

The same processing class contains the user exit preprocessing and user exit postprocessing (see page 3-13) methods.

Output

The possible outcomes of this stage are as follows.

**Processing Outcome**

<table>
<thead>
<tr>
<th>If:</th>
<th>Output is . . .</th>
</tr>
</thead>
<tbody>
<tr>
<td>No preprocessing method exists</td>
<td>The existing integration object, unchanged</td>
</tr>
<tr>
<td>The preprocessing method skips the record</td>
<td>No output, and processing ends</td>
</tr>
<tr>
<td>The preprocessing method stops the record</td>
<td>No output, processing ends, and an exception displays on the user interface screen</td>
</tr>
<tr>
<td>The preprocessing method completes successfully</td>
<td>The existing integration object, possibly updated</td>
</tr>
</tbody>
</table>
Applying Interface Class Processing

**Summary**

The integration framework applies predefined interface processing logic in the interface processing class (if specified) to the integration object.

**Maximo Enterprise Adapter Outbound Processing Activities**

An interface processing class typically implements additional processing logic and converts data from integration object to interface format.

The Maximo adapter does not provide any predefined interface processing classes. The XML format for interfaces defined within the Maximo adapter is comparable to the format of the integration object, as no XML mapping takes place.

**Tip**

To find out if an interface processing class exists for an interface, go to the Interface tab in the Integration Interfaces application. Look on the Outbound Integration Points subtab for an Interface Processing class.

**Output**

The possible outcomes of this activity are as follows.

**Processing Outcome**

<table>
<thead>
<tr>
<th>If:</th>
<th>Output is . . .</th>
</tr>
</thead>
<tbody>
<tr>
<td>No interface processing class exists</td>
<td>The existing integration object, unchanged</td>
</tr>
<tr>
<td>The interface processing class skips the record</td>
<td>No output, and processing ends</td>
</tr>
<tr>
<td>The interface processing class stops the record</td>
<td>No output, processing ends, and an exception displays on the user interface screen</td>
</tr>
<tr>
<td>The interface processing class completes successfully, without mapping</td>
<td>The existing integration object, possibly updated</td>
</tr>
<tr>
<td>The interface processing class completes successfully, with mapping</td>
<td>The interface</td>
</tr>
</tbody>
</table>
Applying User Exit Postprocessing

Summary
The integration framework applies custom processing logic in the user exit class (if specified) to the interface created in the previous stage.

Maximo Enterprise Adapter Outbound Processing Activities

This option is typically used to customize a predefined interface after interface processing takes place. Both the input integration object and the interface created from that integration object are available at this point.

The predefined adapters do not provide any user exit classes.

TIP
To find out if postprocessing customization exists, look at the Interface tab in the Integration Interfaces application. If the Outbound Integration Points subtab shows a user exit class, check that class file for a postprocessing method.

NOTE
The same processing class contains the user exit preprocessing (see page 3-11) user exit postprocessing methods.

Output
The possible outcomes of this stage are as follows

Processing Outcome.

<table>
<thead>
<tr>
<th>If:</th>
<th>Output is . . .</th>
</tr>
</thead>
<tbody>
<tr>
<td>No postprocessing method exists</td>
<td>The existing integration object or interface, unchanged</td>
</tr>
<tr>
<td>The postprocessing method skips the record</td>
<td>No output, and processing ends</td>
</tr>
<tr>
<td>The postprocessing method stops the record</td>
<td>No output, processing ends, and an exception displays on the user interface screen</td>
</tr>
<tr>
<td>The postprocessing method completes successfully</td>
<td>The existing interface, possibly updated</td>
</tr>
</tbody>
</table>
Applying XSL Mapping

Summary
The integration framework applies any custom mapping to the interface.

*Maximo Enterprise Adapter Outbound Processing Activities*

XSL mapping lets users map user-defined interfaces or customize the mapping done by predefined interface processing classes. Both the integration object and interface are available at this point.

**Tip**
To find out if an XSL map exists for an interface, go to the Interface tab in the Integration Interfaces application. Look on the Outbound Integration Points subtab for an XSL map.

Output
The output of this activity must be an XML message in interface format.
Sending the Interface to the External System

Summary

The integration framework writes the interface to the outbound queue, then sends it to the external system.

*Maximo Enterprise Adapter Outbound Processing Activities*

1. The integration framework writes the XML message to the outbound queue specified for the external system.

   **TIP** To find out which outbound queue an external system uses, look on the System tab in the External Systems application.

2. The cron task that polls the outbound queue picks up the message.

3. The cron task passes the message to a message router. The router uses the end point associated with the external system to identify and invoke the correct handler.

   **TIP** To find out which end point a system uses, look on the System tab in the External Systems application.

   To find out which handler an end point uses, select **Add/Modify End Points** from the Select Action menu in the External Systems application.

4. The processing class associated with the handler sends the data to the external system.

   If an error occurs, the message remains in the outbound queue until the next processing run of the cron task. For more information, see Chapter 7, "Error Management," on page 7-1.
Maximo performs two types of inbound integration processing:

- Queue-based, initiated via interface tables, the integration gateway or the Data Import feature of Maximo
- Synchronous, initiated via Maximo Web services

Data synchronization (operation = Notify) interfaces can be processed synchronously or via the queue. Query and response type interfaces can only be processed synchronously (via Maximo Web services).

The following list is an overview of the inbound processing activities. Not every activity applies to every inbound transaction. The diagrams on the following pages show the same activities in a visual format.

- Initiating the Inbound Integration Process
- Writing Messages to the Inbound Queues
- Retrieving Messages from the Inbound Queues
- Identifying the Integration Point
- Applying User Exit Preprocessing
- Applying Interface Class Processing
- Applying User Exit Postprocessing
- Applying XSL Mapping
- Duplicating the Integration Object
- Applying Integration Object Processing Rules
- Building the MBOs
- Applying MBO Processing Rules
- Applying the Integration Point Processing Class
- Applying User Exit MBO Processing
- Applying MBO Processing

For details about these activities, see the following sections of this chapter:

- “Queue-based Inbound Processing,” on page 3-17
- “Web Services Processing,” on page 3-36
Queue-based Inbound Processing

This section describes the processing that occurs when an external system sends a message to Maximo using one of the following mechanisms:

- Data Import (file loading) of XML or flat files
- Interface tables
- XML via the integration gateway (via HTTP post, EJB invocation, or gateway Web service invocation)

**NOTE** Only interfaces defined within internal adapters can use the Data Import feature and interface tables.

The following prerequisites apply to asynchronous inbound processing:

- All applicable integration objects, integration points, interfaces, and external systems must be completely defined.
- The following entities must be enabled:
  - External system(s)
  - Applicable interfaces
  - The cron task that polls the inbound sequential queue (if applicable)
  - The cron task that polls the interface tables (if applicable)

For information about performing the prerequisite activities, see Chapter 6, "Basic Configuration," on page 6-1.
Initiating the Inbound Integration Process

Summary

An external system can send asynchronous messages to Maximo in the following ways:

<table>
<thead>
<tr>
<th>Inbound processing initiation via:</th>
<th>Available for:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gateway using HTTP Post, EJB Invocation, gateway Web service invocation</td>
<td>All external systems</td>
</tr>
<tr>
<td>Data Import (file loading) feature, using XML or flat files</td>
<td>External systems using an internal type adapter</td>
</tr>
<tr>
<td>Interface tables</td>
<td>External systems using an internal type adapter</td>
</tr>
</tbody>
</table>

Maximo Enterprise Adapter Inbound Processing Activities

[Diagram of processing activities]

- Apply MBO Processing
- Apply Interface Class Processing
- Apply XSL Mapping
- Apply User Exit Postprocessing
- Gateway
- Interface Loading
- Interface Tables
- Retrieve Message from the Inbound Queue
- Write to the Inbound Queue

Apply MBO Processing
Apply Interface Class Processing
Apply XSL Mapping
Apply User Exit Postprocessing
Gateway
Interface Loading
Interface Tables
Retrieve Message from the Inbound Queue
Write to the Inbound Queue
Writing Messages to the Inbound Queues

Via the Integration Gateway

The following steps describe the initiation of inbound processing via the integration gateway:

1. The external system delivers a message to the integration gateway via XML over HTTP or an EJB call. The message can be in any XML format.

2. The adapter’s interpreter layer identifies the sender (external system) and the interface name from the message.

3. The integration framework verifies the following:
   - The external system is valid and enabled.
   - The interface is a valid and enabled inbound interface for the system.

   If the verification fails, the integration framework notifies the sender of the error and does not process the message.

   **Tip** To find out if an external system is enabled, look on the System tab in the External Systems application.

   **Tip** To find out if an interface is valid and enabled, look on the Inbound Interfaces tab in the External Systems application.

4. If the verification is successful, the integration gateway identifies the inbound JMS queue assigned to the selected interface and external system.

   **Tip** To find out which inbound queue (continuous or sequential) is assigned to the interface and external system, look on the Inbound Interfaces tab in the External Systems application.

   To find out the location of that queue, look on the System tab in the External Systems application.

5. The integration gateway writes the message to the inbound queue. If the message contains multiple instances of a document (for example, if a single message contains ten purchase orders), the application writes the single message, not ten individual messages, to the queue.

6. The integration framework updates the message header with the external system and interface names.

Continue with “Retrieving Messages from the Inbound Queues,” on page 3-22.
Inbound Integration Processing

**Via the Data Import Feature**

The following steps describe the initiation of inbound processing via the Data Import feature.

**NOTE**

This feature is available only for external systems using an internal adapter.

1. A user selects the **Data Import** option from the Select Action menu in the External Systems application.

2. The user identifies the type (XML or flat), delimiter, and location of the file to be imported into Maximo.

3. The adapter identifies the sender (external system) and the interface name from the message.

4. The integration framework verifies the following:
   - The external system is valid and enabled.
   - The interface is a valid and enabled inbound interface for the system.

   If the verification fails, the integration framework notifies the user of the error and does not process the message.

5. If the verification is successful, the integration framework identifies the inbound JMS queue assigned to the interface and external system.

6. The integration framework writes the message to the inbound queue. If the message contains multiple instances of a document (for example, if a single message contains ten purchase orders), the application writes ten messages to the queue.

7. The integration framework updates the message header with the external system and interface names.

Continue with “Retrieving Messages from the Inbound Queues,” on page 3-22.
The following steps describe the initiation of inbound processing via interface tables. For more information about interface tables, see Chapter 5, "Interface Tables," on page 5-1.

**NOTE** This option is available only for external systems using an internal type adapter.

1. The external system writes transaction data to the appropriate interface tables and updates the MXIN_INTER_TRANS queue table with information about the sequence in which Maximo must process the interface table records.

2. A Maximo cron task regularly polls the MXIN_INTER_TRANS queue table for records to be processed.

3. If any records are ready to be processed, the adapter identifies the sender (external system) and interface name from the MXIN_INTER_TRANS queue table.

4. The integration framework verifies the following:
   - The external system is valid and enabled.
   - The interface is a valid and enabled inbound interface for the system.

   If the verification fails, the integration framework sends an e-mail error notification to the system administrator (defined in the External Systems application) and does not process the inbound message.

5. If the verification is successful, the integration framework identifies the inbound JMS queue assigned to the interface and external system and writes the message to that queue.
Inbound Integration Processing

Retrieving Messages from the Inbound Queues

Summary
The integration framework retrieves messages from the inbound queues.

Maximo Enterprise Adapter Inbound Processing Activities

Users can choose to process inbound messages through the sequential queue, the continuous queue, or a combination of the two. The sequential queue processes messages on a strict FIFO basis, while the continuous queue supports multi-threaded processing of transactions.

When an error occurs while processing a message in the sequential queue, the application sends an e-mail message to the system administrator, flags the message as an error, then stops processing messages in the queue until the error is corrected.

When an error occurs while processing a message in the continuous queue, the application sends an e-mail message to the system administrator, flags the message as an error, then continues processing subsequent messages in the queue.

Tip
To find out which inbound queue (sequential or continuous) an external system-interface uses, look on the Inbound Interfaces tab in the System External Systems application.

For more information about the use of queues, see Chapter 11, "JMS Queue Configuration," on page 11-1.

At this point, inbound messages that were received into the queue from the interface tables or from file loading are converted into Maximo XML format. This applies to messages from external systems using the Maximo adapter or any internal type adapter.

A message remains in the inbound queue until it is fully processed by the integration framework. Once processing is complete, the transaction is deleted from the queue.

Output
The output of this activity is an XML message in the interface format.
Identifying the Integration Point

**Summary**

The integration framework identifies the integration point(s) associated with the interface and creates a copy of the message (interface) for each integration point.

**Maximo Enterprise Adapter Inbound Processing Activities.**

1. The integration framework retrieves the record from the inbound queue.
2. The adapter used by the external system is identified.
3. The interface definition in the adapter lists the inbound integration point(s) to which the interface is mapped.
4. The integration framework creates one copy of the interface for every integration point that is associated with the interface.

The processing sequence associated with the integration points specifies the sequence in which the subsequent integration point processing is to be performed on the interface.

**Tip**

To see the processing sequence of multiple integration points associated with an interface, go to the Interface tab in the Integration Interfaces application. Look at the Process Order field on the Inbound Integration Points subtab.

If the integration framework creates copies of the interface, the remaining inbound processing actions apply to each copy of the interface.

**Output**

The output of this activity is one copy of the interface per integration point.
Inbound Integration Processing

Applying User Exit Preprocessing

Summary

The integration framework applies the custom processing logic in the user exit class (if specified) to the interface.

Maximo Enterprise Adapter Inbound Processing Activities.

A preprocessing method in a user exit class allows manipulation of an interface before any predefined processing takes place. Users typically use this functionality to customize predefined interfaces.

The predefined adapters do not provide any user exit classes.

Tip

To find out if preprocessing customization exists, look on the Interface tab in the Integration Interfaces application. If the Inbound Integration Points subtab shows a User Exit Class, check that class file for a preprocessing method.

Note

The same processing class contains the user exit preprocessing, user exit postprocessing (see page 3-26), and user exit MBO processing (see page 3-34) methods.

Output

The possible outcomes of this activity are as follows.

Processing Outcome

<table>
<thead>
<tr>
<th>If:</th>
<th>Output is . . .</th>
</tr>
</thead>
<tbody>
<tr>
<td>No user exit class exists</td>
<td>The existing interface, unchanged</td>
</tr>
<tr>
<td>The user exit class skips the record (due to non-applicable data)</td>
<td>No output; processing ends and the transaction is deleted from the queue</td>
</tr>
<tr>
<td>The user exit class stops the record (due to an error)</td>
<td>No output; processing ends with the transaction in error and remaining in the queue</td>
</tr>
<tr>
<td>The user exit class completes successfully</td>
<td>The existing interface, updated</td>
</tr>
</tbody>
</table>
Applying Interface Class Processing

Summary
The integration framework applies predefined interface processing logic in the interface processing class (if specified) to the interface.

Maximo Enterprise Adapter Inbound Processing Activities

The Maximo adapter does not provide any predefined interface processing classes. It implements all its integration rules via integration point processing and processing rules.

In general, adapters use the interface processing class for the following purposes:

- ▼ to apply interface-specific business rules that cannot be specified using the processing rules
- ▼ to convert input data from interface format to integration object format

Tip
To find out if an interface processing class exists for the interface, go to the Interface tab in the Integration Interfaces application. Look on the Inbound Integration Points subtab for an Interface Processing class.

Output
The possible outcomes of this activity are as follows.

Processing Outcome

<table>
<thead>
<tr>
<th>If:</th>
<th>Output is . . .</th>
</tr>
</thead>
<tbody>
<tr>
<td>No interface processing class exists</td>
<td>The existing interface, unchanged</td>
</tr>
<tr>
<td>The interface processing class skips the record</td>
<td>No output; processing ends and the transaction is deleted from the queue</td>
</tr>
<tr>
<td>The interface processing class stops the record</td>
<td>No output; processing ends with the transaction in error and remaining in the queue</td>
</tr>
<tr>
<td>The interface processing class completes successfully, with mapping</td>
<td>The integration object</td>
</tr>
</tbody>
</table>
Applying User Exit Postprocessing

Summary

The integration framework applies predefined processing logic in the user exit class (if specified) to the output from the preceding activity.

Maximo Enterprise Adapter Inbound Processing Activities

A user exit processing class typically customizes the integration object after the execution of any predefined interface processing logic. Both the integration object and the interface are available at this point.

The Maximo adapter does not provide predefined user exit classes.

**Tip**

To find out if any postprocessing customization exists, look on the Interface tab in the Integration Interfaces application. If the Inbound Integration Points subtab shows a User Exit Class, check that class file for a postprocessing method.

**Note**

The same processing class contains the user exit preprocessing (see page 3-24), user exit postprocessing, and user exit MBO processing (see page 3-34) methods.

Output

The possible outcomes of this activity are as follows.

**Processing Outcome**

<table>
<thead>
<tr>
<th>If:</th>
<th>Output is . . .</th>
</tr>
</thead>
<tbody>
<tr>
<td>No user exit class exists</td>
<td>The integration object, unchanged</td>
</tr>
<tr>
<td>The user exit class skips the record</td>
<td>No output; processing ends and the transaction is deleted from the queue</td>
</tr>
<tr>
<td>The user exit class stops the record</td>
<td>No output; processing ends with the transaction in error and remaining in the queue</td>
</tr>
<tr>
<td>The user exit class completes successfully</td>
<td>The integration object, updated</td>
</tr>
</tbody>
</table>
Applying XSL Mapping

Summary
The integration framework applies any mapping to the interface, to convert it to integration object format.

Maximo Enterprise Adapter Inbound Processing Activities

The Maximo and ERP adapters do not provide predefined XSL maps.

Tip
To find out if an XSL map exists for an interface, go to the Interface tab in the Integration Interfaces application. Look on the Inbound Integration Points subtab for an XSL map.

Output
The possible outcomes of this activity are as follows.

Processing Outcome

<table>
<thead>
<tr>
<th>If:</th>
<th>Output is:</th>
</tr>
</thead>
<tbody>
<tr>
<td>No XSL map exists</td>
<td>The interface, unchanged</td>
</tr>
<tr>
<td>An XSL map exists</td>
<td>The integration object</td>
</tr>
</tbody>
</table>
Inbound Integration Processing

Duplicating the Integration Object

Summary

If the transaction applies to multiple organizations or sites, the integration framework creates a copy of the integration object for each organization or site.

Maximo Enterprise Adapter Inbound Processing Activities.

<table>
<thead>
<tr>
<th>Interface Integration Point Processing</th>
<th>Interface Processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apply MBO Processing</td>
<td>Gateway</td>
</tr>
<tr>
<td>Apply User Exit MBO Processing</td>
<td>File Loading</td>
</tr>
<tr>
<td>Apply Integration Point Processing</td>
<td>Interface Tables</td>
</tr>
<tr>
<td>Apply MBO Processing Rules</td>
<td></td>
</tr>
<tr>
<td>Build MBO</td>
<td></td>
</tr>
<tr>
<td>Apply Integration Object</td>
<td>Duplicate Integration Object</td>
</tr>
<tr>
<td>Apply Integration Object Processing</td>
<td>Apply XSL Mapping</td>
</tr>
<tr>
<td>Apply User Exit Postprocessing</td>
<td>Apply User Exit Preprocessing</td>
</tr>
<tr>
<td>Apply Interface Class Processing</td>
<td>Apply Integration Point</td>
</tr>
<tr>
<td>Identify Integration Point</td>
<td>Retrieve Message from the Inbound Queue</td>
</tr>
<tr>
<td>Write to the Inbound Queue</td>
<td></td>
</tr>
</tbody>
</table>

In most cases, a transaction applies to a single organization or site. However, in cases where the transaction applies to multiple organizations or sites, a multiplication control directs Maximo to create a copy of the integration object for each applicable organization or site.

Example

You want to insert a vendor from an external system into Maximo, under multiple organizations. Instead of sending the vendor from the external system multiple times, you can send it once and apply a multiplication control that will copy it to each organization.

The Maximo adapter does not provide any predefined multiplication controls.

If the remaining inbound activities determine that a duplicated integration object for a specific organization or site should be skipped due to inapplicable data, the skip action applies to that one integration object only. If they determine that a duplicated integration object should be stopped due to an error in the data, the stop action applies to all the integration objects created in this step (every copy of the original integration object).

Tip

To find out if a multiplication control exists, look on the Inbound Integration Points subtab on the Interface tab in the Interfaces application.

1. The integration framework checks for a multiplication control for the integration point and interface.

2. If a control exists, the integration framework multiplies the integration object as many times as there are values in the control.
The integration framework determines if the value in the multiplication control is an organization or site value, and replaces the organization or site in the integration object with the organization or site from the multiplication control.

Output

The output of this activity is an integration object for each organization or site specified in the multiplication control.
Inbound Integration Processing

Applying Integration Object Processing Rules

Summary

The integration framework applies any processing rules to the integration object, before it builds the MBOs.

**Maximo Enterprise Adapter Inbound Processing Activities**

Integration object processing rules define conditions under which Maximo can skip or stop a transaction, or change data in the integration object, prior to the creation of the MBOs. This is the final opportunity to manipulate the primary/unique keys defined for a MBO.

**Tip**

To find out if processing rules exist for the integration object, look on the Integration Object subtab of the Inbound Processing Rules tab in the Integration Interfaces application.

The possible outcomes of this activity are as follows.

**Processing Outcome**

<table>
<thead>
<tr>
<th>If:</th>
<th>Output is:</th>
</tr>
</thead>
<tbody>
<tr>
<td>No integration object processing rules exist</td>
<td>Integration object, unchanged</td>
</tr>
<tr>
<td>The processing rules skip the record</td>
<td>No output; processing ends and the transaction is deleted from the queue</td>
</tr>
<tr>
<td>The processing rules stop the record</td>
<td>No output; processing ends with the transaction in error and remaining in the queue</td>
</tr>
<tr>
<td>The processing rules complete successfully</td>
<td>Integration object, updated Note: The primary keys of the MBOs are set at this point</td>
</tr>
</tbody>
</table>
Building the MBOs

Summary

The integration framework builds the MBO(s) using the information in the integration object.

Maximo Enterprise Adapter Inbound Processing Activities

Interface Integration Point Processing

- Apply MBO Processing
- Apply User Exit MBO Processing
- Apply Integration Point Processing
- Apply MBO Processing Rules
- Build MBO
- Apply Integration Object Processing
- Duplicate Integration Object
- Apply Integration Object Rules

Interface Processing

- Gateway
- Write to the Inbound Queue
- Retrieve Message from the Inbound Queue
- Interface Point
- Identify Integration Point
- Apply Interface Class Preprocessing
- Apply Interface Exit Preprocessing
- Apply XSL Mapping
- Apply User Exit Postprocessing
- Apply User Exit Class Processing
- Apply User Exit MBO Processing
- Apply User Exit Preprocessing
- Apply MBO Processing Rules

Output

The output of this activity is the MBO(s).
Inbound Integration Processing

Applying MBO Processing Rules

Summary
The integration framework applies processing rules (if specified) to the MBOs it has built, before saving the MBOs.

Maximo Enterprise Adapter Inbound Processing Activities

MBO processing rules let you manipulate data in the MBOs, prior to the MBOs being saved. You can also use processing rules to access and retrieve pertinent data from MBOs that are not included in the integration object.

Tip
To find out if processing rules exist for the MBO, look on the MBO subtab of the Inbound Processing Rules tab in the Integration Interfaces application.

Output
The possible outcomes of this activity are as follows.

Processing Outcome

<table>
<thead>
<tr>
<th>If:</th>
<th>Output is:</th>
</tr>
</thead>
<tbody>
<tr>
<td>No MBO processing rules exist</td>
<td>The MBO</td>
</tr>
<tr>
<td>The processing rules skip the record</td>
<td>No output; processing ends and the transaction is deleted from the queue</td>
</tr>
<tr>
<td>The processing rules stop the record</td>
<td>No output; processing ends with the transaction in error and remaining in the queue</td>
</tr>
<tr>
<td>The processing rules complete successfully</td>
<td>The MBO, possibly updated</td>
</tr>
</tbody>
</table>
Applying the Integration Point Processing Class

Summary
The integration framework applies any predefined logic to the integration point.

Maximo Enterprise Adapter Inbound Processing Activities

Interface Integration Point Processing

<table>
<thead>
<tr>
<th>Apply MBO Processing</th>
<th>Apply Integration Point Processing</th>
<th>Apply MBO Processing Rules</th>
<th>Build MBO</th>
<th>Apply Integration Object Processing</th>
<th>Duplicate Integration Object</th>
</tr>
</thead>
</table>

Interface Processing

<table>
<thead>
<tr>
<th>Apply User Exit MBO Processing</th>
<th>Apply Interface Class Processing</th>
<th>Apply User Exit Preprocessing</th>
<th>Apply Integration Point Processing</th>
</tr>
</thead>
</table>

Output
The possible outcomes of this activity are as follows.

<table>
<thead>
<tr>
<th>If:</th>
<th>Output is:</th>
</tr>
</thead>
<tbody>
<tr>
<td>No integration point processing exists</td>
<td>The MBO, unchanged</td>
</tr>
<tr>
<td>The integration point class skips the record</td>
<td>No output; processing ends and the transaction is deleted from the queue</td>
</tr>
<tr>
<td>The integration point class stops the record</td>
<td>No output; processing ends with the transaction in error and remaining in the queue</td>
</tr>
<tr>
<td>The integration point processing completes successfully</td>
<td>The MBO, possibly updated</td>
</tr>
</tbody>
</table>

Note: The predefined interfaces do not implement the Stop action.
**Applying User Exit MBO Processing**

**Summary**

The integration framework applies any final custom logic to the MBO.

**Maximo Enterprise Adapter Inbound Processing Activities**

**Output**

The possible outcomes of this activity are as follows.

**Processing Outcome**

<table>
<thead>
<tr>
<th>If:</th>
<th>Output is:</th>
</tr>
</thead>
<tbody>
<tr>
<td>No user exit processing</td>
<td>The MBO, unchanged</td>
</tr>
<tr>
<td>The user exit class skips the record</td>
<td>No output; processing ends and the transaction is deleted from the queue</td>
</tr>
<tr>
<td>The user exit class stops the record</td>
<td>No output; processing ends with the transaction in error and remaining in the queue</td>
</tr>
<tr>
<td>The user exit class completes successfully</td>
<td>The MBO, possibly updated</td>
</tr>
</tbody>
</table>
Applying MBO Processing

Summary

The MBOs are passed to Maximo and standard Maximo processing is applied.

NOTE

Maximo Integration treats a single message from the inbound queue as a single Maximo transaction. If the original message ultimately results in multiple integration objects, Maximo must successfully process all the MBOs created from all the integration objects created from the message before it performs the database commit. An error in any one of the multiple integration objects will cause the entire transaction to fail.
Inbound Integration Processing

Web Services Processing

Summary

Two types of interfaces can be processed via Web services:

- Data Synchronization (operation = Notify)
- Query (operation = Query)

Web services processing involves the same processing activities as the queue-based transactions, except that the Web services transactions bypass the inbound queues.

Maximo Enterprise Adapter Web Services Processing

Web services processing is always synchronous. If an external system invokes a Maximo Web service, it receives the result of the processing in the same session/transaction.
Data Synchronization Processing

Data synchronization interfaces are processed in the following way. For a visual representation of the processing, see the diagram on page 3-36.

1. An external client directly invokes Web services.

2. The Web service verifies the following:
   ▼ The external system is valid and enabled.
   ▼ The interface is a valid and enabled inbound interface for the system.

If the verification fails, the Web service returns an exception to the client that invoked the service.

3. If the verification is successful, the following inbound interface processing activities take place, as described previously in this chapter:
   ▼ Identifying the Integration Point
   ▼ Applying User Exit Preprocessing
   ▼ Applying Interface Class Processing
   ▼ Applying User Exit Postprocessing
   ▼ Applying XSL Mapping
   ▼ Duplicating the Integration Object
   ▼ Applying Integration Object Processing Rules
   ▼ Building the MBOs
   ▼ Applying MBO Processing Rules
   ▼ Applying the Integration Point Processing Class
   ▼ Applying User Exit MBO Processing

The integration framework then passes the MBO to Maximo and standard Maximo MBO processing takes place.

If an error occurs, the Web service sends the error to the external system as a response. If processing is successful, the Web service sends a success notification to the external client.
Inbound Integration Processing

Query Processing

Query functionality is available only with interfaces within the Maximo (default) adapter.

Query interfaces are processed in the following way. For a visual representation of the processing, see the diagram on page 3-36.

1. An external client directly invokes Web services.

2. The Web service verifies the following:
   - The external system is valid and enabled.
   - The interface is a valid and enabled inbound interface for the system.

   If the verification fails, the Web service returns an exception to the client that invoked the service.

3. If the verification is successful, the integration framework hands the transaction to a Query Processor for further processing.

4. The Query Processor builds a query and submits it to the Maximo query-by-example (QBE) framework for execution.

5. The following inbound interface processing activities take place, as described previously in this chapter.
   - Identifying the Integration Point
   - Applying User Exit Preprocessing
   - Applying Interface Class Processing
   - Applying User Exit Postprocessing
   - Applying XSL Mapping
   - Building the MBOs

6. The following outbound interface processing activities take place, as described previously in this chapter:
   - Identifying the Integration Point
   - Applying the Integration Point Processing Class
   - Applying Interface Processing Rules
   - Applying User Exit Preprocessing
   - Applying Interface Class Processing
   - Applying User Exit Postprocessing
   - Applying XSL Mapping

7. The Web service hands the response to the external client that initiated the query.
This chapter describes the structure, elements and attributes of XML messages that are created by Maximo.

In general, the content of any XML document created by Maximo is based on the integration object that is associated with the interface. Additional elements that provide information about how the document is to be processed, are added to the content. This information depends on the operation performed by the interface (Notify, Query, or Response). This chapter discusses the XML definition of data synchronization (operation = Notify) interfaces only.

For information about the XML used in query and response type interfaces, see Chapter 17, "Using Integration Queries," on page 17-1.

This chapter includes the following sections:

- Structure of Maximo XML
- Interface Element
- Header Element
- Content Element
- Additional Considerations
- XML Schemas for Maximo Interfaces
The following diagram shows the standard format used for all XML messages exchanged via the Maximo adapter. The root element, Interface, is the name of the interface. Within the Interface element are two required elements, Header and Content. Each XML document has one Header and one Content element.

**Maximo XML Structure**

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

**Character Encoding**

By default, Maximo XML uses UTF-8 encoding. If an inbound transaction has any other encoding specified, the entire message must use the same encoding. If an error is encountered during the processing of an inbound transaction that uses an encoding other than UTF-8, the entire error XML that is written will be encoded as UTF-8.

**Example**

```
<?xml version="1.0" encoding="ISO-8859-2"?>
```
Interface Element

The Interface element contains the name of the interface, as defined in the Integration Interfaces application. The inbound integration processing verifies that this is a valid and enabled inbound interface.

The Interface element can include the following attributes.

**Interface Element Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>xmlns</td>
<td>Namespace identifier</td>
<td>Optional inbound</td>
</tr>
<tr>
<td></td>
<td>Value: <a href="http://www.mro.com/mx/integration">http://www.mro.com/mx/integration</a></td>
<td></td>
</tr>
<tr>
<td>language</td>
<td>Identifier of the base language of the fields in the XML message. (Does not apply to language-specific record (for example, L_ITEM, L_ASSET, and so on).</td>
<td>Optional inbound</td>
</tr>
<tr>
<td></td>
<td>Outbound—contains the Maximo base language code; provided for information only.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Inbound—Maximo does not evaluate this attribute. It assumes the base language of the document is the same as the base language of Maximo.</td>
<td></td>
</tr>
</tbody>
</table>

**Example**

```xml
<MXGLTXNInterface xmlns="http://www.mro.com/mx/integration" language="EN"/>
```

The interface element contains only two child elements—a single Header element and a single Content element.
Header Element

Every XML message contains one header element, which specifies the sender and recipient of the message and uniquely identifies each message.

The Header includes attributes and other elements. Depending on the operation type of the interface (Notify, Query, or Response), the attributes will differ but the elements will be the same.

This section discusses data synchronization interfaces only. For information about query and response type interfaces, see Chapter 17, "Using Integration Queries," on page 17-1.

Header Attributes

The Header element can include the following attributes.

**Header Element Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>operation</td>
<td>The purpose of the XML message. Its value is based on the operation associated with the interface in the Integration Interfaces application.</td>
<td>Inbound—Optional, because the interface name identifies the operation from the definition of the interface in Maximo. Any value provided is not validated by inbound processing.</td>
</tr>
<tr>
<td>event</td>
<td>The origin of an outbound XML message.</td>
<td>Outbound—Provided for information only. Maximo does not evaluate this attribute.</td>
</tr>
<tr>
<td></td>
<td>Valid values are:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>▼ Notify</td>
<td></td>
</tr>
<tr>
<td></td>
<td>▼ Query</td>
<td></td>
</tr>
<tr>
<td></td>
<td>▼ Response</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Valid values are:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 (false)—Generated via the Data Export feature</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 (true)—Generated via an outbound integration event listener (that is, data entry in Maximo)</td>
<td></td>
</tr>
</tbody>
</table>

Example

```xml
<MXGLTXNInterface xmlns="http://www.mro.com/mx/integration" language="EN">
  <Header operation="Notify" event="1">
```

**Header Fields**

The Header element includes the following fields.

**Header Element Fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Required/Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SenderID</strong></td>
<td>An identifier for the system or application sending the message. For more information, see &quot;SenderID Attributes,&quot; following this table.</td>
<td>Required inbound</td>
</tr>
<tr>
<td><strong>CreationDateTime</strong></td>
<td>The date and time the interface was initiated.</td>
<td>Optional inbound</td>
</tr>
<tr>
<td><strong>RecipientID</strong></td>
<td>The receiving system or application for which the message is intended.</td>
<td>Not applicable inbound</td>
</tr>
<tr>
<td><strong>MessageID</strong></td>
<td>Unique identifier for the combination of message and external system.</td>
<td>Optional inbound</td>
</tr>
</tbody>
</table>

**SenderID Attributes**

The SenderID field can include the following attributes. They apply only to outbound transactions.

**SenderID Field Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>&quot;Maximo&quot;</td>
</tr>
<tr>
<td>majorversion</td>
<td>Maximo major release number</td>
</tr>
<tr>
<td>minorversion</td>
<td>Maximo minor release number</td>
</tr>
<tr>
<td>build</td>
<td>Maximo build number</td>
</tr>
<tr>
<td>dbbuild</td>
<td>Maximo database build number</td>
</tr>
</tbody>
</table>

**Example**

```xml
<MXGLTXNInterface xmlns="http://www.mro.com/mx/integration" language="EN">
  <Header operation="Notify" event="1">
    <SenderID type="Maximo" majorversion="6" minorversion="0" build="138" dbbuild="185">MX</SenderID>
  </Header>
</MXGLTXNInterface>
```
Content Element

Every XML document contains one Content element, which contains one or more instances of the same integration object. For example, a message containing the MXPOInterface can contain multiple purchase orders.

The format of the content element varies, depending on the operation type of the interface (Notify, Query, or Response) and the type of the related integration object (standard or merged).

This section discusses data synchronization interfaces only. For information about query and response type interfaces, see Chapter 17, "Using Integration Queries," on page 17-1.

Standard Integration Objects

The Content element for a standard integration object has the following generic structure:

```xml
<InterfaceName>
  <Header></Header>
  <Content>
    <Integration Object Name>
      <Primary MBO Name>
        <Fields>
          ...
          ...
          ...
      </Fields>
    </Primary MBO Name>
  </Content>
</InterfaceName>
```

**NOTE** The bold text in the preceding and following structures indicates the areas where the content of standard and merged integration objects differs.

**Example**

The MXPO integration object is a standard integration object. It has the following structure:

```xml
<MXPOInterface>
  <Header></Header>
  <Content>
    <MXPO>
      <PO>
        ...
        ...
        ...
        <PONUM>
          ...
          ...
          ...
        </PONUM>
      </PO>
      <POLINE>
        <POCOST>
          ...
          ...
          ...
        </POCOST>
      </POLINE>
      <PO>
    </MXPO>
  </Content>
</MXPOInterface>
```
Merged Integration Objects

The content element for a merged integration object has the following generic structure:

```xml
<InterfaceName>
  <Header></Header>
  <Content>
    <Integration Object Name>
      <Integration Object Name>
        .
        .
        .
    </Integration Object Name>
    .
    .
  </Content>
</InterfaceName>
```

**NOTE** The bold text in the preceding and following structures indicates the areas where the content of standard and merged integration objects differs.

**Example**

The MXReceipt integration object is a merged integration object that processes both the MATRECTRANS and SERVRECTRANS MBOs. It has the following structure:

```xml
<MXRECEIPTInterface>
  <Header></Header>
  <Content>
    <MXRECEIPT>
      <MXRECEIPT>
        <ITEMNUM>
          <TOSTORELOC>
            .
            .
          </TOSTORELOC>
        </ITEMNUM>
      </MXRECEIPT>
    </MXRECEIPT>
  </Content>
</MXRECEIPTInterface>
```

**NOTE** The name of the integration object—in this case, MXRECEIPT—appears twice in the structure of a merged integration object.
**Action Attribute**

The action attribute is an optional attribute that can apply to the content of primary and child MBOs in an integration object. It specifies the type of processing the receiving system is to perform on the XML message.

For inbound processing, this attribute can be used in XML documents, flat files or interface tables that perform data synchronization (operation = Notify).

For outbound messages, it is used in data synchronization interfaces defined within the Maximo adapter and generated by an outbound integration event (that is, an action in Maximo). The Data Export feature does not update records and therefore does not include an action attribute in the messages it generates.

The following table lists the valid values for the action attribute.

*NOTE* Action values are case-sensitive

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Add record(s) to the database in the receiving system</td>
</tr>
<tr>
<td>Delete</td>
<td>Delete record(s) from the database in the receiving system</td>
</tr>
<tr>
<td>Change*</td>
<td>Update existing record(s) in the database in the receiving system</td>
</tr>
<tr>
<td>Replace*</td>
<td>Update existing record(s) in the database in the receiving system</td>
</tr>
<tr>
<td>AddChange*</td>
<td>Add or update existing record(s) in the database in the receiving system</td>
</tr>
<tr>
<td>Null</td>
<td>Add record(s) or replace record(s) in the receiving system, depending on whether or not the primary record exists in the database</td>
</tr>
</tbody>
</table>

*These actions differ in the information they include in the XML message and the processing they require of the receiving system. Later sections of this chapter detail the differences.

An action attribute at the primary MBO level specifies the overall processing action that applies to the parent and child records. At the child MBO level, it indicates processing specific to that record.

*NOTE* The business rules defined by Maximo always take precedence over the action attribute. If business rules prohibit the action specified on an inbound XML message, an error occurs. For example, an inbound transaction that attempts to update a closed PO will result in an error.

If a single XML document contains multiple instances of an integration object, each instance of the interface can specify a different action attribute. In the following example, the COMPANIES record has multiple child COMPCONTACT records, each with its own action attribute.
Example

```xml
<MXVENDOR>
  <COMPANIES action="Change">
    <COMPANY>TEST4</COMPANY>
    <NAME>test</NAME>
    <ADDRESS1>100 Main Str</ADDRESS1>
    <COMPCONTACT action="Add">
      <NAME>SMITH</NAME>
      <TITLE>MANAGER</TITLE>
    </COMPCONTACT>
    <COMPCONTACT action="Change">
      <NAME>JONES</NAME>
      <TITLE>ENGINEER</TITLE>
    </COMPCONTACT>
  </COMPANIES>
</MXVENDOR>
```

Add Action

An add action (action = "Add") indicates that the corresponding data will be added to a database. For inbound transactions, an error occurs if the data already exists. The add action on a primary MBO extends to its child MBOs, so it is not necessary to also specify the add action at the child MBO level. Outbound transactions contain an add action when the insert of a MBO generates the transaction.

Delete Action

A delete action (action = "Delete") indicates that the corresponding MBO and its child MBOs will be deleted from the database. The delete action on a primary MBO extends to its child MBOs, so it is not necessary to also specify the delete action at the child MBO level.

An outbound XML message with a delete action on the primary MBO may not include the child MBOs, but the receiving system is responsible for identifying and deleting them.

When a primary MBO in an inbound XML message contains the delete action, Maximo deletes the child MBOs related to the primary MBO. If the parent MBO does not exist in Maximo, no error is reported to the sending system.

Update Actions

The Change and Replace actions indicate that the existing database records will be updated. The AddChange action can result in an update if the record to be processed already exists in the receiving system.

The change and replace actions are interchangeable in the case of updates to single-level (no child MBOs) standard interfaces and merged interfaces. For hierarchical interfaces, the two actions have different XML content and require different processing by the receiving system.

The following sections explain the differences among the three update actions. Also, see "Content Element," on page 4-11, to understand the differences among these actions.
Change Action

A change action (action = "Change") on the primary MBO in an XML message indicates that the message contains one or more parent or child records that will be added, changed, or deleted. It always contains the parent of any child record to be updated, even if the parent is unchanged.

When the primary MBO in an inbound or outbound XML message contains a change action, each child MBO in the message can contain one of the action codes in the following table.

**Child MBO Action Codes**

<table>
<thead>
<tr>
<th>Action Code of Child MBO</th>
<th>Processing Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>action=&quot;Add&quot;</td>
<td>Add the child record; if it exists, error</td>
</tr>
<tr>
<td>action=&quot;Delete&quot;</td>
<td>Delete the child record; if it does not exist, error</td>
</tr>
<tr>
<td>action=&quot;Change&quot;</td>
<td>Update the child record; if it does not exist, error</td>
</tr>
<tr>
<td>action = &quot;&quot; or no action specified</td>
<td>If the child record exists, update it; if child record does not exist, add it</td>
</tr>
</tbody>
</table>

**NOTE**

If an action code not listed in the preceding table is specified, an error will occur in the processing of the XML document.

Replace Action

A replace action (action = "Replace") on the primary MBO in an XML message indicates that the message contains a complete set of MBOs that represent the net result of additions, changes, and deletions to the integration object. These MBOs will replace the existing database records, and any database record that is not referenced in the XML message will be deleted.

**NOTE**

For outbound processing, Maximo always uses the replace action, not the change action.

In inbound processing, Maximo deletes any existing child record that is not explicitly mentioned in the message. External systems must process Replace actions in the same manner; that is, delete any child records that are not part of the XML document.

The replace action can appear only on the primary MBO in an XML message. If a child record in an inbound XML message contains a replace action when the primary MBO contains a change action, Maximo will not be able to process the message. If a child level record contains a replace action when the parent contains any action other than change, Maximo ignores the action on the child record.

AddChange Action

The AddChange action (action = "AddChange") is similar to the replace action, except that any existing child record that is not specifically mentioned in the message will not be deleted. An AddChange action on the primary MBO adds the primary record and all the sub-records provided in the message, if the primary record does not exist in Maximo. If the primary record does exist, it will be updated along with any child record provided, and existing child records that are not provided in the inbound message will not be deleted. The AddChange action does not apply to child MBOs.

The AddChange action is useful in cases where the integration object definition contains elements that are not available in the external system.
Example

The MXVENDORInterface contains both vendor and contact information. If Maximo maintains the contacts for a vendor and external systems maintain the vendor definition itself, sending an inbound vendor record with action = "" (null) will result in the contact information in Maximo being deleted. However, sending a vendor record with action = "AddChange" will result in the vendor information being updated and the contacts remaining as they are.

Default Action Attributes

If an inbound XML message does not specify an action attribute, Maximo processes the message as follows:

If the primary record does not exist in Maximo, it performs add action processing, as described on page 4-9.

If the primary record exists in Maximo, it performs replace action processing, as described on page 4-10.

Comparison of the Change, Replace, and AddChange Actions

The following diagram contrasts the Change, Replace, and AddChange actions applied to a hypothetical purchase order.

The Original PO diagram shows the initial purchase order. It contains three line items, each with two cost lines.

The Changes to the PO diagram shows the changes to the purchase order that originated in Maximo or an external system. The changes are:

- ▼ A change to the POCOST2 record associated with POLINE1
- ▼ Deletion of the POCOST3 record associated with POLINE1
- ▼ A change to POLINE2
- ▼ Deletion of POLINE3 and, by default, its child POCOST records
Records Accompanying Changes to a Purchase Order

Original PO

PO 101
- POLINE 1
  - POCOST 1
  - POCOST 2
- POLINE 2
  - POCOST 1
  - POCOST 2
- POLINE 3
  - POCOST 1
  - POCOST 2

Changes to the PO

PO 101
- POLINE 1
  - POCOST 1
  - POCOST 2
- POLINE 2
  - POCOST 1
  - POCOST 2
- POLINE 3
  - POCOST 1
  - POCOST 2

PO with action="Change" or "AddChange"

PO 101
- POLINE 1
  - POCOST 1
  - POCOST 2
- POLINE 2
  - POCOST 1
  - POCOST 2
- POLINE 3
  - POCOST 1
  - POCOST 2

PO with action="Replace"

PO 101
- POLINE 1
  - POCOST 1
  - POCOST 2
- POLINE 2
  - POCOST 1
  - POCOST 2
- POLINE 3
  - POCOST 1
  - POCOST 2

- action="Change"
- action="Delete"
- action="Replace"
- No update
Records Accompanying the Change Action

The **PO with action**="Change" or "AddChange" diagram shows the records that accompany a change action on the primary (PO) MBO.

For the sample PO, the sending system would send the following records:

- A change action on the PO record (PO 101), due to the changes in its child POLINE and POCOST records
- A change action on POLINE 1, due to changes in its child PCOST records
- A change action on the POCOST 2 record within POLINE 1, due to changes in that record
- A delete action on the POCOST 3 record within POLINE 2
- A change action on the POLINE 2 record, due to changes in that record
- A delete action on the POLINE 3 record (which automatically deletes all its child records)

Records Accompanying the Replace Action

The **PO with action**="Replace" diagram shows the records that accompany a replace action on the primary MBO.

For the sample PO, the sending system would send the following records. Some may contain updated information; some may contain the same data already on the database.

- The PO header
- POLINE 1 and POCOST lines 1 and 2
- POLINE 2 and POCOST lines 1 and 2

The XML message does not include the POCOST 3 record for POLINE 1, or the POLINE 3 record, so the receiving system deletes these records and POLINE 3’s child records.

Records Accompanying the AddChange Action

In this example, the same records that accompany the Change action would accompany the AddChange action.
Valid Action Attribute Combinations

The following table summarizes the combinations of action attributes you can include on primary and child records. Find the action attribute for the parent record in the column on the left, then read across the table to find out if the action attribute for the child is valid.

**Action Attributes on Primary and Child Records**

<table>
<thead>
<tr>
<th>Child Record</th>
<th>Add</th>
<th>Delete</th>
<th>Change</th>
<th>Replace</th>
<th>AddChange</th>
<th>No Value</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary Record</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Add</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>All child values ignored</td>
</tr>
<tr>
<td>Delete</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>All child values ignored</td>
</tr>
<tr>
<td>Change</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Replace and AddChange not allowed at child level</td>
</tr>
<tr>
<td>Replace</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>All child values ignored</td>
</tr>
<tr>
<td>AddChange</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>All child values ignored</td>
</tr>
<tr>
<td>No value</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>All child values ignored</td>
</tr>
</tbody>
</table>
Field Attributes

The following attributes apply at the field level on outbound XML messages:

- `changed`
- `glorder`
- `langenabled`
- `maxvalue`

These attributes appear in messages generated by an outbound integration event (that is, a Maximo action). All except the `changed` attribute also appear in messages generated by the Data Export feature.

**Changed Attribute**

The changed attribute is a Boolean field attribute that contains the value 1 if a Maximo user modified the value in the corresponding field. This information lets external systems and users identify the field-level change that triggered an outbound interface. This attribute is never present in XML created by the Data Export feature.

**NOTE** The outbound XML contains this attribute only when the transaction meets all the following conditions:

- An outbound, event-based transaction creates the message.
- The interface is a data synchronization interface.
- The action attribute on the primary MBO is Change or Replace.
- The integration object uses the same parent-child MBO relationship as the corresponding Maximo application. For example, the MXPO integration object and the Purchase Orders application must use the same relationships between PO and POLINE, and POLINE and POCOST.

**Examples**

```
<Content>
  <MXITEM>
    <ITEM action="Replace">
      <ITEMNUM>560-00</ITEMNUM>
      <DESCRIPTION changed="1">Tubing, Copper-1 In ID X .030 In Wall Test</DESCRIPTION>
    </ITEM>
  </MXITEM>
</Content>
```

**NOTE** In the case of GL type fields, the changed attribute appears on the field name, as in the following example:

```
<GLDEBITACCT changed="1">
  <VALUE>6600-800-SAF</VALUE>
  <GLCOMP glorder="0">6600</GLCOMP>
  <GLCOMP glorder="1">800</GLCOMP>
  <GLCOMP glorder="2">SAP</GLCOMP>
</GLDEBITACCT>
```
Glorder Attribute

Fields that identify general ledger accounts (fields with MAXTYPE = GL) have the following XML structure.

**General Ledger Account Structure**

```
<table>
<thead>
<tr>
<th>Field (e.g. GLDEBITACCT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>VALUE</td>
</tr>
<tr>
<td>GLCOMP (1..20)</td>
</tr>
</tbody>
</table>
```

In outbound XML the value of a GL type field, including delimiters, appears in the VALUE child element within the field. The outbound XML also places the GL type field’s individual components, based on the Maximo definition of the components, in the GLCOMP element. The glorder attribute in the GLCOMP element identifies the level of the component (GLORDER1 through GLORDER20) within Maximo.

**Example**

```
<GLDEBITACCT>
  <VALUE>6600-800-SAF</VALUE>
  <GLCOMP glorder="0">6600</GLCOMP>
  <GLCOMP glorder="1">800</GLCOMP>
  <GLCOMP glorder="2">SAF</GLCOMP>
</GLDEBITACCT>
```

Inbound XML messages can contain GL account numbers in one of the following formats:

- The external system can provide the individual components, as in the following example:

  ```
  <GLDEBITACCT>
    <GLCOMP glorder="0">6400</GLCOMP>
    <GLCOMP glorder="1">2</GLCOMP>
    <GLCOMP glorder="2">10</GLCOMP>
  </GLDEBITACCT>
  ```

  In this case, Maximo validates the components and uses the segment delimiter defined in Maximo to recreate the account number.

- The external system can provide the entire account number, separated by the delimiter defined in Maximo, as in the following example:

  ```
  <GLDEBITACCT>
    <VALUE>6400-2-10</VALUE>
  </GLDEBITACCT>
  ```

  In this case, the delimiter is used to identify each individual component, and the account number is validated accordingly.

**NOTE** If the external system provides both an account number and its individual components, Maximo ignores the components.
Langenabled Attribute

The XML generated by Maximo includes the langenabled attribute on every translatable column, as shown in the following example:

```xml
<DESCRIPTION langenabled="1">Item 1 description</DESCRIPTION>
```

Maxvalue Attribute

Fields associated with a synonym type domain can specify the corresponding maxvalue. This value is available for customization or user/interface exit processing. It is informational only and is not used for processing.

For more information about synonym domains, refer to the IBM Maximo System Administrator’s Guide.

Example

```xml
<MXITEM>
  <ITEM>
    <ITEMNUM>560-00</ITEMNUM>
    <DESCRIPTION>Tubing, Copper-1 In ID X .030 In Wall Test</DESCRIPTION>
    <LOTTYPE maxvalue="NOLOT">NOLOT</LOTTYPE>
  </ITEM>
</MXITEM>
```
Null Columns

If a tag in an inbound transaction contains no value, the MBO updates the corresponding Maximo database column with a null value. If the XML does not include a tag for a particular field, the MBO does not update that field in the Maximo database.

Boolean Columns

In inbound transactions, a tag that represents a Boolean field must contain a value of 0 (false) or 1 (true). If the tag does not contain a 0 or a 1, the MBO generates an error. If the XML does not include a tag for a Boolean field, the MBO updates the corresponding Maximo database value with the default value (0 or 1) defined for that column.

Number Format

Regardless of the locale setting of the Maximo application server or the database, all decimal fields must use a period (.) as the decimal placeholder. There is no formatting of numbers to the left of the placeholder. This format applies to inbound and outbound data.

Example

$1,738,593.64 must be in the following format: 1738593.64

Date Format

Maximo XML supports the ISO 8601 date format.

Example

2004-12-06T10:11:58-05:00

TRANS_LANGCODE Field

Maximo adds the TRANS_LANGCODE field, as a user-defined field, to the primary MBO of all predefined integration objects. For outbound transactions, this field identifies the language code of the logged-in Maximo user who initiated a transaction, and any translatable columns (usually descriptions and long descriptions) in the XML transaction will be in this language.

If an inbound message contains translatable columns in a language other than the Maximo base language, the TRANS_LANGCODE must contain the language code of the translatable columns. All translatable columns in a single transaction must be in the same language.
XML Schemas for Maximo Interfaces

Users can generate XML schemas for all Notify and query type interfaces defined within the Maximo adapter. These schemas can be used to determine the minimum fields required to successfully process an inbound XML message in Maximo.

Schema Content

Maximo supports one-way and request-response message exchange scenarios. Notify operation interfaces are generally used as standalone messages in one-way exchange scenarios, while query and response type interfaces are always paired together in a request-response exchange scenario. Accordingly, the schema file definition for notify type interfaces contains only one interface, while the schema file definition for query type interfaces includes the query type interface and its corresponding response type interface.

The schema generated for an interface is based on the configured integration object associated with the interface. Since users can modify the predefined integration objects, Maximo does not provide generated schemas for the predefined interfaces, but gives users the ability to generate and regenerate them. You do this via the Select Action menu in the Integration Interfaces application.

Key Fields

The generated schema identifies the key fields within an interface, based on the integration object definition and Maximo data dictionary definitions for the corresponding object. Key fields are identified through XML schema annotation.

Example

ITEMNUM is a key field in the MXITEM integration object schema. It is indicated as follows:

```xml
<xsd:element name="ITEMNUM" minOccurs="0" type="MXString">
    <xsd:annotation>
        <xsd:documentation>ITEMNUM is a key field</xsd:documentation>
    </xsd:annotation>
</xsd:element>
```

Validation

Currently, outbound XML generated by Maximo and inbound XML are not validated against the corresponding interface schema. Maximo business rules apply to inbound data regardless of schema validation.

Schema Attribute

The schema generated for an interface includes the following attribute on a column that is multi-language-enabled:

langenabled="1"
Schema Directories

Schema creation results in a schema for the interface and the integration object associated with the interface. Maximo writes the generated schema files to the `<Integration Root>/Schemas` directory, where `<Integration Root>` is the global directory location. This directory contains the following three subdirectories:

- Interfaces
- Integration-Objects
- MetaData

The name of the interface schema file is `<interfacename>.xsd`

The MetaData directory contains two supporting schemas that are required for the generation of valid interface schemas:

- MXDataTypes.xsd contains a schema datatype for each predefined datatype in the Maximo database.
- MXIntegrationMeta.xsd contains the base definition of a Maximo interface.

**CAUTION**

The MXIntegrationMeta.xsd file is provided by Maximo and should not be changed by users. Changing this file can result in invalid schemas and problems with Web services.

Regeneration of the Schema Definition

Changes to the structure of an integration object require manual regeneration of the schema for all interfaces associated with that integration object. Applicable changes include making a database field required or optional, changing a database field’s data type, adding or removing fields from the integration object, or otherwise changing the structure of an integration object.

**CAUTION**

Changes to the integration object or Maximo data dictionary are not automatically reflected in the schemas, so users are responsible for regenerating the affected schemas. You do this via the Select Action menu in the Integration Interfaces application.
Interface tables are an option provided for integration with systems that use database tables to exchange data. The interface tables are generated based on the definition of the integration object associated with the interface.

Interface tables contain a flat (non-hierarchical) representation of the columns in an integration object. Only data synchronization (operation = Notify) messages can be exchanged using interface tables.

This chapter is directed to anyone who will be responsible for configuring Maximo to use interface tables. It contains the following sections:

- Use of Interface Tables
- Location of Interface Tables
- Naming Convention for Interface Tables
- Interface Queue Tables
- Creation of Interface Tables
- Regeneration of Interface Tables
- Deletion of Interface Tables and Records
- Format of Interface Tables
- Interface Table Polling
- External System Requirements
Use of Interface Tables

A single external system can transfer data via XML messages or interface tables, but not both. If you integrate Maximo with multiple external systems, you can use XML with some external systems and interface tables with others.

To use interface tables for an external system, specify an end point that uses an interface table handler, for that system.

Location of Interface Tables

An external system’s end point definition points to the database where the interface tables will reside. This can be a local or a remote database.

The predefined interface table end point (MXIFACETABLE) points to the local Maximo database, and users can add end points for remote databases. One instance of Maximo can support multiple sets of interface tables in different databases.

Naming Convention for Interface Tables

Default interface table names are derived from integration object names. When you create an interface, Maximo generates a name for the corresponding interface table by adding the suffix _IFACE to the name of the integration object. For example, MXPO_IFACE is the name of the interface table that corresponds to the MXPO integration object.

You can change the default name, if desired. For more information, refer to the online help for the Integration Interfaces application.

All predefined interfaces use the same interface table for inbound and outbound transactions. For example, the MXPO_IFACE table can contain inbound and outbound purchase orders.
Interface Queue Tables

The interface queue tables identify the sequence in which a receiving system should process the records in the respective interface tables. Two queue tables are provided, one for inbound transactions and the other for outbound transactions.

**Interface Queue Tables**

<table>
<thead>
<tr>
<th>Interface Queue Table Name</th>
<th>Direction</th>
</tr>
</thead>
<tbody>
<tr>
<td>MXOUT_INTER_TRANS</td>
<td>Outbound</td>
</tr>
<tr>
<td>MXIN_INTER_TRANS</td>
<td>Inbound</td>
</tr>
</tbody>
</table>

Some transactions depend upon the successful processing of a previous transaction—for example, purchase orders must be processed before receipts—making it critical that the receiving system process them in the same sequence that the sending system created them.

All inbound and outbound transactions must have a record inserted into the corresponding (inbound or outbound) queue table. This record contains a unique sequential identifier, called TRANSID, and a value that identifies the interface table to which the transaction data has been written. The corresponding interface table uses the TRANSID to identify the record or records associated with the transaction. This makes it possible to identify the contents of a transaction by looking up all the records with a given TRANSID value in the corresponding interface table.

The sequence of TRANSID identifies the sequence in which records are processed by Maximo. In a case where purchase orders and receipts are loaded into Maximo, the TRANSID values for the purchase orders must be lower than the TRANSID values for the corresponding receipts.

The only difference between the MXIN_INTER_TRANS and MXOUT_INTER_TRANS queue tables is the direction of the interface table records that they track. Maximo writes to, and the external system can read, the MXOUT_INTER_TRANS queue table. The external system must write to, and Maximo will read, the MXIN_INTER_TRANS queue table.

Maximo generates the interface queue tables the first time that you create interface tables for a given end point. Each end point has its own pair of interface table queues and its own counter for maintaining the outbound TRANSID value.
Creation of Interface Tables

Maximo does not create interface tables until you direct it to do so. You create them manually, via the Select Action menu in the External Systems application.

**NOTE** You can create interface tables only for data synchronization interfaces (operation = Notify) within an internal type adapter. Query, response, and system interfaces do not appear in the Create Interface Tables dialog box in the External Systems application.

Interface tables are created for a specific end point; that is, you must identify the location where the table is to be created. The database location referenced by the end point can be in a local database or a remote database.

When you create interface tables on a local database, the columns are registered in the Maximo data dictionary. Local interface tables that use that table and column automatically reflect any updates (not inserts or deletions) to a base column’s attributes (for example, data type) made through the Maximo Database Configure application. However, changes are not automatically reflected in remote databases. You must manually regenerate remote interface tables to reflect these changes.

Regeneration of Interface Tables

The addition or deletion of columns in Maximo database tables affects the integration objects that use those tables and necessitates the regeneration of all local and remote interface tables associated with those integration objects.

You regenerate interface tables via the Create Interface Tables option on the Select Action menu in the External Systems application. The application will optionally back up existing data in an interface table to a table called `ifacetablename_bak`. Restore the data to the new table, if desired.

**CAUTION** If you do not back up the table, the table will be dropped and the data lost when you regenerate the table.

You cannot regenerate an interface table if the MXIN_INTER_TRANS queue table contains a record that points to that interface table. When a row exists in that queue table, it means the corresponding inbound transaction is pending processing and/or is in error.

**NOTE** The interface table creation process does not check for records in the MXOUT_INTER_TRANS queue table.
Deletion of Interface Tables and Records

After Maximo successfully processes one or more related inbound transaction records in an interface table, it deletes the corresponding record from the MXIN_INTER_TRANS queue table. This means that the transaction was delivered successfully to the inbound JMS queue.

**NOTE** Maximo deletes records from the MXIN_INTER_TRANS queue table only, never from the individual interface tables. The system administrator determines when and how to delete records from the interface tables.

For outbound transactions, the external system must manage the deletion and archiving of data in the queue table and interface tables. Users must manage the archiving of data in inbound interface tables.

You cannot delete interface tables through the user interface or through the deletion of the corresponding integration object. A system administrator can manually drop the table, if necessary.

Format of Interface Tables

The format of an interface table is essentially the same as that of the corresponding integration object. The interface table includes the persistent, non-persistent, and user-defined columns included in the integration object, and it excludes the columns excluded from the integration object.

In addition to the columns in the corresponding integration object, the interface tables include additional columns that identify the sequence in which the sending system wrote, and the receiving system must process, the records in the various interface tables. For more information about these columns, see “Integration Processing Columns” on page 5-7.

Key Columns

If the interface table represents a hierarchical integration object (one containing parent-child MBO relationships), the table does not include any part of the child table’s key columns that are already included in the parent table’s key columns.

For example, the key of the PO MBO consists of PONUM and SITE, and the key of the POLINE MBO consists of PONUM, POLINENUM, and SITE. The PONUM and SITE columns appear only at the parent (PO) level in the MXPO_IFACE interface table.
Duplicate Columns and Aliases

The XML representation of a hierarchical integration object allows duplicate column names, but interface table and flat file representation do not. Therefore, any integration object with duplicate non-key column names in both a parent and child MBO will cause a Duplicate Column Name error during the generation of an interface table or a flat file record.

To resolve this problem, change the Maximo alias for duplicate column names. Every Maximo database column has an alias, or alternate name, that defaults to the name of the column. Maximo uses the alias, not the actual column name, when it generates interface tables and flat files. Changing the alias will eliminate the duplicate column name error. For for information, see “Update of an Alias” on page 5-6.

Column Name Lengths and Aliases

Column names in interface tables are restricted to eighteen characters. Typically this restriction is not an issue with database columns in Maximo, but it can affect non-persistent columns defined in the MBOs. For example, the non-persistent column name for a long description field is generally DESCRIPTION_LONGDESCRIPTION, which is more than eighteen characters long.

Integration objects with fields whose names are more than eighteen characters long will cause a Column Name too long error during the generation of an interface table. To resolve this problem, change the Maximo alias for such columns to a shorter name.

In most cases a column’s alias will match the column name, but some columns have aliases that have been altered to support the integration points and interfaces predefined by Maximo.

**CAUTION** A database column can have only one alias, so ensure that any alias you assign to a column will be valid for every integration object that uses that column. If multiple integration objects use the MBO, a change to an alias will affect all interface tables that refer to those integration objects.

Maximo provides unique aliases for the columns in the predefined integration objects, so you have to check for duplicates only when you create a hierarchical integration object or add a MBO to a predefined integration object.

Update of an Alias

If duplicates exist, use the Add/Modify Alias dialog box in the Integration Objects application to override the alias values. You access this dialog box via the Add/Modify Alias option in the Select Action menu.

Check the Add/Modify Alias dialog box whenever you create or update a user-defined integration object. The dialog box displays the fields and aliases for the MBOs within a selected integration object, and identifies any duplicate alias names by a check in the Duplicate column. If a duplicate alias exists, overwrite its value in the Aliasname column. If a duplicate does not exist, you cannot change the alias.
NOTE If you use interface tables, you must regenerate tables that use the integration object containing the changed alias.

Restricted Columns

The HASLD field, which is an internal Maximo column, is excluded from all integration objects. Do not include this column in any integration object that will be associated with an interface table.

Integration Processing Columns

The following table shows the columns used in the sequencing, retrieval and processing of the interface tables. Some appear in either the interface queue tables or the interface tables; some appear in both places. The following sections explain the function of these fields. For information about IMPORTMESSAGE, see “Interface Table Error Handling” on page 7-3.

Interface Table Processing Columns

<table>
<thead>
<tr>
<th>Column Name</th>
<th>In Interface Queue Tables?</th>
<th>In Interface Tables?</th>
</tr>
</thead>
<tbody>
<tr>
<td>IFACENAME</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>TRANSID</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>TRANSSEQ</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>EXTSYSNAME</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>ACTION</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>IMPORTMESSAGE</td>
<td>Yes (used inbound only)</td>
<td>No</td>
</tr>
</tbody>
</table>

IFACENAME Column

The IFACENAME column in the interface queue table contains the name of the interface used in a transaction. The interface name is a Maximo-defined property of the interface. For outbound transactions, Maximo populates the IFACENAME column. For inbound transactions, the external system must populate the column with the interface name that corresponds to the row it is inserting into an interface table.

For a list of predefined interfaces and their corresponding interface table names,
TRANSID Column

The TRANSID column in an interface queue table is a sequential number that uniquely identifies an integration transaction. The TRANSID, along with the interface table name, identifies a unique transaction. The interface queue table can contain only one record with a given TRANSID value, but the corresponding interface table can have one or more records with that TRANSID, depending on the number of records written to that interface table as part of that interface.

Example

Assume that a Maximo user creates a purchase order with one line item. Assume also that this transaction uses the predefined MXPOInterface, and that it increments the TRANSID value associated with the MXOUT_INTER_TRANS queue table to 1065. The transaction will result in the following records:

- One entry in the MXOUT_INTER_TRANS queue table, with IFACENAME equal to MXPOInterface and TRANSID equal to 1065
- One entry in the MXPO_IFACE interface table, with TRANSID equal to 1065

If the same purchase order has three line items, the transaction will result in the following records:

- One entry in the MXOUT_INTER_TRANS queue table, with IFACENAME equal to MXPOInterface and TRANSID equal to 1065
- Three entries in the MXPO_IFACE interface table, each with TRANSID equal to 1065

  **NOTE** In this case, each entry with TRANSID 1065 will have a unique secondary sequence number. For more information, see “TRANSSEQ Column” on page 5-10.

If a transaction writes to multiple interface tables, the interface queue table will contain a separate record, with a unique TRANSID, for each of those interface tables.

Each interface queue table maintains its own TRANSID counter. Maximo initializes the outbound TRANSID when it generates the interface table queue records. You must manually create and maintain the TRANSID counters that populate inbound queue tables and interface table records.

**CAUTION** If the external systems do not correctly manage the inbound TRANSID counters, sequential processing by Maximo is not guaranteed. Also, ensure that the TRANSID values that the external system generates will not overlap with the TRANSID values that Maximo generates, or errors will occur. This is important for interfaces that process both inbound and outbound data.

Each end point has its own set of interface table queues and its own outbound TRANSID counter.
The following diagram shows an example of the relationship between the interface queue tables and the interface table records. The interface tables contain both inbound and outbound transactions.

**Relationship of Interface Queue Table and Interface Table Records**

In the preceding diagram, the data in the MXOUT_INTER_TRANS queue table directs the external system to process the outbound interface table records as follows:

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Interface Table</th>
<th>Identifier (TRANSID) of Record in Interface Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MXPR_IFACE</td>
<td>1001</td>
</tr>
<tr>
<td>2</td>
<td>MXITEM_IFACE</td>
<td>1002</td>
</tr>
<tr>
<td>3</td>
<td>MXINVENTORY_IFACE</td>
<td>1003</td>
</tr>
<tr>
<td>4</td>
<td>MXITEM_IFACE</td>
<td>1004</td>
</tr>
</tbody>
</table>

The data in the MXIN_INTER_TRANS queue table directs Maximo to process the inbound interface table records as follows:

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Interface Table</th>
<th>Identifier (TRANSID) of Record in Interface Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MXPO_IFACE</td>
<td>2007</td>
</tr>
<tr>
<td>2</td>
<td>MXITEM_IFACE</td>
<td>2008</td>
</tr>
<tr>
<td>3</td>
<td>MXINVENTORY_IFACE</td>
<td>2009</td>
</tr>
</tbody>
</table>
TRANSSEQ Column

When multiple records in an interface table share the same TRANSID, the TRANSSEQ column provides a secondary sequence number that indicates the sequence in which those records should be processed.

Consider again the earlier example of a purchase order with three line items. That transaction might result in the following records:

- One entry in the MXOUT_INTER_TRANS queue table, with IFACENAME equal to MXPOInterface and TRANSID equal to 1065
- Three entries in the MXPO_INTERFACE table, as follows:
  - One entry (PO line 1) with TRANSID equal to 1065 and TRANSSEQ equal to 1
  - One entry (PO line 2) with TRANSID equal to 1065 and TRANSSEQ equal to 2
  - One entry (PO line 3) with TRANSID equal to 1065 and TRANSSEQ equal to 3

The TRANSSEQ column appears only in the interface tables.

EXSYSNAME Column

An interface table can contain both inbound and outbound transactions. The EXSYSNAME column in the interface queue tables indicates whether a record contains inbound or outbound data.

<table>
<thead>
<tr>
<th>Value of EXSYSNAME</th>
<th>Direction of Transaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>The external system, as defined in Maximo, that is the destination of the transaction</td>
<td>Outbound (initiated in Maximo)</td>
</tr>
<tr>
<td>The value of a valid and enabled external system that is defined in Maximo</td>
<td>Inbound (initiated in an external system)</td>
</tr>
</tbody>
</table>
**ACTION Column**

The ACTION column in an interface table queue can optionally specify the processing to be applied to the corresponding interface table.

<table>
<thead>
<tr>
<th>Value in ACTION Column</th>
<th>Maximo or External System Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Insert the data provided in the message.</td>
</tr>
<tr>
<td>Delete</td>
<td>Delete the parent data, and any related child data, from the database.</td>
</tr>
<tr>
<td>Change</td>
<td>Update parent and child data per the contents of the message, but do not delete existing child data not explicitly specified in the message.</td>
</tr>
<tr>
<td>Replace</td>
<td>Replace the existing record(s) with the contents of the message and delete existing child data not referenced in the message.</td>
</tr>
<tr>
<td>AddChange</td>
<td>If the primary record does not exist, process as Add action; otherwise, process as a Change action.</td>
</tr>
<tr>
<td>Null</td>
<td>If the primary record does not exist, process as Add action; otherwise, process as a Replace action.</td>
</tr>
</tbody>
</table>

Maximo populates the ACTION column in outbound messages. If the external system does not populate the column in inbound messages, Maximo tries to retrieve and update the corresponding database record. If the record does not exist, Maximo tries to add it to the database.

For more information about these actions, see Chapter 4, "Maximo XML and Schema," on page 4-1.

**Long Description Columns in Oracle Databases**

Long description columns in Maximo are stored in a CLOB column in an Oracle database. However, interface tables contain two versions of each long description column, one with data type ALN (alphanumeric, length = 4000) and one with data type CLOB (character large object). The name of the alphanumeric column is the column’s alias. The name of the CLOB column is the column’s alias with the suffix 2.

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Name of Description Column</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALN</td>
<td>PO_DESCRIPTION_LD2</td>
</tr>
<tr>
<td>CLOB</td>
<td>PO_DESCRIPTION_LD</td>
</tr>
</tbody>
</table>

Maximo populates both columns in outbound transactions. For inbound transactions, Maximo uses the value in the ALN column if it is not null; otherwise, it uses the value in the CLOB column.
A predefined cron task, IFACETABLECONSUMER, polls the MXIN_INTER_TRANS queue table and uses the IFACENAME, EXTSYSNAME, and TRANSID values in the queue table to sequentially place the corresponding interface table records into the appropriate inbound JMS queue. From there, Maximo processes the individual records. The cron task does not poll the interface tables directly.

The polling process verifies that the record’s external system and interface name are valid and currently enabled. If they are not, the record is marked in error and remains in the interface table.

If you disable polling, new records will remain in the interface tables, but Maximo will process any messages already sent to the inbound JMS queue.

Users must set up a mechanism to retrieve outbound transactions from the interface tables. This can be done using a polling program, as Maximo does for inbound transactions; triggers, or any other mechanism.

The cron task has the following configurable parameters. All are optional.

**Cron Task Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXITCLASS</td>
<td>Java exit class that allows for the manipulation of data before it is written to inbound queue.</td>
</tr>
<tr>
<td>ENDPPOINT</td>
<td>End point associated with the interface table. Defaults to the predefined end point that points to the local database.</td>
</tr>
<tr>
<td>INTERFACENAME</td>
<td>Interface to be polled. The default (null) is all interfaces. If you specify a value for this parameter, you also must specify a value for EXTSYSNAME parameter.</td>
</tr>
<tr>
<td>TARGETENABLED</td>
<td>Optional Boolean flag that controls whether the cron task runs in a specific Maximo application server. The default is 0 (false).</td>
</tr>
<tr>
<td>EXTSYSNAME</td>
<td>External system to be polled.</td>
</tr>
<tr>
<td>QUEUETABLE</td>
<td>Inbound interface queue table. The default is MXIN_INTER_TRANS.</td>
</tr>
</tbody>
</table>
External System Requirements

To use interface tables, you must create the tables and configure the IFACETABLECONSUMER cron task. For information about these activities, see Chapter 6, "Basic Configuration," on page 6-1.

You also must also program the external system(s) to read and write to interface queue tables and interface tables. Use this section as a checklist for preparing to use interface tables.

**Overall Processing**
- Have procedures in place to restore the backups of the interface tables.
- Create interface tables via the user interface.
- Manage archiving of interface tables.

**Inbound Processing**
- Create and initialize the outbound TRANSID counter.
- For each interface table that an inbound transaction writes to:
  - Build an interface table record and populate it with the following information:
    - The transaction data
    - The incremented TRANSID value
    - If multiple records for the same interface table, the incremented TRANSSEQ value
  - Build an MXIN_INTER_TRANS queue record with the following information:
    - The same TRANSID value as in the interface table record
    - The name of the interface that corresponds to the interface table, in the IFACENAME column
    - The ACTION value (optional)
    - The identifier of the external system, in the EXTSYSNAME column
  - Perform a single commit to commit all records for a transaction at one time.

**Inbound Null Columns**

**NOTE** If a column in an inbound interface table contains a null value, the applicable MBOs process the column as follows:

By default, the field is not updated by the transaction.

If you add the empty tag via a user exit, the MBO nulls the field in the Maximo database.
External System Requirements

**Outbound Processing**

▼ Set up a process to retrieve outbound interface table transactions using the MXOUT_INTER_TRANS queue table. This can be done via a polling program, triggers, or any other mechanism.

▼ For the polling program to process transactions sequentially, complete the following steps:

- Read the records in the MXOUT_INTER_TRANS queue table in TRANSID sequence.

- For each record in the MXOUT_INTER_TRANS queue table:
  
  ▼ Determine the name of the interface table that corresponds to the IFACENAME (interface) value in the queue table. For a list of corresponding predefined interfaces and interface table names, see Appendix A, "Maximo Adapter Interface Components," on page A-1.

  ▼ Access the interface table that you just identified, and retrieve the first record whose TRANSID value matches the TRANSID value in the current MXOUT_INTER_TRANS queue record. If the interface table contains multiple records with the same TRANSID value, retrieve and process them in TRANSSEQ sequence.

  ▼ Process according to the value in the ACTION column of the interface table queue.

- Commit all records for a single database transaction at the same time.

- Delete the current record from the MXOUT_INTER_TRANS queue table.

▼ Implement error management, based on your external system’s requirements.
This chapter explains the steps in configuring Maximo for basic integration processing. For information about advanced configuration options and customization, see the Advanced Topics section of this guide.

This chapter is addressed to the system administrator. It includes the following sections:

- Assumptions
- Predefined Components and Queues
- Configuration Checklist
- Prerequisite Activities
- Configuring Integration Administration Information
- Configuring and Enabling the JMS Queues
- Configuring External System and Interfaces
- Configuring External System Interface Controls
- Enabling the External System
- Restarting the Maximo Application Server
- Creating Interface Tables
- Configuring the Data Export Feature
- Configuring the Data Import Feature
Assumptions

This chapter assumes that you created and configured the JMS queues on your application server and accepted the default queue settings presented during Maximo installation.

NOTE If you did not create and configure the JMS queues during Maximo installation, do so now. See Appendix D, "Creating the JMS Queues," on page D-1.

Predefined Components and Queues

Maximo provides the following predefined integration components:

- An external system (EXTSYS1)
- An adapter (Maximo)
- Integration Objects
- Integration Points
- Interfaces

In addition, the installation process described in the Maximo Installation Guide builds the following JMS queues:

- Inbound sequential (sqin)
- Inbound continuous (cqin)
- Outbound sequential (sqout)

CAUTION Do not delete external system EXTSYS1 or any of its interfaces. Create a copy of external system EXTSYS1 and use the copy to configure your interfaces.

The predefined system integration objects, integration points, and interfaces are required during the installation of program patches and other adapters from IBM Corporation. You cannot disable or update system components.

You can use the predefined queues for development and production. This chapter assumes that you will use the predefined queues and default settings, and not change the name of any queue.
Configuration Checklist

The table on the following pages lists basic configuration activities. The codes in the checklist have the following meanings.

*Configuration Checklist Codes*

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>Required</td>
</tr>
<tr>
<td>N</td>
<td>Not applicable</td>
</tr>
<tr>
<td>O</td>
<td>Optional</td>
</tr>
</tbody>
</table>

For details about each activity, refer to the page in the *Refer to* column.
## Configuration Checklist

### Checklist of Configuration Activities

<table>
<thead>
<tr>
<th>Activity</th>
<th>General Integration</th>
<th>Data Import</th>
<th>Data Export</th>
<th>Web Services</th>
<th>Comments</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure administration information</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Set Maximo user name for inbound processing</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>update requires restarting Maximo server</td>
<td>6-7</td>
</tr>
<tr>
<td>Set Global Directory Location</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>update requires restarting Maximo server</td>
<td>6-7</td>
</tr>
<tr>
<td>Set Sendersysid Update on Export</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>recommended</td>
<td>6-7</td>
</tr>
<tr>
<td>Set domain creation</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>recommended</td>
<td>6-7</td>
</tr>
<tr>
<td>Set e-mail addresses</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>recommended; required for e-mail error notification</td>
<td>6-7</td>
</tr>
<tr>
<td>Set mail server host property</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>required for e-mail error notification; update requires restarting Maximo server</td>
<td>6-7</td>
</tr>
<tr>
<td>Configure and enable the JMS queues</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create queues on the application server</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td></td>
<td>6-8</td>
</tr>
<tr>
<td>Define queue properties in Maximo</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Not applicable if using default settings</td>
<td>6-8</td>
</tr>
<tr>
<td>Enable the cron task for the sequential queues</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Applies only if using sequential queues</td>
<td>6-10</td>
</tr>
<tr>
<td>Enable the message beans for the continuous queue</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Applies only if using inbound continuous queue</td>
<td>6-12</td>
</tr>
<tr>
<td>Configure external system and interfaces</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create external system</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td>6-13</td>
</tr>
<tr>
<td>Identify the queues to the external system</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Applies only if not using predefined queues</td>
<td>6-14</td>
</tr>
<tr>
<td>Task</td>
<td>General</td>
<td>Data Import</td>
<td>Data Export</td>
<td>Web Services</td>
<td>Comments</td>
<td>Refer to</td>
</tr>
<tr>
<td>---------------------------------------------------------------------</td>
<td>---------</td>
<td>-------------</td>
<td>-------------</td>
<td>--------------</td>
<td>---------------------------------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>Configure end point</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Applies only to interface tables and outbound processing</td>
<td>page 6-15</td>
</tr>
<tr>
<td>Assign end point to external system</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Applies only to interface tables and outbound processing</td>
<td>page 6-17</td>
</tr>
<tr>
<td>Disable unneeded interfaces</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td></td>
<td>page 6-19</td>
</tr>
<tr>
<td>Add outbound and inbound interfaces to external system</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td>page 6-20</td>
</tr>
<tr>
<td>Specify inbound queues</td>
<td>O</td>
<td>O</td>
<td>N</td>
<td>N</td>
<td>Applicable to inbound processing only</td>
<td>page 6-21</td>
</tr>
<tr>
<td>Enable outbound integration events</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Applicable to outbound processing only</td>
<td>page 6-17</td>
</tr>
<tr>
<td>Configure external system interface controls</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Change system level default values</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td></td>
<td>page 6-23</td>
</tr>
<tr>
<td>Set organization- and site-level values</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>Applies only to controls configured for organization- or site-level settings</td>
<td>page 6-28</td>
</tr>
<tr>
<td>Enable the external system</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td>page 6-32</td>
</tr>
<tr>
<td>Restart the Maximo application server</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>Required after updating global directory location or mail server host property</td>
<td>page 6-32</td>
</tr>
<tr>
<td>Create interface tables</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create tables</td>
<td>O</td>
<td>N</td>
<td>O</td>
<td>N</td>
<td>Required if using interface tables. Interfaces tables not available with Web services</td>
<td>page 6-33</td>
</tr>
<tr>
<td>Enable the cron task</td>
<td>O</td>
<td>N</td>
<td>O</td>
<td>N</td>
<td>Required if using interface tables</td>
<td>page 6-35</td>
</tr>
</tbody>
</table>
Prerequisite Activities

Before you proceed with the configuration process, you must make the following decisions about how you will use Maximo Integration:

- the inbound and outbound integration points that you will use
- the end point(s) to which Maximo will send outbound data
- the queue(s) that inbound interfaces will use

Configuring Integration Administration Information

The Integration Administration dialog box contains several properties, some of which have default values that you should consider updating. They are:

- Maximo user name
- Global directory
- Update SENDERSSID setting
- Domain values setting
- Administrator E-mail Address
- Sender E-mail address

The following section describes these properties. To update the properties in the Integration Administration dialog box, complete the following steps:

1. In the External Systems application, select any system.

2. From the Select Action menu, select **Integration Administration Setup**. The Integration Administration Setup dialog box appears.

3. Update the applicable field(s). For information about a field, press Alt+F1.

4. Click **OK**.
Maximo User Name

The Maximo user name is the user ID that is used in all inbound integration transactions. If you do not use the default value, enter a value that points to a valid Maximo user with a status of ACTIVE.

Global Directory

The Global Directory is the name of the directory to be used for all generated schema files, XML files, and error files. Those files are in predefined directories within the global directory. By default, this property has no value and the files are located in the same directory as the Maximo processing logs. If you specify an alternate location, it must be accessible from all Maximo application servers.

**NOTE** You must restart the Maximo application server after changing this property.

Update SENDERSYSID Setting

**Update SENDERSYSID on Data Export?** specifies whether Maximo writes the Maximo system identifier (the value of MAXVARS.MXSYSID) to the SENDERSYSID field when it generates an outbound transaction via the Data Export feature. The default value is no.

This setting applies to outbound transactions only.

Domain Values Setting

The Interface creates Domain values? setting specifies whether inbound interface processing automatically adds the value of a field to an ALN or Numeric domain, if the value is not already in the domain. The default value is yes.

This setting applies to inbound transactions only.

Administrator and Sender E-mail Addresses

The Integration Administration dialog box contains two e-mail addresses, the Administrator E-mail Address and the Sender E-mail Address.

Though both addresses are optional, integration processing cannot send e-mail error notifications unless both fields contain an address.

Mail Server Host Property

In order for the system administrator to receive e-mail notification of errors, you must configure the name of the host running the SMTP server. If you configured this value during the Maximo installation, ignore this section. Otherwise, edit the mail.smtp.host property in the maximo.properties file, then rebuild the EAR file.
Configuring and Enabling the JMS Queues

The Maximo installation procedure installs the following JMS queues.

**Predefined JMS Queues**

<table>
<thead>
<tr>
<th>Queue Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cqin</td>
<td>Continuous inbound</td>
</tr>
<tr>
<td>sqin</td>
<td>Sequential inbound</td>
</tr>
<tr>
<td>sqout</td>
<td>Sequential outbound</td>
</tr>
</tbody>
</table>

Configuring the queues involves the following activities:

- Defining queue properties in Maximo
- Enabling the cron task for the sequential queues
- Enabling the message bean for the continuous queue

Perform these activities only for the queues that you will use. Web services processing does not use the JMS queues. All other integration processes, including data import and data export, require one or more JMS queues.

**CAUTION**

If you have not yet created and configured the JMS queues, do so now. See Appendix D, "Creating the JMS Queues," on page D-1.

**Queue Properties**

The Add/Modify Queues dialog box displays the predefined settings for the queue properties. You do not need to change any of these values, although you might consider updating the Maximum Try Count value.

The Maximum Try Count, which is not available if you are using the WebLogic application server, is the number of times Maximo tries to process a message before writing the message to the error log and sending an e-mail notification to the system administrator. For this release, valid values for the WebLogic application server are 0, 1, and 2. For WebSphere, valid values are any number equal to or greater than zero.
To update this value, complete the following steps:

1. In the External Systems application, select **Add/Modify Queues** from the Select Action menu. The Add/Modify Queues dialog box displays the predefined queues.

2. To update a queue, click View Details to the left of the queue name. The Row Details appear.

**Add/Modify Queues Dialog Box with Row Details**

3. Update the **Maximum Try Count** field.

4. Click **OK**.
Enabling the Cron Task for the Sequential Queues

The default settings direct the JMSQSEQCONSUMER cron task to poll the outbound queue, sqout, and the sequential inbound queue, sqin, every 30 seconds. You must activate the applicable instance(s) of the cron task or inbound and outbound messages will remain unprocessed in the queues.

If you will not use one or both of these queues, do not activate these cron task instances.

To configure and activate the cron task, complete the following steps:

1. Go to the Cron Task Setup application in the Configuration module.
2. On the List tab, enter JMSQSEQCONSUMER in the Cron Task field and select the Cron Task tab.

Selection of JMSQSEQCONSUMER Cron Task
To update the cron task, perform one or both of the following actions:

- To update the cron task that polls the inbound sequential queue, click View Details to the left of cron task instance SEQQIN.

- To update the cron task that polls the outbound sequential queue, click View Details to the left of cron task instance SEQQOUT.

**Cron Task Tab with Row Details**

3 To change the polling frequency, update the **Schedule** field.

**NOTE** The **Run as User** value does not apply to this cron task.

5 Update the Cron Task Parameters, as needed. For information about the parameters, see Chapter 11, "JMS Queue Configuration," on page 11-1.

6 To activate the cron task instance, select the **Active?** check box.

7 Click Save Cron Task Definition.

For more information, refer to the online help for the Cron Task Task Setup application.
Enabling the Message Beans for the Continuous Queue

If you will not use the continuous inbound queue, cqin, disregard this section.

If you selected the **Enable Maximo Enterprise Adapter** check box during the Maximo installation, the message beans for the inbound continuous queue were automatically enabled. If this option was not selected, or if you are unsure if this option was selected, perform/verify the following updates to the deployment-application.xml and application.xml files. The files are in the applications\maximo\META-INF directory under the Maximo root directory.

**deployment-application.xml**

Change from:  
```xml
<!-- JMS MDB is not deployed by default
<module id="EjbModule_1077124925237">
  <ejb>meajmsejb.jar</ejb>
</module>
-->
```

Change to:  
```xml
<!-- JMS MDB is not deployed by default-->
<module id="EjbModule_1077124925237">
  <ejb>meajmsejb.jar</ejb>
</module>
```

**application.xml**

Change from:  
```xml
<!-- JMS MDB is not deployed by default
<module>
  <ejb>/meajmsejb/ejbmodule</ejb>
</module>
-->
```

Change to:  
```xml
<!-- JMS MDB is not deployed by default -->
<module>
  <ejb>/meajmsejb/ejbmodule</ejb>
</module>
```

**NOTE**

If you update the files, rebuild and redeploy the EAR file. For more information about rebuilding and redeploying EAR files, refer to the IBM Maximo System Administrator's Guide.

If you use a WebLogic application server, navigate to the WebLogic administrative console and ensure that the meajmsejb module is targeted to the correct server.
Configuring External System and Interfaces

On the System tab in the External Systems application, you perform the following configuration activities:

- Create an external system
- Identify the queues to the external system
- Create/configure an end point
- Assign the end point to an external system
- Enable outbound integration events

On the Inbound and Outbound Interfaces tabs in the External Systems application, you perform the following configuration activities:

- Disable unused interfaces
- Associate inbound and outbound interfaces with the external system
- Specify the inbound JMS queue

Creating an External System

The easiest way to create an external system is to duplicate an existing one. Duplicating an external system copies the interfaces, interface controls, end point, and queues from the existing system. You can then modify those entities as needed.

**CAUTION**

Do not modify external system EXTSYS1 in any way.

To duplicate an external system, complete the following steps:

1. In the External Systems application, display the system that you want to duplicate.

2. From the Select Action menu, select **Duplicate External System**. The application displays a copy of the original external system data, without a value in the **System** field.

3. Enter a unique value in the **System** field.

4. Click **Save External System**.
Configuring External System and Interfaces

Identifying the Queues to the External System

The System tab in the External Systems application identifies the inbound (sequential and continuous) and outbound queues that the external system will use. If you duplicated the default system, EXTSYS1, the default queue names appear on the system tab for your new system, and you do not have to update the queue information.

**System Tab in External Systems Application**

![System Tab in External Systems Application](image)

**NOTE** If you will not use all three queues, it makes no difference if you clear the names of the unused queues from the System tab or leave them there.
Creating or Configuring an End Point

If you will process outbound interfaces, you must specify the end point (output file location), to which the transactions will be delivered and the handler that will route the transactions to the end point. Maximo provides the following predefined end points and handlers:

*Predefined End Points*

<table>
<thead>
<tr>
<th>End Point</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MXFLATFILE</td>
<td>Receives ASCII data in the form of rows and columns; implements the FLATFILE handler.</td>
</tr>
<tr>
<td>MXIFACETABLE</td>
<td>Receives data in an interface table; implements the IFACETABLE handler.</td>
</tr>
<tr>
<td>MXXMLFILE</td>
<td>Receives data in XML format; implements the XMLFILE handler.</td>
</tr>
</tbody>
</table>

*Predefined Handlers*

<table>
<thead>
<tr>
<th>Handler</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EJB</td>
<td>Delivers outbound messages as an XML document to an Enterprise JavaBean (EJB) executing in the local application server or a remote application server.</td>
</tr>
<tr>
<td>FLATFILE</td>
<td>Delivers Maximo data into a flat file whose location is configurable.</td>
</tr>
<tr>
<td>HTTP</td>
<td>Delivers outbound messages as an XML document to a URL over the HTTP or HTTPS protocols.</td>
</tr>
<tr>
<td>IFACETABLE</td>
<td>Delivers outbound messages into the appropriate interface table in a relational database.</td>
</tr>
<tr>
<td>JMS</td>
<td>Delivers outbound messages as an XML document into the appropriate queuing system that has been enabled through Java Messaging Service (JMS).</td>
</tr>
<tr>
<td>WEBSERVICE</td>
<td>Delivers outbound XML messages as an XML document to a Web services component using SOAP over HTTP.</td>
</tr>
<tr>
<td>XMLFILE</td>
<td>Delivers outbound messages as an XML document to the designated file in the local machine or a shared network folder.</td>
</tr>
</tbody>
</table>

*NOTE*  With the exception of interface tables, end points apply only to outbound transactions.
Configuring External System and Interfaces

The first procedure below describes the generic assignment of a handler to an end point. The second procedure describes how to create an HTTP end point, using the generic HTTP handler.

To select the handler that your external system will use, complete the following activities.

1. Select **Add/Modify End Point** from the Select Action menu.

2. Click View Details to the left of the end point to which your system will send outbound data. The Row Details open.

Add/Modify End Point Dialog Box with Row Details

3. In the **Handler** field, enter the name of the handler that will deliver transactions to the selected end point, or click Select Value. Depending on the handler that you select, the Properties for End Point table window might display a list of properties you must define for the combination of end point and handler.

4. If the end point and handler combination requires property values, enter those values. For a description of the properties required by each handler, see Chapter 9, "Router," on page 9-1.

5. Click OK.

To create an HTTP end point using the predefined HTTP handler, complete the following activities.

1. Select **Add/Modify End Point** from the Select Action menu.

2. Click **New Row**.

3. In the **End Point** field, enter a unique name.

4. In the **Handler** field, enter HTTP. The Properties for End Point table window will display the properties you must define for the end point and handler combination. For a description of the properties required by the HTTP handler, see Chapter 9, "Router," on page 9-1.

5. Click OK.
Assigning the End Point to an External System

The System tab in the External Systems application identifies the end point associated with that system. If you duplicated the default system, EXTSYS1, the default end point is MXXMLFILE. If you are using a different end point, update the End Point field on the System tab.

System Tab in External Systems Application

Enabling Outbound Integration Events

Enabling integration event listeners directs Maximo to automatically build and process related outbound interfaces when certain Maximo user actions occur.

An integration event is associated with an integration point. A user action in Maximo (for example, the update of an item via the Maximo Item application) triggers the event listener, which in turn triggers the processing of the related interface(s). By default, integration events are disabled.

**NOTE** If you plan to send all outbound data through the Data Export feature, do not enable outbound event listeners. Integration Event listeners do not apply to inbound transactions.
Configuring External System and Interfaces

To enable an outbound integration event listener, complete the following steps:

1. In the External Systems application, select the system that you want to update.

2. Select **Enable/Disable Integration Events** from the Select Action menu.

**Enable/Disable Integration Events Dialog Box**

3. In the Enable/Disable Integration Events dialog box, select the **Event Enabled?** check box to the right of the integration point(s) for which you want to enable the outbound event listener.

4. Click **OK**.
Disabling Unneeded Interfaces

When you duplicate an external system, all the interfaces provided by the corresponding adapter are enabled and available for use by the new system. You can see the interfaces on the Inbound Interfaces and Outbound Interfaces tabs in the External Systems application. If you will not use all the interfaces, disable the unneeded ones. To do so, complete the following steps:

1. In the External Systems application, select the external system you want to update.
2. On the Inbound Interfaces tab or the Outbound Interfaces tab, clear the Enabled? check box to the right of the interface(s) that you want to disable.

Inbound Interfaces Tab in External Systems Application

3. Click Save External System.
Associating Inbound and Outbound Interfaces

When you create an external system rather than duplicate an existing one, the interfaces within the corresponding adapter are available for use, but you must manually associate them with the new external system. You do this separately for inbound and outbound processing. To do so, complete the following steps:

1. In the External Systems application, select the system that you want to update.

2. On the Inbound Interfaces tab or the Outbound Interfaces tab, click New Row.

3. In the Interface field, click Detail Menu, then click Select Value. The Select Value dialog box displays the available inbound or outbound interfaces.

4. Select the interface that the new system will use in the selected direction (inbound or outbound). The selected interface will appear on the Inbound Interfaces or Outbound Interfaces tab.

5. To enable the interface, select the Enabled? check box to the right of the interface name.

6. Repeat steps 2 through 5 for each interface to be used in the selected direction.

7. Click Save External System.
Specifying the Inbound Queues

By default, inbound non-system interfaces use the continuous inbound queue and system interfaces use the sequential inbound queue. The primary differences between the two types of queues are:

- The sequential queue processes transactions in FIFO order and stops processing when it encounters an error in a transaction. Use this queue to process interfaces that are dependent upon the successful processing of previous interfaces.

- The continuous queue processes messages in a multi-threaded mode and continues processing after it encounters an error in a transaction. Use this queue to load data that is not dependent upon the successful processing of previous interfaces.

Within an external system, you can use the sequential queue for some interfaces and the continuous queue for others, or you can use one queue for all the inbound interfaces.

To change the queue that an inbound interface uses, complete the following steps:

1. In the External Systems application, select the system that you want to update.

2. Click the Inbound Interfaces tab. The application displays the interfaces associated with the system.
3 Perform one of the following actions:

- To write messages to the sequential inbound queue, clear the Use Continuous Queue? check box to the right of the applicable interface(s).

- To write messages to the continuous inbound queue, select the Use Continuous Queue? check box to the right of the applicable interface(s).

4 Click Save External System.

For more information about the inbound and outbound sequential queues, see Chapter 11, "JMS Queue Configuration," on page 11-1.
Configuring External System Interface Controls

Maximo provides system-level default values for the interface controls that are provided by the Maximo adapter and used by its predefined processing classes and processing rules. You might want to change the system level default values, assign override values at the organization or site level, or configure multiplication controls to create copies of inbound transactions for multiple Maximo organizations or sites.

You do not have to configure controls associated with interfaces that you are not using. To find out which interfaces use which predefined interface controls, see Appendix B, "Maximo Adapter Interface Specifications," on page B-1.

**NOTE**  Most, but not all, predefined interface controls allow for overrides at the organization or site level. By default, all controls use the system-level default if no organization- or site-level value is specified.

Changing System-Level Default Values

The procedure for changing a control’s system level default value varies, depending on the type of control (value, Boolean, list, or cross-reference). In all cases, you begin by performing the following actions:

1. In the External Systems application, select the system that you want to update.
2. On the Interface Controls tab, select the subtab that corresponds to the control type (value, Boolean, list, or cross reference).

**Interface Controls Tab in External Systems Application**

The subtab displays any predefined controls of the selected type. If the subtab does not display any controls, none exist for that type.
Configuring External System Interface Controls

Value Type Controls

To override the system-level default value for a value type control, complete the following steps:

1. On the Interface Controls tab in the External Systems application, select the Value subtab.

2. Click View Details to the left of the control that you want to update. The Row Details open.

Value Subtab with Row Details

3. Update the value in the Value field.

4. Click Save External System.
**Boolean Type Controls**

To override the system-level default value for a Boolean type control, complete the following steps:


2. Click View Details to the left of the control that you want to update. The Row Details open.

**Boolean Subtab with Row Details**

3. Select or clear the **True?** check box.

4. Click Save External System.
List Type Controls

You cannot overwrite an existing value for a list type control, but you can delete predefined values and add new values.

To override the system-level default value for a list type control, complete the following steps:

1. On the Interface Controls tab in the External Systems application, select the List subtab.

2. Click View Details to the left of the control that you want to update. The Row Details open and the default value(s) appear in the Values for Control table window.

List Subtab with Row Details

3. To delete an existing value, click Mark Row for Delete to the right of the value(s). You can mark multiple rows for deletion.

4. To add additional values to the control, perform the following actions:
   a. In the Values for Control table window, click New Row.
   b. Enter a value in the Value field.

5. Click Save External System.
In a cross-reference type control, a one-to-one relationship generally exists between an external system value and a Maximo value. Inbound integration processing converts an inbound transaction’s external system value to the corresponding Maximo value, and outbound integration processing converts an outbound transaction’s Maximo value to the corresponding external system value.

A multiplication control is a type of cross-reference control that allows for a one-to-many relationship between an external system value and multiple Maximo organizations or sites. When a multiplication control is associated with an interface and integration point, inbound integration processing creates a copy of the inbound transaction for each Maximo organization or site that corresponds to the external system value. You identify the control as a multiplication control in the Integration Interfaces application, on the Inbound Integration Points subtab of the Interface tab.

**NOTE** Multiplication controls apply to inbound transactions only. They can be used only to multiply a transaction for multiple organizations or sites. You cannot use them in processing rules.

To override the system-level default value for a cross-reference type control, complete the following steps:

2. Click View Details to the left of the control that you want to update. The Row Details open and the default values appear in the Values for Control table window.
3. Update the value in the **Maximo Value** and/or **External Value** fields.
4. Click Save External System.
Assigning Organization and Site Level Values

Depending on the definition of an interface control, you might be able to assign values at the organization or site level. When you look at a control in the user interface, you will see the following check boxes; a check indicates the corresponding action is allowed.

- Organization Override?
- Site Override?
- Use System Default?

A check in the Use system default? check box indicates that Maximo uses the default system-level value if no match exists for the organization or control.

Organization override and site override are mutually exclusive. You cannot change these settings.

The procedure to assign organization- and site-level values varies, depending on the type of control (value, Boolean, list, or cross-reference).

**Value Type Controls**

To add an organization- or site-level value for a value type control, complete the following steps:

1. On the Interface Controls tab in the External Systems application, select the Value subtab.

2. Click View Details to the left of the control that you want to update. The Row Details open.

3. In the Override Values for Control table window, click New Row.

**Value Subtab with Override Fields**
Configuring External System Interface Controls

**Basic Configuration**

4 Perform one of the following actions:

- To enter an organization-level value, enter a value in the **Organization** field, or click Select Value.

- To enter an site-level value, enter values in the **Organization** and **Site** fields, or click Select Value.

5 Enter a value in the **Value** field.

6 Click Save External System.

**Boolean Type Controls**

To add an organization- or site-level value for a Boolean type control, complete the following steps:

1 On the Interface Controls tab in the External Systems application, select the Boolean subtab.

2 Click View Details to the left of the control that you want to update. The Row Details open.

3 In the Override Values for Control table window, click **New Row**.

**Boolean Subtab with Override Fields**

4 Perform one of the following actions:

- To enter an organization-level value, enter a value in the **Organization** field, or click Select Value.

- To enter an site-level value, enter values in the **Organization** and **Site** fields, or click Select Value.

5 Clear or select the **True?** check box.

6 Click Save External System.
List Type Controls

To add an organization- or site-level value for a list type control, complete the following steps:

1. On the Interface Controls tab in the External Systems application, select the List subtab.
2. Click View Details to the left of the control that you want to update. The Row Details open.
3. In the List Type Interface Controls table window, click **Override**. The Override Values for List Controls dialog box appears.

**List Subtab with Override Fields**

4. In the Organization/Site Override for List Type Controls table window, click **New Row**.
5. Perform one of the following actions:
   - To enter an organization-level value, enter a value in the **Organization** field or click Select Value.
   - To enter an site-level value, enter values in the **Organization** and **Site** fields or click Select Value.
6. In the Values for Organization/Site table window, click **New Row**.
7. Enter a value in the **Value** field.
8. Click **OK**.
To add an organization- or site-level value for a cross-reference type control, complete the following steps:


2. Click View Details to the left of the control that you want to update. The Row Details open.

3. In the Cross Reference Type Interface Controls table window, click Override. The Override Values for Cross Reference Controls dialog box appears.

4. In the Organization/Site Override for Cross Reference Type Controls table window, click New Row.

5. Perform one of the following actions:
   
   ▼ To enter an organization-level value, enter a value in the Organization field or click Select Value.
   
   ▼ To enter an site-level value, enter values in the Organization and Site fields or click Select Value.

6. In the Values for Organization/Site table window, click New Row.

7. Enter values in the Maximo Value and External Value fields.

8. Click OK.
Enabling the External System

By default, external systems are not enabled. Enable your external system(s) after you have completed all other configuration activities.

To enable the external system, complete the following steps:

1. In the External Systems application, select the system that you want to update.
2. On the System tab, select the Enabled? check box.
3. Click Save External System.

Restarting the Maximo Application Server

If you changed either of the following fields while configuring Maximo for integration, you must restart the Maximo application server for your changes to take effect:

- Global Directory Location in the Integration Administration dialog box
- mail.smtp.host properties in the maximo.properties file

If you changed the maximo.properties file, you also must rebuild the EAR file. For more information, refer to the IBM Maximo System Administrator's Guide.
Creating Interface Tables

If you will use interface tables instead of, or in addition to, XML messages, you also must perform the following additional configuration activities:

- Create the interface tables
- Configure cron task

You also must program the external system to read and write transactions from and to the tables. For details, see Chapter 5, "Interface Tables," on page 5-1.

**NOTE** Whether you use the interface tables to process inbound transactions, outbound transactions, or both, you must configure the appropriate end point. Do this before you create the interface tables, as the table creation process references the end point.

Creating the Tables

To create the interface tables, complete the following steps:

1. In the External Systems application, select the system that you want to update.

2. Select **Create Interface Tables** from the Select Action menu.

*Create Interface Tables Dialog Box*
Creating Interface Tables

3 In the Create Interface Tables dialog box, enter MXIFACETABLE in the **End Point** field, or click Select Value. The application displays the interface tables that you can create.

**List of Possible Interface Tables**

4 In the Interface Tables table window, click the Select Row check box to the left of the interface(s) for which you want to create interface tables. To create interface tables for all the interfaces displayed in the current dialog box, click the Select All Records check box next to the Interface Table Name column header.

5 Click **Create**. The Create Table Confirmation dialog box asks for confirmation.

6 Click **OK** to continue, or **Cancel** to stop, the table creation. If you click **OK**, a message box appears and asks you to wait. Depending on the number of interface tables you are creating, the process might take some time. After the creation of the table(s), a confirmation message appears.

7 Click **OK**.
Creating Interface Tables

Configuring the Cron Task for the Interface Tables

You must configure and activate the cron task that polls the interface tables.

To configure and activate the cron task, complete the following steps:

1. Go to the Cron Task Setup application in the Configuration module.

2. On the List tab, enter IFACETABLECONSUMER in the Cron Task field and select the Cron Task tab.

**Cron Task Tab in Cron Task Setup Application**

3. Click New Row.

4. Enter values in the Cron Task Instance Name and Schedule fields.

   **NOTE** The Run as User value does not apply to this cron task.

5. Update the Cron Task Parameters, as needed. For information about the parameters, see Chapter 5, "Interface Tables," on page 5-1.

6. To activate the cron task instance, select the Active? check box.

7. Click Save Cron Task Definition.

For more information, refer to the online help for the Cron Task Setup application.
You can use the Maximo Data Export feature even if you do not have real time integration enabled. The Data Export feature lets you perform a bulk export of transaction data from Maximo to an external system.

**NOTE**  
The Data Export feature is available only for data synchronization (operation = Notify) interfaces defined within an internal adapter. You cannot use flat files to export system interfaces.

Before you use the Data Export feature, read the following sections of this guide to gain a basic understanding of the way in which outbound interfaces are configured and processed:

- Chapter 2, "Architecture," on page 2-1
- "Outbound Integration Processing," on page 3-2

You also must perform several configuration activities. For a list of configuration requirements, see the Data Export column of the configuration checklist on page 6-4.

The export takes place at the external system-interface level; that is, you initiate the process for each combination of interface and external system. To initiate the export, you create a query to select records. The export performs the standard outbound processing for the selected interface, on the result set of the query.

If the standard processing executes processing classes or rules that exclude or modify records or fields, you might need to create and use a new integration point that does not filter or modify records.

**NOTE**  
To avoid memory limitations when exporting a large number of records, specify the following property in the maximo.properties file:

```
mxe.mea.nombocache=true
```

The default value of this property (if not specified) is false. If this property is enabled, MBO rules cannot be used for that interface. The MEA will not cache the MBOS and will release them during export.

For each interface that you want to export, complete the following steps:

1. In the External Systems application, select the external system to which Maximo will export the data.

2. On the Outbound Interfaces tab, select the interface that you want to export.

3. Click **Data Export**.
The Data Export dialog box appears.

**Data Export Dialog Box**

4. In the **Integration Point** field, enter the name of the corresponding integration point, or click Select Value.

5. In the **Export Condition** field, enter a properly constructed SQL query (optional). The query must be against the primary (top-level) MBO in the integration object.

   **CAUTION** To avoid processing problems due to the size of the XML transaction, use the Export Condition to filter out unnecessary records.

6. Click **OK**.
The Data Import feature lets you perform a bulk import of transaction data from an external system to Maximo.

**NOTE**
The Data Import feature is available only for data synchronization (operation = Notify) interfaces defined within an internal adapter. You cannot use flat files to import system interfaces.

Before you use the Data Import feature, read the following sections of this guide to gain a basic understanding of the way in which inbound interfaces are configured and processed:

- Chapter 2, "Architecture," on page 2-1
- "Inbound Integration Processing," on page 3-16

You also must perform several configuration activities. For a list of configuration requirements, see the Data Import column of the configuration checklist on page 6-4.

The data that you import must be in a delimited flat file or Maximo XML format. The flat file data format must be the same as that of the interface table that corresponds to the interface. The Data Import feature can load predefined or user-defined interfaces.

**Flat File Restrictions**
If the data to be imported is in a flat file format, review the column names within the corresponding integration object to determine if duplicates exist. If they do, modify the Alias attribute for the duplicate columns, to ensure that all column names are unique and Maximo can generate the flat file without errors. For more information about this, see “Duplicate Columns and Aliases” on page 5-6.

If you use multiple languages for translatable columns, any one file can contain only one language.

The FLATFILE handler encodes outbound flat files in the standard UTF-8 format; and the Data Import mechanism assumes that inbound flat files are encoded in UTF-8 format.

If the data to be imported contains the delimiter character or a double quote ("), you must format the data correctly. For more information, see “Flat File Formatting” on page 9-7.

**Error Handling**
If Maximo encounters a problem while importing a file, it generates a recovery file called recovery_filename.txt, which identifies the last record that was successfully imported. Correct the next record in the import file, then restart the data import process.

To start or restart the data import process, complete the following steps:

1. In the External Systems application, select the external system from which Maximo will import the data.

2. From the Select Action menu, select **Data Import**.
3 In the Data Import dialog box, do one of the following:

- To import a flat file, click **Flat File** and modify the value in the **Delimiter** field, if necessary.

- To import an XML file, click **XML File**.

*Data Import Dialog Box*

4 In the **Server Directory** field, enter the location of the file.

5 In the **File Name** field, enter the name of the file or click Select Value.

6 Click **Import**. Maximo begins the data import process. If the validation of the external system and interface fail, Maximo displays an error message.
This chapter discusses how Maximo manages errors it encounters while processing outbound and inbound interfaces and how system administrators should address those errors. It contains the following sections:

- Queue Error Handling
- Interface Table Error Handling
- Error Management Configuration
- Error Notification
- Error Management Folders
- Error Correction
Queue Error Handling

Maximo uses JMS queues as a staging mechanism. Inbound messages from external systems are first stored in a queue, then subsequently picked up for processing. Similarly, outbound messages are stored in a queue, from which they are picked up for dispatch to an external system.

Queue error handling takes place when an error occurs as a message is being processed from an inbound or outbound queue.

In outbound transactions, errors can occur as the message is being sent from the queue to the external system. These errors are typically due to a communication failure between the queue and external system or, if writing to interface tables or files, a problem with table or file space. In inbound transactions, errors can occur when Maximo tries to process a message into Maximo. They are typically due to business rule validations in the MBOs or in the inbound processing of Maximo.

The sequential queue processes messages one at a time, in FIFO sequence. When Maximo encounters an error in processing a message in a sequential queue, inbound or outbound, the error management mechanism is initiated and the message is flagged as having an error. Subsequent messages in the queue are not processed until the message in error has been resolved. As a result, only one error at a time can exist in a sequential queue.

The continuous queue processes messages in a multi-threaded mode. When an error is encountered in the continuous queue, the error management mechanism is initiated and the message is flagged as having an error. However, Maximo continues to process subsequent messages in the queue. As a result, multiple errors can exist in a continuous queue.

Depending on your system configuration, Maximo might make several attempts to reprocess the message before determining that the error requires intervention. At this point, Maximo performs the following activities:

- It sends a notification to a specified e-mail account, informing the recipient that an error has occurred.

**NOTE** For WebSphere environments, Maximo sends an additional e-mail message to the specified e-mail account each time you successfully restart your Maximo application server.

- It creates an error file that contains the original message and information about the error.
Interface Table Error Handling

There are two stages during the processing of inbound interface tables where errors can occur:

- Writing data from the interface table to the queue
- Processing data from the queue into Maximo

Maximo does not apply any business rules to the inbound interface table records as it writes them to the inbound JMS queue, so errors that occur at this point will be due to one of the following:

- The JMS queue is deactivated or freespace is unavailable.
- The interface name or external system name is invalid.
- The interface is not enabled for the external system.
- The external system is not enabled.

When an error occurs during inbound interface table processing, the polling program writes the exception trace in the IMPORTMESSAGE column of the MXIN_INTER_TRANS queue table. If this is the first error in the MXIN_INTER_TRANS queue, the system sends an e-mail notification to the Administrator E-mail Address specified in the Integration Administration Setup dialog box in the External Systems application.

The cron task continues to process subsequent records in the MXIN_INTER_TRANS queue table, then sleeps. When it resumes processing, it tries to process the records in error, as well as new records added to the MAX_INTER_TRANS queue table.

After sending an error notification, the cron task does not send notification of additional errors as long as the queue table contains one transaction marked in error. The assumption is that the person who was notified of, and is researching, the original error will see and correct additional errors when he or she looks at the queue table. After all current errors are corrected, the cron task will send a notification when it encounters a new error.

Any errors that occur after the cron task successfully passes an interface table transaction to an inbound queue will be managed via the error handling process for the queues, as discussed in “Queue Error Handling,” on page 7-2.
Error Management Configuration

Error management requires the configuration of several properties before use.

Integration Administration Setup Properties

Configure the following properties in the Integration Administration Setup dialog box within the External Systems application:

▼ Global Directory Location

This property specifies the root folder under which all integration configuration files are located. If this value is null, the folders are created under the directory from which the Maximo application server is started or from the current working directory of the Maximo application server.

This location must be an existing folder that is accessible by a given Maximo application server. In a clustered environment, all Maximo application servers must have access to this location.

If you change this value, you must restart the Maximo application server(s) in order for Maximo to recognize the new location.

▼ Administrator E-mail Address

This property specifies the primary address(es) to which Maximo sends notification of integration processing errors. You can enter multiple addresses, delimited with a comma (,).

You can optionally use the e-mail address property at the queue level to override this address, per queue. This option lets you specify different e-mail addresses for each queue. For more information, see “External Systems Configuration Properties,” on page 7-5.

**CAUTION**

If you do not configure e-mail address properties, there will be no e-mail notification when errors occur in processing messages from the queues.

▼ Sender E-mail Address

This property specifies the from address that will appear in error notifications that Maximo sends; for example, MXINTADM@ZZZ.com. Some SMTP servers require this to be a valid e-mail address format; some accept any value.
External Systems Configuration Properties

You configure the following error management properties in the Add/Modify Queues screen in the External Systems application:

▼ Maximum Try Count

This property specifies the number of times Maximo will try to reprocess an error. After the specified number of tries, Maximo notifies the system administrator (if configured to do so) and writes the message to an error file.

If the Maximum Try Count value is set to 0, there is no limit to the number of times Maximo retries the transaction. It notifies the system administrator and writes the message to the error file after the first unsuccessful attempt to process the transaction. This value is typically set to 0 for outbound queues.

**NOTE** For this release, valid values for the WebLogic application server are 0, 1, and 2. For WebSphere, valid values are any number equal to or greater than zero.

If the Maximum Try Count is 2, Maximo will not report an error until it retries the transaction and encounters an error a second time.

▼ E-Mail Address

This is an optional property that can be specified if you want e-mail error notification messages to be sent to different addresses for each queue. You can enter multiple addresses, delimited with a comma (,). The value in this property overrides the value in the Administrator E-mail Address property. If no value is specified, e-mail notifications will be sent to the e-mail address(es) specified for the Administrator E-mail Address property.

Maximo.properties Configuration

You configure the following error management properties in the maximo.properties file in the `<Maximo root>\applications\Maximo\properties` folder.

▼ mail.smtp.host

This property specifies the host running the SMTP server. This property is not unique to the Integration module, and may already have been configured for other Maximo applications.

**NOTE** If you change the maximo.properties file, you must rebuild and redeploy the Maximo EAR file.
Error Notification

When an inbound or outbound transaction results in an error in a queue, Maximo sends an e-mail notification to the system administrator only if no other errors are awaiting correction in the same queue. The notification informs the system administrator that the queue contains one or more transactions with errors. If multiple errors exist in the queue, the system administrator must resolve all of them before he or she will receive notification of new errors. A typical e-mail error message contains a Java error stack trace and the error file name with its full folder path information, relative to the server location.

Example

C:\error\jms.mro.int.queues.cqin\EXTSYS1\EXTSYS1_MXCOAInterface_11017328950785131.xml

The notification process works the same way regardless of the type of queue (continuous or sequential) or the processing direction (inbound or outbound) in which the error occurs, or whether Maximo is running in a clustered or non-clustered environment.

Example

Assume that the continuous inbound queue contains ten messages. Maximo successfully processes the first four, then encounters an error in the fifth. (Depending on the value of the Maximum Try Count property, Maximo might have retried the message one or more times.) This is the first error in the queue, so Maximo sends an e-mail notification to the system administrator, then continues to process subsequent messages in the queue. If Maximo encounters another error in the seventh message, it does not send another e-mail notification if the system administrator has not yet resolved the original error. If the system administrator has resolved the original error and no errors are pending, Maximo sends a new e-mail notification to the system administrator.

If the error were encountered in a sequential queue, the only difference in the processing just described is that Maximo would not process subsequent messages until the message with the error had been resolved.

NOTE

Multiple errors will exist only in the continuous inbound queue. In a clustered environment, the administrator might receive up to one e-mail error notification per Maximo server, depending upon the timing of the transactions in error.
Maximo employs an error management process that uses the following folders. Subsequent sections of this chapter describe each folder in more detail.

### Error Management Folders

<table>
<thead>
<tr>
<th>Folder</th>
<th>Contents</th>
<th>Owner</th>
</tr>
</thead>
<tbody>
<tr>
<td>error</td>
<td>Messages in which integration processing encountered an error</td>
<td>Maximo</td>
</tr>
<tr>
<td>retry</td>
<td>Corrected messages, to be reprocessed by Maximo</td>
<td>User</td>
</tr>
<tr>
<td>delete</td>
<td>Messages to be deleted</td>
<td>User</td>
</tr>
<tr>
<td>deleted history</td>
<td>Log of deleted messages</td>
<td>Maximo</td>
</tr>
</tbody>
</table>

These folders are created in the location specified in the Global Directory Location property. The Owner column in the preceding table identifies the agent responsible for placing messages into the corresponding directory.

Within each of these folders, messages are grouped by queue name; that is, subfolders exist for each queue defined in Maximo. Within each queue folder, subfolders exist for each external system. These subfolders do not exist by default; they are created by Maximo at the time an error is written for a specific queue-external system combination.

The folders have the following subfolder hierarchy:

```
folder name [error, retry, delete, deletedhistory]
  queue name
    external system name
      error file
      ...
```

### Error Folder

After Maximo has unsuccessfully tried to process a message from a queue the number of time specified in the Maximum Try Count property for that queue, it writes the message to the corresponding error folder location (`<GlobalDirectory>/error/<QueueName>/<ExternalSystemName>`). The original message remains in the queue.

The error file contains the error XML for the message. Maximo assigns the file a name, in the following format:

```
external system_interfacename_messageID.xml
```

**Example**

```
EXTSYS1_MXCOAInterface_11017328950785131.xml
```

**CAUTION**

Do not change the name of the error file, as Maximo uses this name in reprocessing the message.
The following is an example of an error XML file. It contains the following information:

- the error message, in the ERRORMESSAGE element
- the message from the queue, in the ER element
- the integration object XML, in the IR element

The IR element is present only for inbound transactions, and only if interface processing and user exit processing was successfully applied to the message. It represents the integration object created during interface and/or user exit processing.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ERROR>
  <ERRORMESSAGE>Error occurred while processing IR record CHARTOFACCOUNTS at number 1. Error is:
  Not a valid GL Account. The COST CENTER component value is not valid.</ERRORMESSAGE>
  <ER>
    <MXCOAInterface xmlns="http://www.mro.com/mx/integration" language="en">
      <Header operation="Notify" event="1">
      ...
      </Header>
      <Content>
        <MXCOA>
          <CHARTOFACCOUNTS action="Add">
          ...
          </CHARTOFACCOUNTS>
        </MXCOA>
      </Content>
    </MXCOAInterface>
  </ER>
  <IR>
    <MXCOA>
      <CHARTOFACCOUNTS action="Add">
      ...
      </CHARTOFACCOUNTS>
    </MXCOA>
  </IR>
</ERROR>
```

**NOTE**
The ER element in an error file created for inbound messages from interface tables or flat files has a flat (non-hierarchical) structure.

For more information about the ER and IR sections of the XML, see Chapter 15, "Customization with User Exits," on page 15-1.

ER and IR elements do not exist in the corresponding error file when the Maximo integration framework encounters critical errors. For more information about correcting critical errors, see “Error Correction,” on page 7-9.
Error Correction

The system administrator can manually change the contents of a message in error, then place the updated message in the retry folder. During its next cycle, Maximo tries to process the message in the retry folder. Only the <ER> element can be edited; the <IR> element is provided for information only and any change to the <IR> is ignored when the message is reprocessed.

**CAUTION**

Correcting an error in the XML file could potentially lead to a mismatch in data between the sending and receiving systems.

The retry process works the same way regardless of the type (continuous or sequential) or direction (inbound or outbound) of the queue in which the error occurred; or whether Maximo is running in a clustered or non-clustered environment.

Critical Errors

Critical errors are transaction processing exceptions that the Maximo error correction process cannot retry. Transaction processing exceptions can occur when invalid data, such as a special character, is in the XML file.

When the Maximo integration framework identifies a critical error, ER and IR sections in the corresponding error file are not present. To correct the critical error, remove the invalid data from the inbound queue. You then must place the error message into the delete folder. For more information about the delete folder, see “Error Management Folders,” on page 7-7.

You can see invalid data associated with a critical error in the 'view' folder in the Maximo integration global directory. For more information about researching critical errors, see “Researching Errors,” on page 7-10.

Common Causes of Errors

Errors that occur when processing a message from an outbound queue are usually related to the delivery of the message to the end point specified for the external system. Typical problems that can cause such an error are a disruption of the communication link between Maximo and the external system, or database table space or file space issues in the external system. Resolving an outbound error normally does not require modification of the XML message.

Errors that occur when processing a message from an inbound queue are usually related to a business rule validation in a MBO or in the inbound processing of the interface. Examples are:

- ▼ A validation failed because related data must first be updated in the Maximo application. In this case, an administrator can move the failed transaction to the retry folder after a user has made the necessary update to Maximo.

- ▼ A related inbound message must be processed first, because it contains data required by the message that failed. For example, an inbound general ledger interface that adds a GL account number to Maximo
Error Correction

must be processed before a receipt transaction that uses that account number. In this case, an administrator can move the failed message to the retry folder after the prerequisite interface is processed successfully.

▼ The message contains incorrect data. In this case, someone can manually update the XML, then the system administrator can move the updated message to the retry folder.

▼ The message contains invalid data in the form of a special character. To correct the critical error, remove the invalid data from the inbound queue. You then must place the error message in the delete folder.

CAUTION Correcting an error in the XML file could potentially lead to a mismatch in data between the sending and receiving systems.

Researching Errors

Upon receipt of the error notification, the system administrator should evaluate the error message in the e-mail and/or look at the XML file in the error folder for the specified queue and system. Depending on the type of queue (sequential or continuous) and the number of messages in the queue, the system administrator may see zero, one, or multiple error XML files in the error folder.

No XML Files Exist If no error XML file exists in the folder, the message was retried and the error was not encountered again, so Maximo deleted the error file.

Example

An error occurs in an inbound receipt message due to an invalid general ledger (GL) account. After the error occurs, an online user subsequently enters that GL account in Maximo. Maximo successfully reprocesses the message and the data is saved successfully in Maximo.

Example

An outbound transaction encounters a communication error. When the communication problem is resolved, the message is sent to the external system and the error file is deleted.
One or More XML Files Exist

When an error occurs in a sequential queue (inbound or outbound), processing of the queue stops until the error is resolved. In these cases, one XML file will exist in the error folder for the queue.

When an error occurs in a continuous queue, processing of the queue continues and additional errors may occur before the initial error is resolved. In these cases, multiple XML files will exist in the error folder for the queue.

No ER and IR Elements in the Error File

If the corresponding error file contains no ER or IR element, the Maximo integration framework encountered a critical processing error.

The following example is an error XML file that the Maximo integration framework creates when it encounters a critical processing error:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ERROR>
    <ERRORMESSAGE>A major exception has occurred. Error on line 14: An invalid XML character (Unicode: 0x1b) was found in the element content of the document.</ERRORMESSAGE>
</ERROR>
```

Due to the critical nature of the error, the Maximo integration framework cannot retry the XML file. To correct the critical error, remove the invalid data from the inbound queue. You then must place the error message in the delete folder. For more information about the delete folder, see “Error Management Folders,” on page 7-7.

MEA error processing uses a view queue utility to help you identify the application data content that caused the critical error. The utility creates a “view queue file” and places the file in the view folder under the integration global directory. The “view queue file” is identified with the same file name as the error XML file and its data content includes the message initially received in the inbound queue.
The following example is a “view queue file,” which the MEA view queue utility created due to a critical processing error:

```xml
<?xml version="1.0" encoding="UTF-8"?
<MXCRAFTInterface xmlns="http://www.mro.com/mx/integration"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
language="EN">
  <Header operation="Notify" event="1">
    <SenderID type="MAXIMO" majorversion="6" minorversion="1"
build="127" ddbuild="V600-467">MX</SenderID>
    <CreationDateTime>2006-10-17T13:50:59-04:00</CreationDateTime>
    <RecipientID>EXTSYS1</RecipientID>
    <MessageID>11611074597855980</MessageID>
  </Header>
  <Content>
    <MXCRAFT>
      <CRAFT action="Replace">
        <CRAFT>ISHA02</CRAFT>
        <ORGID>GLENA</ORGID>
        <DESCRIPTION changed="1">33333333</DESCRIPTION>
        <CRAFTID>46</CRAFTID>
        <STANDARDRATE>0.0</STANDARDRATE>
        <SENDERSYSID>MX</SENDERSYSID>
        <SOURCESYSID />
        <OWNERSYSID />
        <EXTERNALREFID />
        <TRANS_LANGCODE>EN</TRANS_LANGCODE>
      </CRAFT>
    </MXCRAFT>
  </Content>
</MXCRAFTInterface>
```

The invalid character on line 14 of the “view queue file” caused the critical processing error. The Maximo integration framework does not process the message until you remove the invalid data from XML file.

### Updating Message Data

The system administrator has the ability to change the content of inbound or outbound messages. Changes to inbound messages are more common.

To correct the data in an error file and make it available for reprocessing, perform the following actions:

1. Copy the file from the error folder to another location; for example, c:\temp.
2. Edit the ER data as necessary. The IR data is for informational purposes only.
3 Move the modified file to the retry folder for the queue and external system in which the error occurred. Maximo automatically tries to process this file during its next cycle.

If successful, it deletes the transaction from the queue, the error folder, and the retry folder. If unsuccessful, it performs the following actions:

a. Deletes the message from the retry folder
b. Saves the file in the error folder with extension .bak
c. Overwrites the original file in the error folder
d. Retains the original transaction in the queue

4 Repeat the previous steps until the message is successfully processed.

NOTE All backups of the error file (extension .bak) remain in the folder for audit purposes. You must manually delete them.

When the Maximo integration framework identifies a critical error, ER and IR sections in the corresponding error file are not present. To correct the critical error, remove the invalid data from the inbound queue. You then must place the error message in the delete folder. For more information about critical errors, see “Critical Errors,” on page 7-9.

**Deleting Errors**

You can delete a message from a queue, if necessary.

**CAUTION** After you delete a message from the error folder, Maximo cannot reprocess it.

To delete a message from a queue, perform the following action:

- Copy the error file from the error folder to the deleted folder for the queue and external system.

In its next cycle, the error management processing deletes the message from queue. It saves a copy of the message in the appropriate subfolder of the deleted history folder, for logging purposes only, and deletes the XML file from the error and delete folders.
Advanced Topics

Chapter 8: Integration Gateway
Chapter 9: Router
Chapter 10: Advanced Interface Table Polling
Chapter 11: JMS Queue Configuration
Chapter 12: Security
Chapter 13: Cluster Configuration
Chapter 14: Customization with Processing Rules
Chapter 15: Customization with User Exits
Chapter 16: Adding and Modifying Integration Components
Chapter 17: Using Integration Queries
Chapter 18: Maximo Web Services
Chapter 19: Multiple Language Support
The integration gateway provides the framework for receiving XML transactions from an external system and writing them to the inbound queues. The gateway processes transactions received via HTTP/HTTPS, Web services, and Enterprise JavaBeans (EJBs).

The integration gateway is not involved in processing transactions received via interface tables and flat files (using the Data Import feature), and data synchronization and query type interfaces exposed through interface Web services.

This chapter describes the receipt of XML transactions via the integration gateway and the function of the interpretation layer of the gateway. It is directed to developers and support personnel and contains the following sections:

- Gateway Framework
- Interpretation Layer
- Writing Your Own Interpreter
- Bypassing Interpretation
- Sending Transactions via EJB
- Sending Transactions via HTTP/HTTPS
- Sending Transactions via the Gateway Web Service

For information about inbound interface tables, flat files, and interface Web services, see the following documentation:

- Chapter 5, "Interface Tables," on page 5-1
- "Configuring the Data Import Feature," on page 6-38
- Chapter 18, "Maximo Web Services," on page 18-1
The gateway framework provides entry points through which external systems can deliver messages to Maximo. The gateway consists of an Enterprise JavaBean, which provides the following types of support:

- Remote Method Invocation (RMI)
- Enterprise JavaBeans (EJB)
- Java Naming and Directory Interface (JNDI)

The EJB can also be exposed as a Remote Procedure Call (RPC) based Web service, thereby providing SOAP support.

**Components in the Gateway Framework**

There are two ways to invoke the gateway EJB. One involves interpretation and the other does not.

**Interpretation Layer**

Before the gateway writes a transaction to an inbound queue, it must obtain and validate the names of the interface and the external system in which the transaction originated. These properties are included in the header of the JMS message that the gateway writes to the queue. Interpretation is the process of parsing an XML document to obtain the information that Maximo needs to process the document.

If these values are in a consistent location in the XML for all interfaces within an adapter, or if you can provide a fixed value for either property, you define the location of these properties on the Gateway Properties tab. Otherwise, you must write a custom interpreter class that will identify the sender and interface names. If you write an interpreter class, you define the interpreter class, but no properties, on the Gateway Properties tab. To include additional data in the message header, define that data as one or more properties in the Add/Modify Adapters dialog box.

The Maximo adapter uses the following two gateway properties:

- INTERFACE (the interface name), derived from the root element (top tag) in the XML message
- SENDER (the external system name), derived the SenderID tag

**CAUTION** Do not delete these predefined gateway properties from the Maximo adapter.
When you define the gateway properties, Maximo interprets their values as follows:

- If you select the XML Tag? check box and leave the Value field null, the gateway uses the name of the root element in the XML message as the value for the corresponding property.

- If you select the XML Tag? check box and enter a tag name in the Value field, the gateway uses the value for that tag as the value for the corresponding property. If the tag appears multiple times in the XML message, the adapter uses the value of the first occurrence of the tag.

- If you clear the XML Tag? check box and enter a data value in the Value field, the gateway uses that data as the value for the corresponding property.

Example

If you do not define a value for the INTERFACE property, and the top tag of an inbound XML message is <MxItemInterface>, the gateway uses MXItemInterface as the value for INTERFACE property.

Writing Your Own Interpreter

Use the base interpreter, interpreter.java, when the names of the interface and sender are derived from XML tags in the transaction or have a fixed value. If the adapter’s interfaces communicate with a single external system, for example, you can prespecify the external system identifier in the SENDER property.

If the interface and sender names must be derived in another way, the adapter must implement its own interpreter. You can create this class by extending the base interpreter class and overriding the following method:

```java
public Map interpretMessage(byte[] extData)
```

In this method, extData is the input XML to be interpreted.

The properties are protected variables, called props, in the base class. The return from this method is a map, which contains the property name as the key and the property value as the value.
Bypassing Interpretation

An external system can provide the interface and sender names by invoking the following method of the EJB

```java
processExternalDataWithParams(byte[] extData, String ifaceName, String sender)
```

The EJB bypasses the interpreter layer and writes data directly to the appropriate inbound queue. Using this approach results in better performance than using the interpreter class.

For more information, see the discussion of this method in the following section, “Sending Transactions via EJB.”
Sending Transactions via EJB

EJB invocation can be performed by J2EE™ clients written as per the Sun Microsystems®, Inc. J2EE client specification. Invocation can be done with or without the interface and sender names.

To invoke the EJB without the interface and sender names, call the following method:

```java
public String processExternalData(byte[] extData, String ifaceType)
```

In this method, ifaceType is the name of the adapter (for example, Maximo).

This method uses the XML data and the adapter as parameters and determines the sender and interface details by parsing the XML data.

To invoke the EJB with the interface and sender names, call the following method:

```java
public String processExternalDataWithParams(byte[] extData, String ifaceName, String sender)
```

A client can use the JNDI name of the gateway EJB (syncMaximoDataService) to look up the EJB reference and invoke either method. The client also needs the following:

- access to the Home and Remote interface class files
- access to the J2EE jar files for the server
- the URL of the server hosting the EJB
- the context factory class name

The client code must instantiate the default InitialContext object. The context derives the provider URL and the context factory from the environment.

**NOTE**

To keep the client code server- and URL-neutral, set the provider URL and context factories as –D parameters in the .bat or .sh script for the J2EE client.

To see an example of a J2EE client .bat/.sh file, go to the following folder:

`tools\maximo\j2eeclient` folder

**Security**

For information about security for EJB transactions, see Chapter 12, "Security," on page 12-1.

**Response**

If the invocation is successful, the EJB returns null.

If the invocation is unsuccessful, the EJB raises an exception detailing the error.
Sending Transactions via HTTP/HTTPS

A servlet provides the ability for external systems to post transactions to the integration gateway. Use the following URL to post the transactions:

http://<domain>:<port>/meaweb/measervlet/<adapter>

where <domain> is the machine or domain name and <adapter> is the name of the adapter (for example, Maximo).

The maximum size of a message that can be sent to Maximo via HTTP POST is controlled by the IntegrationPostSize parameter in the web.xml file of the meaweb application. The default value is 5MB, and you can change this value as needed. For security reasons, a message is rejected if it exceeds the size specified in this parameter. A very large message can cause an OutOfMemoryError when an XML parser tries to parse it.

If the XML transaction uses character encoding other than UTF-8, the message must contain the encoding attribute, as in the following example:

<?xml version="1.0" encoding="ISO-8859-2"?>

If the message does not contain this attribute, Maximo assumes that the XML message uses UTF-8 encoding. Since the integration servlet does not look at the charset value in the HTTP Content-Type header, simply providing the charset value is not sufficient.

The XML data should be contained directly in the body of the HTTP Post message. You can set the HTTP Content-Type header to text/xml.

Security

For information about security for HTTP transactions, see Chapter 12, "Security," on page 12-1.

Response

If the transaction is successful, the servlet returns a status code of 200 and the return message, if any, as part of the response body. The response content type is text/plain and is encoded in UTF-8.

If the transaction is unsuccessful, the servlet returns a status code of 500 and the error message it encountered. The response content type is text/plain and is encoded in UTF-8.
Sending Transactions via the Gateway Web Service

The gateway Web service, which is deployed out of the box, can be used as an alternative to posting data into Maximo. The two methods described in "Sending Transactions via EJB," on page 8-5, are available for invocation via SOAP-RPC client. The client code can be written in any language and can use any SOAP-RPC tool which works with that language. For example, in Java, any JAX-RPC implementation can be used to invoke this Web service.

To invoke this Web service, the client must use the following values.

**Gateway Web Service Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>wsdl URL</td>
<td>http://&lt;host&gt;:@&lt;port&gt;/meaweb/services/syncMaximoDataService?wsdl</td>
</tr>
<tr>
<td>service name</td>
<td>MEAGatewayService</td>
</tr>
<tr>
<td>portname</td>
<td>syncMaximoDataService</td>
</tr>
<tr>
<td>target namespace</td>
<td>http://&lt;host&gt;:@&lt;port&gt;/meaweb/services/syncMaximoDataService</td>
</tr>
</tbody>
</table>

**Security**

For information about security for gateway Web service transactions, see Chapter 12, "Security," on page 12-1.

**Response**

If the invocation is successful, the response is a SOAP message with an empty SOAP body.

If the invocation is unsuccessful, the response is a SOAP fault message detailing the error.
Sending Transactions via the Gateway Web Service
The router is responsible for routing outbound messages from the outbound JMS queue to an end point associated with an external system.

The outbound queue cron task process invokes the router and passes the message body and the header properties to it. The router uses the header properties to determine the external system and the handler and end point properties for that system. The router executes the handler, which sends the data to the external system specified by the end point with which the handler is associated.

This chapter explains how to configure the predefined handlers Maximo offers, or create new ones. It contains the following sections:

- End Points
- Handlers
- Writing Custom Handlers
End Points

An end point represents an application component to which Maximo delivers outbound transactions. Maximo provides the following predefined end points.

**Predefined End Points**

<table>
<thead>
<tr>
<th>End Point</th>
<th>Handler</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MXFLATFILE</td>
<td>FLATFILE</td>
<td>Writes flat files to a prespecified directory location.</td>
</tr>
<tr>
<td>MXIFACETABLE</td>
<td>IFACETABLE</td>
<td>Writes outbound transactions to local interface tables.</td>
</tr>
<tr>
<td>MXXMXMLFILE</td>
<td>XMLFILE</td>
<td>Writes XML files to a prespecified directory location.</td>
</tr>
</tbody>
</table>

The configuration of an end point involves the definition of the end point, the association of a handler to the end point, and the setting of the handler properties, if any, for that end point.

From an architectural and implementation perspective, an end point is an instance of a handler with specific properties that the handler needs in order to connect and write outbound data.

You can use the predefined handlers or create new ones that enable physical entities, such as FTP servers, for which predefined handlers do not exist. You must define the handler before you define the end point. You configure end points and handlers via the Select Action menu in the External Systems application.

To define an end point for an existing handler, complete the following activities:

1. Use the Add/Modify End Point action to define a new end point or associate the handler with an existing end point.
2. Set the handler properties that apply to the corresponding end point.
3. Associate the end point with one or more external systems.

**Note**

A property is associated with a handler, but the property value is specific to the end point implementing the handler. For example, three external systems (end points) that use the EJB handler can have different values in the property fields. Therefore, you enter property values at the time you associate the handler with an end point.
Handlers

This section describes the predefined handlers and their required and optional properties.

Maximo provides the following predefined handler types, which you can access through the Select Action menu in the External Systems application.

- EJB
- FLATFILE
- HTTP
- IFACETABLE
- JMS
- WEBSERVICE
- XMLFILE
**EJB Handler**

The EJB handler is a Java component that delivers Maximo data to an Enterprise JavaBean (EJB) executing in the local application server or a remote application server.

**NOTE**

If the EJB is in a remote application server, the remote and home interface classes of the EJB must be available in the local Maximo application server. For more information, refer to the documentation for your application server.

The EJB handler has the following properties:

**CONTEXTFACTORY Property**

This required property specifies a J2EE context. Refer to the documentation for your application server for the name of the default context factory.

**CONTEXTFACTORY Property Values**

<table>
<thead>
<tr>
<th>Server</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM WebSphere</td>
<td>com.ibm.websphere.naming.WsnInitialContextFactory</td>
</tr>
<tr>
<td>BEA WebLogic</td>
<td>weblogic.jndi.WLInitialContextFactory</td>
</tr>
</tbody>
</table>

**EJBEXIT Property**

This optional property is used for customization. It specifies the fully qualified name of a custom Java class that implements the EJBExit interface.

If you do not specify a value for this property, Maximo executes the default exit, `DefaultEJBExit`, which attempts to resolve the EJB's method signature and parameters.

If your EJB has its own method signature and parameters, create a Java class that contains your version of the EJBExit interface and implementations of the following four methods.

```java
public Class[] getClassParams()
```

The `getClassParams()` method returns the method signature in the form of an array of Java classes.

```java
public Object[] getobjectParams(byte[] data, String interfaceName, String destinationName) throws MXException
```

The `getObjectParams()` method returns the parameters of the EJB business method in the form of an array of Java objects.

```java
public void responseOk(Object response) throws MXException
```

The `responseOk()` method is called after a successful EJB invocation.

```java
public void responseError(Exception e) throws MXException
```

The `responseError()` method is called with the originating exception as a parameter if an error is encountered during EJB invocation.
Handlers

Examples

If your EJB has a business method with a byte array and a string, your implementation of getClassParams() will look like the following:

```java
Class[] classParams = {byte[].class, String.class};
return classParams;
```

If your EJB has a business method with a byte array and a string, your implementation of getObjectParams() would look like the following:

```java
byte[] data;
String ifaceType;

Object[] objParams = {data, ifaceType};
return objParams;
```

To identify the location of the package structure for the EJBExit class file, complete one of the following actions:

- Place the class in the maximo java package structure applications/maximo/businessobjects/classes folder.
- Modify the mboweb\webmodule\META-INF\MANIFEST.MF classpath to include the package structure.

**NOTE** Rebuild the Maximo EAR file and include this file in it. For more information about the EJBEXIT property, see Chapter 15, "Customization with User Exits," on page 15-1.

**JNDINAME Property**

This required property specifies the name by which the EJB is registered in the application server's Java Naming and Directory Interface (JNDI) tree.

**JNDINAME Property Values**

<table>
<thead>
<tr>
<th>Server</th>
<th>Value</th>
</tr>
</thead>
</table>
| IBM WebSphere | ibm-ejb-jar-bnd.xml <ejbBindings xmi:id="Session_syncMaximoDataEJB_Bnd"
jndiName="syncMaximoDataService">
<enterpriseBean xmi:type="com.ibm.etools.ejb:Session"
href="META-INF/ejb-jar.xml#Session_syncMaximoDataEJB"/>
</ejbBindings> |
| BEA WebLogic | weblogic-ejb-jar.xml <weblogic-enterprise-bean>
<ejb-name>syncMaximoDataEJB</ejb-name>
<jndi-name>syncMaximoDataService</jndi-name>
</weblogic-enterprise-bean> |

**METHODNAME Property**

This required property specifies the public business method exposed by the EJB that is invoked through this handler.
Handlers

**PROVIDERURL Property**

This required property specifies the URL that provides access to the EJB component.

**PROVIDERURL Property Values**

<table>
<thead>
<tr>
<th>Server</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM WebSphere</td>
<td>corbaloc:iiop:&lt;hostname&gt;:&lt;iiopport&gt;</td>
</tr>
<tr>
<td>BEA WebLogic</td>
<td>t3://&lt;hostname&gt;:&lt;port&gt;</td>
</tr>
</tbody>
</table>

**USERNAME and PASSWORD Properties**

If access to the EJB requires a user name and password, these properties specify those values. The user name must match the value configured during the definition of Users and Groups in the application server.

For more information about EJB security, refer to the documentation for your application server.

**FLATFILE Handler**

The FLATFILE handler converts outbound data from the queue into a flat file and writes it to a directory whose location is configurable. Flat files contain ASCII data in the form of rows and columns. Each line of text constitutes one row, and a separator separates each column in the row.

The FLATFILE handler encodes outbound flat files in the standard UTF-8 format; and the Data Import mechanism assumes that inbound flat files are encoded in UTF-8 format.

**NOTE**

The FLATFILE handler can be used only with external systems that use an internal type adapter.

**Flat File Naming Convention**

File names have the following format:

`externalsystemname_interfacename_uniqueidentifier.DAT`

where

- `externalsystemname` is the identifier of the Maximo system (the value of MAXVARS.MXSYSID)
- `interfacename` is the name of the interface
- `uniqueidentifier` is a number based on current system time.

**Example**

The file name `MX_MXASSETInterface_10971102668641498.dat` indicates the following:

- The file was generated from a Maximo system with MXSYSID = "MX"
- The file contains the MXASSETInterface interface
The first two lines of the file contain header information. The first line has the following format:

```
<externalsystemname><separator><interface name><separator>
[action]
```

The second line of the file contains the names of the columns, separated by the separator character. The column names are the same as those in the corresponding interface table.

**Flat File Formatting**

If the data within the flat file contains the flat file delimiter character, the data will be wrapped in the text qualifier, which is " (double quote). If the data contains a double quote, the handler escapes the double quote.

**NOTE** You cannot use a double quote as the delimiter character.

**Example**

The following data uses a comma (,) as a delimiter. The INVOICEDESC value (Rotating Custom Item, No 71) contains a comma. When the flat file is written, the INVOICEDESC value will be wrapped in double quotes.

```
EXTSYS1,MXINVOICEInterface,Add
INVOICENUM, INVOICEDESC, PONUM, VENDOR, CONTACT, PAYMENTTERMS
1071,"Rotating Custom Item, No 71",1000,A0001,/
```

**Example**

The following data uses a comma (,) as a delimiter. The INVOICEDESC value (Rotating "Custom" Item No 71) contains double quotes. When the flat file is written, the double quote in INVOICEDESC data is escaped with another double quote and the entire string is wrapped in double quotes.

```
EXTSYS1,MXINVOICEInterface,Add
INVOICENUM, INVOICEDESC, PONUM, VENDOR, CONTACT, PAYMENTTERMS
1071,"Rotating "Custom" Item No 71",1000,A0001,/
```

**Example**

The following data uses a comma (,) as a delimiter. The INVOICEDESC data (Rotating "Custom" Item, No. 71) contains the delimiter character and double quotes. When the flat file is written, the INVOICEDESC value will appear as in the following example.

```
EXTSYS1,MXINVOICEInterface,Add
INVOICENUM, INVOICEDESC, PONUM, VENDOR, CONTACT, PAYMENTTERMS
1071,"Rotating "Custom" Item, No. 71",1000,A0001,/
```

**Flat File Properties**

This handler has the following properties:

**FLATFILEDIR Property**

This required property specifies the location of the flat file. The location must already exist, on the local machine where the Maximo application server is executing or on a shared network drive.
Handlers

FLATFILESEP Property
This required property specifies the character that separates the columns in each row.

HTTP Handler

The HTTP handler is a Java component that delivers data as an XML document to a URL using HTTP or HTTPS protocols, and evaluates the response code received from the external system.

This handler has the following properties:

HTTPEXIT Property
This optional property is used for customization. It specifies the fully qualified name of a Java class that interprets the HTTP response. An external system might require additional code to interpret the HTTP response, and this exit lets users implement the necessary code.

The Java class must be available in the classpath specified for the application server or the Maximo EAR file.

**HTTPEXIT Property Values**

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java class</td>
<td>PSFTHTTPExit.java</td>
</tr>
<tr>
<td>Package</td>
<td>psdi.iface.router</td>
</tr>
<tr>
<td>HTTPEXIT Property</td>
<td>psdi.iface.router.PSFTHTTPExit</td>
</tr>
</tbody>
</table>

Maximo provides a default implementation of the HTTP exit. The Java class is DefaultHTTPExit, which is in the psdi.iface.router package and implements the psdi.iface.router.HTTPExit interface. The key method within this class is the processResponseData() method, which has the following signature:

```java
public void processResponseData(int responseCode, String responseMsg, byte[] msgBodyData) throws MXException
```

The default implementation compares the response code from the external system to a range of valid codes (values 200 through 299). If the response code falls outside that range, Maximo assumes the transaction was not delivered to the external system. An exception is raised and the message remains in the queue.

If you require additional processing for a specific implementation, extend the default implementation and override the processResponseData() method. As an alternative, you can implement the psdi.iface.router.HTTPExit interface yourself. The overriding method must raise an exception if the response received from the external system does not pass the validation in this class.

If you leave this property empty, Maximo executes the default implementation of HTTPExit.

For more information about the HTTPEXIT property, see Chapter 15, "Customization with User Exits," on page 15-1.
URL Property
This optional property specifies a valid URL to which XML data can be posted. It is expected that a response will be generated whenever an HTTP POST is performed upon this URL.

USERNAME and PASSWORD Properties
If the URL requests basic authorization, these properties specify the required values. Maximo encodes both values and passes them to the URL.

For more information about basic authorization for HTTP, refer to the documentation for your HTTP server.

IFACETABLE Handler

The IFACETABLE handler writes data from an outbound queue to an interface table in a local or remote database. There are no Java exits for this handler.

NOTE
The IFACETABLE handler can be used only with external systems that use an internal type adapter.

This handler has the following properties:

ISREMOTE Property
This required property specifies if interface tables are available in the local Maximo database in the Maximo schema or in a different schema. Its value can be 0 (zero) or 1.

A value of 0 (false) indicates the interface tables are available in the local Maximo database in the Maximo schema. You do not have to enter any other handler properties. For the predefined MAXIFACETABLE handler, the value of this property is 0.

A value of 1 (true) indicates the interface tables are in a remote database. If so, you must specify values for all the handler properties.

DRIVER Property
This property specifies the JDBC driver to be used to connect to a remote database containing the interface tables. This property applies only when the value of ISREMOTE is 1.

URL Property
This property specifies the JDBC URL that indicates the location, port number, and database name. This property applies only when the value of ISREMOTE is 1.

Example
jdbc:db2://mea6:5000/MERLIN

USERNAME and PASSWORD Properties
If access to the database instance requires a user name and password, these properties specify those values. These properties apply only when the value of ISREMOTE is 1.
JMS Handler

The JMS handler is a Java component that delivers XML data into a messaging system that has been enabled through Java Messaging Service (JMS).

Depending upon the messaging model you implement, messages are placed in a virtual channel called a queue or topic. In the point-to-point messaging model, messages are generated by a sender and placed in a queue. Only one receiver can obtain the message from the queue. In the publish-subscribe messaging model, messages are generated by a publisher and placed in a topic. Multiple subscribers can retrieve the message from the topic.

The following diagram illustrates these two messaging models.

Publish-Subscribe and Point-to-Point Messaging Models

The messaging system discussed in this section represents a queue or topic available in the local application server, a remote application server, or a remote dedicated queuing system such as IBM MQ Series. To use this handler, all such messaging systems must be enabled through JMS.

For more information, refer to the documentation for the application server.

NOTE The messaging system discussed in this section is distinct from the standard internal queues used by Maximo. The standard internal queues reside in the local application server where Maximo is executing.
This handler has the following properties:

**CONFACTORYJNDINAME Property**

This required property specifies a Java object that is used to manufacture connections to a Java Message Server provider. Before directly connecting to a queue or topic, Maximo must first obtain a reference to a queue or topic connection factory.

Application servers usually provide a default connection factory. However, implementing your own connection factory lets you tune the resource attributes and properties to suit your implementation. In this case, use the following value for this property:

```
.jms/mro/int/qcf/intqcf
```

**DESTINATIONTYPE Property**

This required property specifies the JMS entity (queue or topic) to which the message will be delivered.

**DESTINATIONTYPE Property Values**

<table>
<thead>
<tr>
<th>Destination</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Topic</td>
<td>javax.jms.Topic</td>
</tr>
<tr>
<td>Queue</td>
<td>javax.jms.Queue</td>
</tr>
</tbody>
</table>

**DESTJNDINAME Property**

This required property specifies the name by which the JMS queue/topic is registered in the application server’s Java Naming and Directory Interface (JNDI) tree.

**CONTEXTFACTORY Property**

This required property specifies a J2EE context. Refer to the documentation for your application server for the name of the default context factory.

**CONTEXTFACTORY Property Values**

<table>
<thead>
<tr>
<th>Server</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM WebSphere</td>
<td>com.ibm.websphere.naming.WsnInitialContextFactory</td>
</tr>
<tr>
<td>BEA WebLogic</td>
<td>weblogic.jndi.WLInitialContextFactory</td>
</tr>
</tbody>
</table>

**ISCOMPRESS Property**

This required property specifies whether the message will be compressed before being placed into a queue or topic. Compression is an optimization technique that delivers smaller messages to a queue or topic.

**ISCOMPRESS Property Values**

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do not compress data</td>
<td>0</td>
</tr>
<tr>
<td>Compress data</td>
<td>1</td>
</tr>
</tbody>
</table>

**NOTE**

Compressed messages must be decompressed after receipt. Decompress the messages by creating the appropriate JMS receiver or subscriber component and placing Java decompression logic within the receiver or subscriber. Use the standard Java Inflater() class that is part of the java.util.zip package. The Maximo-provided compression uses the standard Java Deflator() class.
Handlers

**JMSEXIT Property**

This optional property is used for customization. It specifies the fully qualified name of a Java class that extends the JMSExit interface. The Java class must implement the getMessageProperties() method that is defined in the JMSExit interface.

This option lets you change or add header information in the JMS message. If this property does not contain a value, the header attributes for the message are not changed when the message is delivered to the external queue or topic.

For more information about the JMSEXIT property, see Chapter 15, "Customization with User Exits," on page 15-1.

**PROVIDERURL Property**

This required property specifies a local or remote URL where the JMS component can be accessed. It can be local or remote.

**PROVIDERURL Property Values**

<table>
<thead>
<tr>
<th>Server</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM WebSphere</td>
<td>corbaloc:iiop:&lt;hostname&gt;:&lt;iiopport&gt;</td>
</tr>
<tr>
<td>BEA WebLogic</td>
<td>t3://&lt;hostname&gt;:&lt;port&gt;</td>
</tr>
</tbody>
</table>

**USERNAME and PASSWORD Properties**

If the application server controls access to the JMS resource, these properties specify the user name and password needed to connect to the resource. The user name must match the value configured during the definition of Users and Groups in the application server.

For more information, refer to the documentation for your application server.

**WEBSERVICE Handler**

The WEBSERVICE handler is a Java component that invokes a specified Web service with Maximo data as a SOAP request parameter. This handler is a Dynamic Invocation Interface (DII) using the JAX-RPC API. It supports a document-literal type Web service.

This handler has the following properties:

**ENDPOINTURL Property**

This required property specifies a valid Web service URL on which to invoke the document-literal style Web service. You can use the WSEXIT class to override the end point URL just before the Web service is invoked.

**ONEDAYWS Property**

This optional property specifies whether the Web service is one-way. Valid values are 0 (false) and 1 (true). If you do not provide a value, Maximo uses the default value 0 (false). You can use the WSEXIT class to override the value specified in the user interface just before the Web service is invoked.

**SERVICENAME Property**

This optional property specifies the name of the Web service deployed in the URL. If you do not provide a value, Maximo uses the interface name in the XML. You can use the WSEXIT class to override the service name just before the Web service is invoked.
**SOAPACTION Property**
This optional property specifies the value of SOAPAction HTTP header to be used while invoking the Web service. If you do not provide a value, Maximo uses the default value "" (empty string). You can use the WSEXIT class to override the value specified in the user interface just before the Web service is invoked.

**TARGETNAMESPACE Property**
This required property specifies the namespace of the Web service.

**USERNAME and PASSWORD Properties**
If the specified Web service is secured (that is, if HTTP basic authentication is enabled), specify a user name and password. Maximo encodes the password.

**WSEXIT Property**
This optional property is used for customization. It specifies the fully qualified name of a custom Java class that implements the psdi.iface.router.WSExit interface. This interface defines the following methods:

```java
public String getServiceName(Map metaData, String endpointURL, String serviceName, String interfaceName, String targetNameSpace) throws MXException
```

The `getServiceName()` method returns the service name of the Web service to be invoked.

```java
public String getEndpointURL(Map metaData, String endpointURL, String serviceName, String interfaceName, String targetNameSpace) throws MXException
```

The `getEndpointURL()` method returns the end point URL of the Web service to be invoked.

```java
public void responseOk(org.w3c.dom.Document response) throws MXException
```

The `responseOk()` method is called after a successful invocation of the external Web service.

```java
public void responseError(Exception e) throws MXException
```

The `responseError()` method is called with the originating exception as a parameter if an error was encountered during the Web service invocation.

```java
public boolean getOneWayWsInfo(Map metaData, String endpointURL, String serviceName, String interfaceName, String targetNameSpace, boolean oneWayWs) throws MXException
```

The `getOneWayWsInfo()` method returns a Boolean value that specifies whether the Web service being invoked is one-way.

```java
public String getSoapAction(Map metaData, String endpointURL, String serviceName, String interfaceName, String targetNameSpace, String soapAction) throws MXException
```

The `getSoapAction()` method returns the SOAPAction HTTP header to be used while invoking the Web service.

These methods can be used to override the default Service name and endpoint URL, and to handle the response and error after the Web service invocation.
Handlers

There is a default implementation of the WSExit interface, psdi.iface.router.DefaultWSExit. This class overrides the getEndpointURL() method to concatenate the servicename at the end of endpoint URL, to form the new endpoint URL.

**NOTE**  Use this class name in the WSEXIT property if you communicate with a remote Maximo system via Web services.

**XMLFILE Handler**

The XMLFILE handler is a Java component that converts data in the outbound queue into an XML file format, then delivers it to the xmlfiles directory within the global directory. You define the global directory in the Integration Administration Setup dialog box in the External Systems application.

This handler does not include any properties.

File names have the following format:

```
externalsystemname_interfacename_uniqueidentifier.xml
```

where

- `externalsystemname` is the identifier of the Maximo system (the value of MAXVARS.MXSYSID).
- `interfacename` is the name of the interface.
- `uniqueidentifier` is a number based on current system time.

**Example**

The file name MX_MXASSETInterface_10971102668641398.xml indicates the following:

- The file was generated from a Maximo system where MXSYSID = “MX”.
- The file contains the MXASSETInterface interface.
Writing Custom Handlers

To write a custom handler, you must implement the RouterHandler interface. That interface has two methods:

```
getParameter()
```

This method returns a list of properties needed for the handler to send data to the desired end point.

- The `RouterPropsInfo` object represents a property. This method can return a list of `RouterPropsInfo` objects.

- The `isCrypto` attribute in the `RouterPropsInfo` objects indicates whether to encrypt the property value while storing. For password properties, the value of this attribute generally is `True`.

```
sendData(Map metaData, byte[] data, Map destinationMap)
```

This method sends data to the specified end point.

- `metaData` provides information about the external system and the interface

- `data` is the XML data

- `destinationMap` specifies the end point

You must define the handler and associate it with an end point, in the External Systems application. During this process Maximo introspects the handler class and displays the properties for which you must specify values.

**Example**

For an example of a handler, see `FTPHandler.java` in the `psdi.iface.samples` directory.
Maximo provides the option of using interface tables to exchange data with external systems. This chapter discusses how to perform advanced configuration of the interface table polling process to improve its performance reading data from interface tables. This chapter is addressed to the system administrator managing Maximo Integration activities.

NOTE Taking steps to improve interface table polling performances is necessary only if you send inbound transactions through the continuous JMS queue and do not require transactions to be maintained in FIFO sequence.

As a prerequisite, read Chapter 5, "Interface Tables," on page 5-1.

This chapter contains the following sections:

- Cron Tasks
- Selectors
- Queue Tables

Unless otherwise indicated, the configuration activities in this section are performed in the Cron Task Setup application in the Maximo Configuration module.
Cron Tasks

The interface table polling process uses a single default cron task called IFACETABLECONSUMER. This cron task reads all transactions from all interface tables for all external systems that write to the tables.

For improved single-server and multi-server performance, you can configure multithreaded interface table polling by defining multiple instances of this task with different property values. Multithreaded polling is particularly useful when running in a clustered configuration, as different threads can run on different servers, thereby balancing the load.

Example

To designate an instance of the cron task to run on a specific Maximo application server, perform the following actions. Assume the name of the cron task instances is instance1:

- In the Cron Task Setup application, set the TARGETENABLED property to 1
- In the Maximo application server1 setup, set –DIFACETBCONSUMER.instance1=1

For more information about cron tasks, refer to the IBM Maximo System Administrator’s Guide and the online help for the Cron Task Setup application.

CAUTION When implementing multiple cron tasks, you also must implement mutually exclusive selectors to avoid processing a transaction more than once.

Selectors

A selector lets you add a where clause to a cron task. If you define multiple instances of the cron task, you must define selectors, so each instance reads mutually exclusive interface table rows. For example, if Maximo exchanges data with two external systems, the first thread might poll one system and the second thread might poll the second system.

You define selectors by assigning values to the EXTSYSNAME (external system) and INTERFACENAME (interface) parameters in the Cron Task Setup application. You can add an IN clause to a selector by entering a pipe delimited set of values.

Examples

To direct a cron task to select only purchase order records for system EXTSYS1 from the queue tables, perform the following actions:

- Set INTERFACENAME=MXPOInterface
- Set EXTSYSNAME=EXTSYS1
Queue Tables

To list multiple interface names in the INTERFACE NAME property, perform the following action:

- Set INTERFACE NAME=MXPOInterface | MXPRInterface | …

**CAUTION** When you define multiple instances of the cron task, ensure that:

- The selectors are mutually exclusive, so transactions are not processed multiple times
- The selectors retrieve all the interfaces that you use, so no transactions are left unprocessed

Queue Tables

Another approach to improving interface table processing is the implementation of multiple queue tables. The MXIN_INTER_TRANS queue table, which Maximo creates when it creates the interface tables, is the default queue table. The IFACETABLECONSUMER cron task reads the MXIN_INTER_TRANS table and uses it as a driver to find and process the corresponding data in the interface tables.

You can improve performance by setting up multiple queue tables. For example, you can write each interface to a separate queue table and define separate cron tasks to process the queue tables independently of one another; or you can set up separate queue tables for each external system and, within each queue table, define selectors for each interface. Depending upon the complexity of your integration, you can use multiple queue tables instead of multiple selectors or in conjunction with them.

To set up multiple queue tables, you must manually create the queue tables in the same database as the interface tables, and include all the columns that are in the MXIN_INTER_TRANS queue table. You must design the external system to write to the appropriate queue table(s).

**CAUTION** Ensure that the external system does not insert an interface table transaction into more than one queue table, or the transaction will be processed multiple times.
Maximo writes transactions to JMS queues after receiving inbound messages from an external system via the integration gateway, interface tables or file loading, and before sending outbound messages to external systems. Outbound transactions remain in the outbound queue until they are successfully sent to an external system or deliberately deleted from the queue via error management processing. Inbound transactions remain in an inbound queue until they are successfully processed in Maximo or deliberately deleted from the queue via error management processing.

The use of JMS queues allows for scalability within a single Maximo application server or across a cluster of application servers. There are three default queues—one outbound sequential queue and two inbound queues, one sequential and one continuous.

This chapter discusses the configuration of JMS queues. It is addressed to the system administrator, and contains the following sections:

- Sequential Queues
- Continuous Queue
- Selectors
- Queue Utilities
- Queue Creation
- Queue Security

For information about error management, see Chapter 7, "Error Management," on page 7-1.

For information about queues in a clustered environment, see Chapter 13, "Cluster Configuration," on page 13-1.
Sequential Queues

The sequential queue is a JMS queue with a Maximo cron task as a consumer. Transactions in sequential queues are processed on a strict FIFO (first in first out) basis, ensuring that transactions are processed in the order they were generated/received. When a transaction results in an error, Maximo generates an error file and does not process subsequent transactions in the queue until the error is cleared.

A predefined cron task, JMSQSEQCONSUMER, polls the queues. There are two instances of the task, one that polls the inbound queue and one that polls the outbound queue. The cron task has the following configurable parameters.

**Cron Task Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MESSAGEPROCESSOR</td>
<td>Java class that processes the transactions from the queue. Maximo provides this class.</td>
</tr>
<tr>
<td>QUEUENAME</td>
<td>Queue JNDI name, as defined during the creation of the queue on the application server. Maximo provides a default name.</td>
</tr>
<tr>
<td>SELECTOR</td>
<td>Optional where clause for configuring an instance of the cron task to process a subset of transactions in the queue.</td>
</tr>
<tr>
<td>TARGETENABLED</td>
<td>Optional Boolean flag that controls whether the cron task runs in a specific Maximo application server. The default is 0 (false).</td>
</tr>
</tbody>
</table>
Continuous Queue

The continuous queue is a JMS queue with a message driven bean (MDB) as a consumer.

By default, this queue is configured for inbound interfaces only. It does not process transactions in FIFO sequence, as the sequential inbound queue does, but in a multi-threaded mode that allows for better performance. When a transaction results in an error, Maximo generates an error file, then continues processing subsequent messages in the queue.

You can configure the following properties of the continuous queue:

- Number of beans
- Caching of messages
- Redelivery delay

Considerations

You can improve performance by increasing the number of MDBs for a particular queue and/or introducing additional application servers via a cluster. However, since transactions are processed in a multi-threaded mode, errors may occur due to the random order of processing.

The following examples describe errors that can occur. In both scenarios, the integration error management processing might successfully reprocess the error before the system administrator is able to review it.

Example

You are batch loading a large volume of item and inventory transactions via the continuous queue, and multiple inventory records exist for the same item number. If an inventory transaction for Item A is processed before the item transaction that adds Item A to Maximo, the inventory transaction errors out because Item A does not exist. Processing continues with the next transaction. Eventually the item transaction for Item A is processed and Item A is added to Maximo. Maximo can then successfully reprocess the inventory transaction that errored out earlier. In this case, the error corrected itself without manual intervention.

This type of scenario can occur when you load related transactions via the continuous queue at the same time. It is more likely to happen when the volume of transactions is high, but can occur whenever two transactions process related data concurrently.

Example

Two transactions try to update the same Maximo record at the same time. One will succeed and the other will fail. However, the error management processing of Maximo should be able to successfully process the second transaction after the first update has been completed.
Continuous Queue

Number of Beans

You should understand how your application server manages MDBs, before you configure the number of beans yourself. Server-specific extensions control the maximum number of beans that can be created. By default, the server is configured to control the number of beans based on transaction volume.

The following example shows how to set the number of beans. For more information, refer to the documentation for your application server.

Entry to Set Number of Beans

<table>
<thead>
<tr>
<th>Server</th>
<th>Deployment Descriptor File</th>
<th>Sample Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEA WebLogic</td>
<td>weblogic-ejb-jar.xml</td>
<td>&lt;pool&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;max-beans-in-free-pool&gt;1&lt;/max-beans-in-free-pool&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;initial-beans-in-free-pool&gt;0&lt;/initial-beans-in-free-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pool&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;/pool&gt;</td>
</tr>
<tr>
<td>IBM WebSphere</td>
<td></td>
<td>From the IBM WebSphere Administrative Console,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>choose Default messaging provider &gt; JMS activation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>specification &gt; meajmsact &gt; Maximum concurrent</td>
</tr>
<tr>
<td></td>
<td></td>
<td>endpoints</td>
</tr>
</tbody>
</table>

Caching of Messages

Continuous queue processing uses a parameter called the pipeline size, which controls the number of messages the server will cache per instance of the MDB. The default value of the parameter is –1, which allows the server to manage the pipeline size. The value is not configurable in this release.

Redelivery Delay

Redelivery delay is the time that elapses between a message erroring out and being reprocessed by the queue. The message is not visible in the queue for the period of time specified for redelivery delay. This delay improves performance by processing other messages instead of immediately processing the error message. The default delay is 30 seconds.

Since the same connection factory is used for both the sequential and continuous queues, set the redelivery delay value in the destination (queue) level configuration rather than the connection factory level configuration, so as not to affect the sequential consumer.
Selectors

Selectors act as *where* clauses in the JMS queue consumer. They can be applied to header and message properties of a JMS message, in both the continuous and sequential queues.

You specify selectors in the following ways.

**Selector Specification**

<table>
<thead>
<tr>
<th>Type of Queue</th>
<th>Where to Identify Selector</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequential queue</td>
<td>As a property of the cron task</td>
</tr>
<tr>
<td>Continuous queue</td>
<td>In the ejb-jar.xml of the MDB</td>
</tr>
</tbody>
</table>

**Cron Task Tab with SELECTOR Property**

Applying selectors virtually splits a queue into smaller queues, each of which contains a subset of data that is being consumed by a separate cron task. An error in one subset of the data does not stop processing in the others.

While selectors provide flexibility in separating the processing of transactions, they impair the performance of polling processing. Depending on the volume of transactions, you might prefer to implement multiple queues rather than one queue with multiple selectors. The former option generally results in better performance.
Example

Adding the following statement to the SELECTOR property of the SEQQIN instance of the JMSQSEQCONSUMER will cause the cron task to process the MXPOInterface and MXPRInterface transactions from the corresponding external system:

\[\text{SENDER=’EXTSYS1’ and INTERFACE in \{’MXPOInterface’, ’MXPRInterface’\}}\]

Example

If two external systems send data to an inbound sequential queue, an error in any record will cause Maximo to stop processing all transactions in that queue, to maintain FIFO order. Creating multiple instances of a cron task, each with a selector that processes a different external system, can prevent an error in one system from stopping transactions from the second system.

**CAUTION**

Ensure that the *where* clauses in selectors identify mutually exclusive sets of transactions, but include all the transactions being inserted into the queues.

## Queue Utilities

The application servers let users see the properties and number of messages in a queue, but not the content of the messages. Maximo provides the following utilities for viewing and deleting messages in the queues.

### Queue Utility Files

<table>
<thead>
<tr>
<th>Purpose</th>
<th>File</th>
</tr>
</thead>
<tbody>
<tr>
<td>View messages</td>
<td>viewqueue.bat and viewqueue.sh</td>
</tr>
<tr>
<td>Delete messages</td>
<td>deletequeue.bat and deletequeue.sh</td>
</tr>
</tbody>
</table>

The viewqueue utility places the data in the view folder under the integration global directory.

**CAUTION**

Use the deletequeue utility only in a test or development environment. The queue error management processing provides a mechanism for deleting individual messages from production queues.
Queue Creation

Users can create additional queues if necessary. To do so, complete the following actions:

1. Create the queue on the application server.

   For more information, see Appendix D, "Creating the JMS Queues," on page D-1.

2. To configure the queue, use the **Add/Modify Queues** option on the Select Action menu in the External Systems application.

   Define a queue that uses MDBs as a continuous queue, and a queue that does not use MDBs as a sequential queue.

3. If you create a sequential queue, perform one of the following actions:

   ▼ Create an instance of the JMSQSEQCONSUMER cron task to process transactions from this queue.

      For more information, refer to the online help for the Cron Task Setup application.

   ▼ Implement a custom consumer.

4. To assign the queue to an external system, use the System tab in the External Systems application.

Queue Security

For information about securing the JMS queues, see Chapter 12, "Security," on page 12-1.
This chapter highlights the security options provided for the multiple entry and exit points through which integration transactions can be processed in Maximo. Users can enable security as needed, based on their implementation.

This chapter is directed to the system administrator. It contains the following sections:

- Securing Integration Queues
- Securing the Integration Gateway
- Securing the Gateway Web Service
- Securing the Servlet (HTTP)
- Securing Maximo Web Services
- Securing the Remote Integration APIs (MicService)
- Securing Interface Tables
- Outbound Security
Securing Integration Queues

The JMS queues used by integration processing in Maximo support basic J2EE security (user id and password based authentication and authorization).

Assigning a user ID and password prevents unauthorized outside access to the queues, even if the JNDI name of the queue is known. Multiple queues can use the same or different user IDs.

You can set these J2EE restrictions via the server’s administrative console. To allow the Maximo Integration producer and consumer programs to access the queue, you must enter the same user ID and password in the Add/Modify Queues dialog box in the External Systems application. You access this dialog via the Select Action menu.

For the continuous queue, update the MDB (the consumer of the continuous queue) deployment descriptors as follows, to access a secured queue.

1. Under the <enterprise-beans> section in the ejb-jar.xml file, add the elements shown below in bold text:

   ```xml
   <enterprise-beans>
     <message-driven id="MessageDriven_JMSContQueueProcessor_1">
       <ejb-name>JMSContQueueProcessor-1</ejb-name>
       <ejb-class>psdi.iface.jms.JMSContQueueProcessor</ejb-class>
       <transaction-type>Container</transaction-type>
       <message-driven-destination>
         <destination-type>javax.jms.Queue</destination-type>
       </message-driven-destination>
       <env-entry>
         <env-entry-name>MESSAGEPROCESSOR</env-entry-name>
         <env-entry-type>java.lang.String</env-entry-type>
         <env-entry-value>psdi.iface.jms.QueueToMaximoProcessor</env-entry-value>
       </env-entry>
       <security-identity>
         <run-as>
           <role-name>integrationuser</role-name>
         </run-as>
       </security-identity>
     </message-driven>
   </enterprise-beans>
   
   2. Under the <assembly-descriptor> section in the ejb-jar.xml file, add the elements shown below in bold text:

   ```xml
   <assembly-descriptor>
     <security-role>
       <role-name>integrationuser</role-name>
     </security-role>
     <container-transaction>
       <method>
         <ejb-name>JMSContQueueProcessor-1</ejb-name>
         <method-name>*</method-name>
       </method>
       <trans-attribute>Required</trans-attribute>
     </container-transaction>
   </assembly-descriptor>
   ```
3 Under the `<weblogic-enterprise-bean>` tag in the `weblogic-ejb-jar.xml` (server specific) file, add the elements shown below:

```xml
<security-role-assignment>
  <role-name>integrationuser</role-name>
  <principal-name>secured_user</principal-name>
</security-role-assignment>
```

`secured_user` is the user ID assigned to the queue in the application server and to the continuous queue in the Add/Modify Queues dialog box.

**Considerations**

Refer to the IBM WebSphere Administrative documentation for information on how to configure a WebSphere-specific security setup.

J2EE supports multiple principals acting on the same queue but having different access control, but this is not currently supported for integration processing.

**Confidentiality**

No data encryption is needed, as the queues are internal to the system.
Securing the Integration Gateway

The integration gateway implements an EJB as the entry point for receiving data from an external system. Being a J2EE core component, an EJB supports the basic J2EE security. Each remote method access in the EJB can be controlled using different principles.

1. Under the <enterprise-beans> section of the ejb-jar.xml file in the mboejb module, add the elements shown below in bold text:

   ```xml
   <enterprise-beans>
   <session id="Session_syncMaximoDataEJB">
      <ejb-name>syncMaximoDataEJB</ejb-name>
      <home>psdi.iface.gateway.MEAGatewayHome</home>
      <remote>psdi.iface.gateway.MEAGateway</remote>
      <ejb-class>psdi.iface.gateway.MEAGatewayBean</ejb-class>
      <session-type>Stateless</session-type>
      <transaction-type>Container</transaction-type>
   </session>
   <security-role-ref>
      <description>User access to MEA Gateway</description>
      <role-name>integrationuser</role-name>
      <role-link>integrationuser</role-link>
   </security-role-ref>
   ...
   ...
   </enterprise-beans>
   
   NOTE This example shows security using a single principal only.

2. Under the <assembly-descriptor> section in the ejb-jar.xml file, add the elements shown below in bold text:

   ```xml
   <assembly-descriptor>
   <security-role>
      <description>User access to MEA Gateway</description>
      <role-name>integrationuser</role-name>
   </security-role>
   ...
   ...
   <method-permission>
      <description>Gateway permissions</description>
      <role-name>integrationuser</role-name>
      <method>
         <ejb-name>syncMaximoDataEJB</ejb-name>
         <method-name>processExternalData</method-name>
      </method>
   </method-permission>
   ...
   ...
   </assembly-descriptor>
   
   NOTE To define more fine-grained access control, you can include multiple method permission tags.
3  Under the `<weblogic-enterprise-bean>` tag in the `weblogic-ejb-jar.xml` (server specific) file, add the elements shown below in bold text:

```xml
<security-role-assignment>
  <role-name>integrationuser</role-name>
  <principal-name>secured_user</principal-name>
</security-role-assignment>
```

*Secured_user is the user ID assigned to the EJB on the application server.*

**NOTE**  Refer to the IBM WebSphere Administrative documentation for information on how to configure a WebSphere-specific security setup.

4  To create a secure context for invoking the EJB, perform one of the following procedures:

- Add the following sample code to the client code:

```java
Properties env = new Properties();

if(userid != null && password != null)
{
  env.put(Context.SECURITY_CREDENTIALS, password);
  env.put(Context.SECURITY_PRINCIPAL, userid);
}

Context ctx = new InitialContext(env);
// instead of using the default InitialContext() constructor
```

- Use the default InitialContext constructor and pass the security information via the –D parameters in the .bat/.sh script that launches the client.

```
-Djava.naming.security.principal=<username>
-Djava.naming.security.credentials=<password>
```

**Confidentiality**  Data encryption can be done using the SSL version of iiop protocol (in the provider URL) while communicating with the EJB. For more information, refer to the documentation for your application server.
Securing the Gateway Web Service

The gateway Web service sits on top of the gateway EJB and inherits all the security features of the gateway EJB. The user ID- and password-based basic authentication for the EJB also applies to the Web service.

The calling client code must pass the secure SOAP context while invoking the Web service. For Web service invocation, ensure that the client program uses the user name and password in the JAX-RPC Call object, as shown below:

```java
call.setProperty(Call.USERNAME_PROPERTY, <username>);
call.setProperty(Call.PASSWORD_PROPERTY, <password>);
```

*Username* and *password* are the EJB principal/credentials used for the method level permissions.

**Confidentiality**

Data encryption can be done by using SSL (HTTPS) to invoke the Web service. Set up SSL in the server, following the server documentation. Modify the URL for the invocation as shown below in bold text:

```text
https://<host>:<port>/meaweb/services/syncMaximoDataService?wsdl
```

For more information, see Chapter 8, "Integration Gateway," on page 8-1.
Securing the Servlet (HTTP)

Inbound HTTP posts to Maximo Integration are handled by a servlet, which is a J2EE component and follows the J2EE security principles.

HTTP basic authentication can be used to secure the servlet provided with Maximo and allow only authorized users with a valid user name and password to post an XML transaction to Maximo. To enable HTTP basic authentication, modify the web.xml and weblogic.xml files of the MEA Web Application in the following way:

1. Remove the comments from the <security-constraint> section of the integration servlet, as shown below:

   ```xml
   <web-app>
     <display-name>MEA Web Application</display-name>
     ...
     <security-constraint>
       <web-resource-collection>
         <web-resource-name>Integration Servlet (HTTP POST)</web-resource-name>
         <description>Integration Servlet (HTTP POST) accessible by authorized users</description>
         <url-pattern>/measervlet/*</url-pattern>
         <http-method>GET</http-method>
         <http-method>POST</http-method>
       </web-resource-collection>
       <auth-constraint>
         <description>Roles that have access to Integration Servlet (HTTP POST)</description>
         <role-name>integrationuser</role-name>
       </auth-constraint>
       <user-data-constraint>
         <description>data transmission guarantee</description>
         <transport-guarantee>NONE</transport-guarantee>
       </user-data-constraint>
     </security-constraint>
   </web-app>
   ``

   **NOTE** The preceding <security-constraint> section refers to a single role, *integrationuser*, which is defined farther down in web.xml file, as shown below. By default, this section is not commented out.

   ```xml
   <security-role>
     <description>An Integration User</description>
     <role-name>integrationuser</role-name>
   </security-role>
   ```
Securing the Servlet (HTTP)

2 In the weblogic.xml file, add the following element. This file contains a mapping of role names to principals.

```xml
<weblogic-web-app>
  <security-role-assignment>
    <role-name>integrationuser</role-name>
    <principal-name>integrationusers</principal-name>
  </security-role-assignment>
</weblogic-web-app>
```

**NOTE** Refer to the IBM WebSphere Administrative documentation for information on how to configure a WebSphere-specific security setup.

3 Change the principal name to match the principal name (user ID) on the application server.

By default, the `integrationuser` role is mapped to a single principal name (user ID) called `integrationusers`. If needed, you can define additional mappings of roles to principal names here.

**Confidentiality**

Data encryption can be done by using SSL (HTTPS) to post the XML transaction. Set up the SSL in the application server, with the appropriate digital certificates. For information about this setup, refer to the application server documentation.

Modify the URL for posting the transaction as shown below in bold:

```
https://<host>:<port>/meaweb/measervlet/ADAPTER
```
Securing Maximo Web Services

The object-based Web services can be secured by using HTTP basic authentication via standard J2EE security. This allows only authorized users with a valid user name and password to access Maximo Web services. To enable HTTP basic authentication, you must modify the web.xml and weblogic.xml files of the MEA Web Application.

**NOTE** Refer to the IBM WebSphere Administrative documentation for information on how to configure a WebSphere-specific security setup.

As part of Web service deployment, XML schema and WSDL files are generated on the server. These are accessible via HTTP using a servlet. Thus, there are two parts to securing the interface-based Web services:

- Restricting access to Web service resources such as schemas and WSDL files
- Restricting Web service invocation (transaction)

The web.xml file of the MEA Web application contains two `<security-constraint>` sections, one for Web service invocation, the other for Web service resources. By default, both sections are commented out. Each can be secured independently of each other.

1. Remove the comments from one or both `<security-constraint>` sections of the integration servlet, as shown below in bold text:

```xml
<web-app>
    <display-name>MEA Web Application</display-name>
    ...
    <security-constraint>
        <web-resource-collection>
            <web-resource-name>Integration Web Services</web-resource-name>
            <description>Integration Web Services accessible by authorized users</description>
            <url-pattern>/services/*</url-pattern>
            <http-method>GET</http-method>
            <http-method>POST</http-method>
        </web-resource-collection>
        <auth-constraint>
            <description>Roles that have access to Integration Web Services</description>
            <role-name>integrationuser</role-name>
        </auth-constraint>
        <user-data-constraint>
            <description>Data transmission guarantee</description>
            <transport-guarantee>NONE</transport-guarantee>
        </user-data-constraint>
    </security-constraint>
    ...
    <security-constraint>
        <web-resource-collection>
            <web-resource-name>Integration Web Service Resources (XML Schemas, WSDL)</web-resource-name>
            <description>Integration Web Service resources (XML Schemas, WSDL) accessible by authorized users</description>
        </web-resource-collection>
    </security-constraint>
</web-app>
```
Securing Maximo Web Services

```xml
<url-pattern>/wsdl/*</url-pattern>
<url-pattern>/schema/*</url-pattern>
<http-method>GET</http-method>
<http-method>POST</http-method>
</web-resource-collection>

<auth-constraint>
  <description>Roles that have access to Integration Web Service Resources (XML Schemas, WSDL)</description>
  <role-name>integrationuser</role-name>
</auth-constraint>

<user-data-constraint>
  <description>data transmission guarantee</description>
  <transport-guarantee>NONE</transport-guarantee>
</user-data-constraint>
</security-constraint>

2 The preceding <security-constraint> elements refer to a single <role-name>, integrationuser. Add integrationuser to the web.xml file, as shown below in bold text:

```xml
<security-role>
  <description>An Integration User</description>
  <role-name>integrationuser</role-name>
</security-role>
```

3 Add the mapping of role to principal in the weblogic.xml file, as shown in below in bold text:

```xml
<weblogic-web-app>
  <security-role-assignment>
    <role-name>integrationuser</role-name>
    <principal-name>integrationusers</principal-name>
  </security-role-assignment>
</weblogic-web-app>
```

**NOTE** Refer to the IBM WebSphere Administrative documentation for information on how to configure a WebSphere-specific security setup.

4 Change the principal name to match the principal name (user ID) on the application server. By default, the integrationuser role maps to a single principal name (user ID) called integrationusers. You can define additional mappings of roles to principal names here, if necessary.

5 If the Web services are secured by HTTP authentication, configure the user name and password in the Web Service Administration dialog box. You access this dialog box via the Select Action menu in the Integration Interfaces application. For more information, see Chapter 18, "Maximo Web Services," on page 18-1.

6 For Web service invocation, ensure that the client program uses the user name and password in the JAX-RPC Call object, as shown below:

```java
call.setProperty(Call.USERNAME_PROPERTY, username);
call.setProperty(Call.PASSWORD_PROPERTY, password);
```
Securing Maximo Web Services

Confidentiality

Maximo Web services can be securely deployed/invoked by using SSL (HTTPS). Set up the SSL in the application server, with the appropriate digital certificates. For more information, refer to the documentation for the application server.

When using SSL, modify the URL for accessing the Web service as shown below in bold:

```
https://<host>:<port>/meaweb/services/InterfaceName
```

Also configure the Integration Web Application URL in the Web Service Administration dialog box in the Integration Interfaces application. This URL is used to deploying and undeploy Web services, and can be done securely.
Securing the Remote Integration APIs (MicService)

Some MicService remote APIs have been given a secure access by forcing the user of those methods to provide the UserInfo object. If a valid UserInfo object is not provided, the method throws an exception and the call is not completed. Only remote methods that provide sensitive information or perform sensitive data transaction processing have been protected. They are:

- `exportData(..)`
- `deleteQueueData(..)`
- `processDataIn(..)`
- `processExternalData(..) (both versions)`
- `query(..)`
- `viewQueue(..)`

To invoke these methods, the caller must get a valid UserInfo object and pass it to the method to be able to get in through the secure layer. A UserInfo object is a serialized object containing user details (user, password, locale, language, and time zone information). It is used by Maximo for security purposes.

Confidentiality

Maximo uses Java RMI/JRMP. You can communicate to Maximo services via a secure version of JRMP protocol using SSL. For more information, refer to your Java RMI documentation.

Securing Interface Tables

Securing interface tables uses the default database authentication and authorization. If authentication and authorization are in effect, external programs that read or write to the interface tables must provide proper authorization. To read from and write to the interface tables, Maximo uses the `USERNAME` and `PASSWORD` values configured for the end point that implements the interface table handler. You configure these properties in the Add/Modify End Point dialog box in the External Systems application.

Outbound Security

The outbound router handlers have support for authorization and confidentiality. The EJB, HTTP, JMS, Web service, and interface table handlers have support for security. For more information, see Chapter 9, "Router," on page 9-1.
Maximo can be implemented within a cluster of application servers, and integration services can run across the cluster. This chapter discusses special considerations regarding Maximo Integration components running in a cluster. It is directed to the system administrator and it contains the following sections:

- Cron Task Configuration
- JMS Queues
- Global Directory
- Integration Gateway
Cron Task Configuration

The interface table cron task, IFACETABLECONSUMER, and the JMS queue cron task, JMSQSEQCOONSUMER, are designed to be cluster aware. By default the cron task framework runs a task in a randomly chosen server.

To target a specific server within a cluster, perform the following configuration. Assume the name of the cron task instance is instance1:

1. In the Cron Task Setup application, set the TARGETENABLED property for instance1 to 1.

2. In the Maximo application server1 setup, set -D crontaskname.instance1 to 1, and restart the server.

In this configuration, the task is not automatically failed over to another server if the targeted server crashes. To target another server in place of the one that crashed, perform configuration step 2 above, then restart the server that will take over the polling.

To configure a group of servers to support failover, set -D crontaskname.instance1 to 1 in multiple servers. For more information, see the following chapters:

- Chapter 5, "Interface Tables," on page 5-1
- Chapter 11, "JMS Queue Configuration," on page 11-1

The preceding configuration procedures do not apply to the continuous queue, which deploys an MDB consumer instead of a cron task.

JMS Queues

In the default configuration, the WebLogic and WebSphere JMS queues are pinned to one member of the cluster but are accessible via the JNDI tree replication across all the members on the cluster. WebLogic queue messages are persisted in a file store. Since the queue physically resides in one server, the queue must be transferred to another server in the event of a server crash. To move the queue to another server, complete the following steps:

1. Identify the JNDI name of the server to which the queue will be moved.

2. Copy the store file to a location in the new server. The folder structure must be the same as that of the original file, as set up during the JMS queue creation on the application server.

3. Use the WebLogic Server Administration Console to move the queue from the crashed server to the new server.

In WebSphere, the server automatically moves the message queue to a different server in the event of a server crash.
The preceding procedure applies to all JMS queues. For more information, refer to the WebLogic and WebSphere documentation regarding queue configuration.

Global Directory

In a cluster environment, the global directory must be accessible to all members of the cluster.

Integration Gateway

This section describes how the following gateway components are implemented within a cluster:

- EJB
- Servlet (HTTP)
- Web service

EJB

With a single server, the provider URL for accessing the JNDI tree is the single server URL. With a cluster, the provider URL is the URL of any one of the servers which has the EJB deployed. All members of the cluster share the JNDI tree, so accessing the JNDI tree of any member accomplishes the lookup. As a result of the lookup, the client will get a cluster aware proxy of the EJB, which will load balance all the subsequent calls using that proxy. This happens transparently to the client code, so there is no difference in the code for a single server and for a cluster setup.

Servlet (HTTP)

The integration servlet is deployed across all members of the server. With a single server, the URL is the HTTP(S) URL of that server. With a cluster, the URL is the HTTP(S) URL of the cluster which for the WebLogic or WebSphere server is the URL of the administrative server. All HTTP(S) calls are load balanced by the administrative server.

Gateway Web Service

The gateway Web service is deployed across all members of the server. With a single server, the WSDL URL HTTP(S) address points to the server’s HTTP address. With a cluster the WSDL URL HTTP(S) address and the target namespace point to the cluster URL (the administrative server’s URL). All the SOAP/HTTP(S) calls are load balanced by the administrative server.
Maximo Web Services

Maximo Web services are homogeneously deployed across all the server members in the cluster. Web service access for a cluster is the same as for a single server, except that the Web service URL and WSDL URL (optional) point to the cluster rather than to a specific machine in the cluster. All the SOAP/HTTP(S) calls are load balanced by the administrative server.

The following properties in the Web Service Administration dialog box must point to the cluster URL:

- Integration Web Application URL
- Integration UDDI Registry Inquiry URL
- Integration UDDI Registry Publish URL

You access that dialog box via the Select Action menu in the Integration Interfaces application.
Customization with Processing Rules

Processing rules let you change the behavior of predefined integration processing without having to write Java classes. Processing rules can access and evaluate values in XML and MBO fields, MBO sets, and interface and system controls; and they can change the values in XML and MBO fields, or stop or skip processing all or part of a transaction.

This chapter is directed to developers and support personnel. It contains the following sections:

- MBOs and Sub-records
- Defining a Processing Rule
- Defining Conditions and Evaluations
- Interface Controls
MBOs and Sub-records

An integration object consists of one or more sub-records that correspond to Maximo MBOs. During inbound integration processing, Maximo creates MBOs and populates the MBO fields from the corresponding sub-record fields in the integration object, before applying standard Maximo application processing. During outbound processing, Maximo populates the sub-record fields from the corresponding fields in the original MBO. With the exception of certain generic integration fields, MBOs are not updated in outbound transactions.

When defining processing rules, use the following guidelines to determine if you should apply the rule to an integration object sub-record or to a MBO:

- In the outbound direction, you can apply processing rules to integration object sub-records only.
- In the inbound direction, you can apply processing rules to integration object sub-records or MBOs.
  - If an inbound rule changes the value of a key field, apply it to the integration object sub-record.
  - If an inbound rule does not evaluate or manipulate a MBO or MBO set, apply it to the integration object sub-record.
  - If an inbound rule evaluates or manipulates a user-defined field, apply it to the integration object sub-record.
  - If an inbound rule evaluates or manipulates a MBO or MBO field, apply it to the MBO.

**NOTE** As much as possible, apply all rules for inbound interfaces to either MBOs or to integration object sub-records; that is, avoid creating rules for both MBOs and integration object sub-records. Defining processing rules for both results in increased processing time for inbound transactions.
Defining a Processing Rule

A processing rule defines an action to be performed on a field in a sub-record or MBO, or on the sub-record or MBO itself. You define a processing rule on the Inbound Processing Rules or Outbound Processing Rules tab of the Integration Interfaces application. The application displays the inbound or outbound integration points for the selected interface. After you select the applicable integration point, the application displays the sub-records that comprise the corresponding integration object. Select the sub-record on which you will apply the processing rule.

The following sections of this chapter address the following aspects of defining a processing rule:

- **Defining the Processing Rule**
  - Specifying the sub-record or MBO to which the rule applies
  - Specifying when to trigger the processing rule
  - Specifying the processing rule action
  - Specifying the processing sequence

- **Defining Conditions and Evaluations**
  - Specifying a condition
  - Specifying the type of evaluation
  - Specifying the data to be evaluated
  - Specifying when to evaluate data
  - Specify evaluation criteria
  - Specify a comparison field

Specifying the Sub-record or MBO

To define a processing rule, you begin by specifying the sub-record or MBO to which the rule applies.

**NOTE** If the integration object is a merged integration object, define the rule for each sub-record or MBO within the integration object. Since the sub-records in merged integration objects use common fields, any processing action that applies to a field must be defined for each sub-record or MBO that contains the field. For example, to validate that a receipt does not contain a null GLDEBITACCT field, you have to add the processing rule for that field in both the MATRECTRANS and SERVRECTRANS sub-records or MBOs.
Defining a Processing Rule

Specifying when to Trigger the Processing Rule

A processing rule is triggered by a database action on the primary MBO within the corresponding integration object (or on any MBO within a merged integration object). You can specify that one or more of the following actions trigger the rule:

- An insertion to the primary MBO
- A deletion to the primary MBO
- An update to the primary MBO

**NOTE** You can place the rule on a primary or a child sub-record or MBO, but the database action that triggers a rule occurs on the primary MBO associated with the integration object.

**Example**

To prevent users from changing the item number on an existing POLINE on a purchase order, place a processing rule with a Stop action on the POLINE MBO and specify that the rule is to be applied when an update occurs on the primary (PO) MBO. This rule will perform the Stop action if a user changes an item number while updating a PO.

**NOTE** When an outbound transaction is generated via the Data Export feature rather than through a user action in Maximo, all processing rules that are enabled will be applied, regardless of the trigger settings that you specify.

Specifying the Processing Rule Action

A processing rule can act on an interface as a whole—for example, bypass a transaction—or it can manipulate the value in a data field within the transaction.

Three processing rule actions—Stop, Skip, and Skipchildren—act on an interface transaction as a whole. Four processing rule actions—Combine, Split, Set, and Replace—manipulate the value in a data field within an interface transaction.

**Message Processing Actions**

The following three processing rule actions stop or skip the interface transaction as a whole, or skip entire sub-records within the transaction:

- SKIP
- STOP
- SKIPCHILDREN

**Skip Action**

The Skip action bypasses a transaction that meets the specified criteria. Maximo will not process the inbound transaction and will not send the outbound transaction to an external system. Skip processing does not generate an error and it updates the maximo.log file with the rule that caused the skip action.

For inbound transactions, Maximo clears the message from the inbound queue, as it would a successfully processed message. For outbound transactions that are skipped, nothing is written to the queue.
Maximo provides some predefined rules with a Skip action. These rules look up interface control values to ensure that outbound transactions are in a valid status before being sent to the external system.

**Stop Action**

The Stop action halts the processing of a transaction that meets the specified criteria. An outbound transaction is rolled back and an error is thrown to the Maximo user. An inbound transaction remains in the inbound JMS queue and, if the error was the result of a direct invocation of the inbound interface (via a Web service), the calling program is notified of the error.

Maximo does not provide predefined rules with a Stop action. This option is a utility for users to customize the behavior of an interface.

Whenever possible, use the Skip action rather than the Stop action for inbound interfaces. The latter results in a processing error and, depending on the source of the data, the transaction remains in the inbound queue or the initiator receives an error response. These results do not occur when you use the Skip action.

**NOTE**

If a processing rule with a Stop action applies to an interface generated by the Data Export feature, Maximo treats the Stop action as it would a Skip action. If the Stop action evaluates true, the transaction will be skipped and the user will be notified of that action.

**Skipchildren Action**

The Skipchildren action deletes the child records of the sub-record or MBO to which the rule is applied. Apply the processing rule on the sub-record or MBO whose child level records will be skipped.

**Example**

Use the Skipchildren action on a PO to strip POLINE data and its children from an purchase order when a status change occurs and the external system does not need the accompanying PO information.

**Field Transformation Actions**

The four field transformation actions manipulate data fields within a transaction. The four actions are:

- SET
- REPLACE
- COMBINE
- SPLIT

Apply the field transformation rule to the sub-record or MBO that contains the field to be transformed. A field transformation rule can be applied to a single field or multiple fields in the selected sub-record.

You specify how the rule will manipulate the data, in the action-specific dialog box that you access via the Sub-Record Fields or MBO Fields button on the Inbound Processing Rules tab or the Outbound Processing Rules tab. For details about the fields in these dialog boxes, refer to the online help for the Integration Interfaces application.

**NOTE**

The Data Export feature does not recognize changed fields. An error will be reported if an interface generated by the Data Export feature uses a processing rule that evaluates whether a field has changed. For more information, see “Specifying when to Evaluate the Field” on page 14-15.
Defining a Processing Rule

Set Action

The Set action writes a value to a specified data field. When you define the rule, you specify the data to be set and the source of the new value, and indicate if the rule always writes the new value to the target field or only writes it when the field is null (the default action). You can use this action to initialize the value in a data field.

**Set MBO Fields Dialog Box**

**CAUTION** If the rule always writes the new value to the target field, it overwrites any existing value in the field.

The source of the new value can be one of the following:

- A value type interface control
- A hard-coded value
- A system control (in the MAXVARS database table)
- Another field in the specified sub-record or MBO
- A field in a related MBO
**Replace Action**

The Replace action replaces a value in a data field with another value. When you define the rule, you specify the name of a cross-reference control that contains the original and replacement values for the data field. The source of the new value must be a cross-reference type control.

**Replace Sub-Record Field Values Dialog Box**

![Replace Sub-Record Field Values Dialog Box]

**Example**

Use this action when an external system and Maximo have different identifiers for the same entity. For example, the plant identifier of an external system might translate to a Maximo site identifier.

**NOTE**

To replace a field value in both inbound and outbound transactions—such as the Maximo SITEID value in an outbound interface with an external PLANTID value, and to replace the external PLANTID value in an inbound interface with a Maximo SITEID value—you can use a single cross-reference type control. For more information see “Interface Controls” on page 14-18.
Combining Action

The Combine action concatenates values from multiple source fields into a single target field. When you define the rule, you identify the target field, the source fields, and the sequence in which the source data is to be written to the target field. The source data can be a data field or an interface control that contains a data value. You can also specify an interface control containing the delimiter to separate the segments in the target field.

**Combine Sub-Record Field Values Dialog Box**

Use this action in an inbound processing rule when a mismatch exists between the Maximo definition and the external system definition of an entity; for example, a two-part external system key that maps to a single part key in Maximo.

**Example**

An inbound processing rule can combine a vendor ID and a vendor location field from an external system into the Maximo COMPANY field when creating the COMPANIES record in Maximo. (An outbound processing rule with the Split action can then separate the combined field into separate values when sending data to the external system.

For more information, see “The Split action is the reverse of the Combine action. The Split action separates the value in one field into multiple fields. When you define the rule, you identify one source field, one or more target fields, and the way in which the rule processor identifies segments of the source field.” on page 14-9.

**NOTE**

The source and target fields must be in the same MBO.

**CAUTION**

This action always overwrites the existing value in the target field. Ensure that the source and target fields are alphanumeric fields, or processing errors might occur.
The Split action is the reverse of the Combine action. The Split action separates the value in one field into multiple fields. When you define the rule, you identify one source field, one or more target fields, and the way in which the rule processor identifies segments of the source field.

**Split Sub-Record Field Values Dialog Box**

The source field(s) can be the following:

- A field in the selected sub-record or MBO
- An interface control containing the delimiter that separates the segments in the source field

The source and target fields must exist in the same MBO.

**CAUTION** This action always overwrites the existing value in the target fields. Ensure that the source and target fields are alphanumeric fields, or processing errors might occur.

**Example**

If you combined multiple fields in an inbound transaction, as in the preceding example, you might need to split the combined field into individual fields in the outbound direction.

There are two ways to identify how to split the field. You can specify the length of each segment of the source field, or you can identify a delimiter that separates the segments.

**Identifying Fixed Length Data Segments**

If the field length of each segment of source data is constant, use the Field Length option. The rule processor breaks up the source field from left to right, based on the length you specify for each field and the sequence of the fields, and sets their values accordingly.
Defining Conditions and Evaluations

Identifying Variable Length Data Segments

If the length of the source field segments is variable but the source field contains a distinct delimiter that identifies the segments, use the separator option. Do this by identifying an interface control that defines the separator. The same separator must delimit all the segments. The rule processor parses the source field from left to right looking for the delimiter, breaks up the string into multiple values, and moves each value into the designated target field.

Specifying the Processing Sequence

Processing rules are applied sequentially for each sub-record or MBO within an integration object, starting with the primary MBO and working down to the child MBOs. If you define multiple processing rules for a single sub-record or MBO, you can modify the default processing sequence on the Inbound Processing Rules or Outbound Processing Rules tab. This is especially important if a rule depends upon the successful result of an earlier rule.

NOTE
If Maximo successfully applies a rule with a Stop or Skip action, it does not check subsequent processing rules.

Defining Conditions and Evaluations

Processing rules are generally applied conditionally; that is, after evaluating one or more conditions that must be met in order for Maximo to carry out the record processing or field transformation action specified in the rule. These conditions can involve one or more evaluations, or comparisons, of data in an XML field, a MBO field, a MBO set, an interface control, or a system control.

Specifying a Condition

It is important to understand the difference between a condition and an evaluation. A condition is a grouping of one or more evaluations. Multiple conditions can be specified, and their sequence identified by the condition number.

Each evaluation returns a value of true or false. For example, if an evaluation checks if the values of two fields are equal, it returns true if the fields are equal and false if they are not.

NOTE
Conditions also return a value of true or false. If every evaluation within a condition is true, the condition is true. If any evaluation within the condition is false, the condition is false. If a processing rule contains multiple conditions, only one condition must be true in order for Maximo to perform the action associated with the processing rule.
Specifying the Evaluation Category

Before you define the specifics of an evaluation, you select the type of data to be evaluated. You do this by selecting the appropriate subtab in the Add/Modify Conditions dialog box.

The following table describes the evaluation categories.

**Processing Rule Evaluation Categories**

<table>
<thead>
<tr>
<th>Category</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML Field</td>
<td>Evaluate a value in an integration object sub-record field or compare the values in two sub-record fields.</td>
</tr>
<tr>
<td>MBO Field</td>
<td>Evaluate the value in a MBO field or compare the values in two fields in related MBOs. The MBO field can be part of the integration object definition or part of a MBO that can be accessed via a relationship with a MBO in the integration object definition.</td>
</tr>
<tr>
<td>MBO Set</td>
<td>Check for the existence of records in a related MBO.</td>
</tr>
<tr>
<td>Control</td>
<td>Evaluate the value in a value type or Boolean type interface control or a system control.</td>
</tr>
</tbody>
</table>

The following table shows valid combinations of evaluation category, processing direction (outbound or inbound), and record type (sub-record or MBO). Because inbound processing rules are applied before MBOs are built, they cannot evaluate MBO fields or MBO sets.

**Valid Processing Rule Evaluations**

<table>
<thead>
<tr>
<th>Direction of Processing Rule</th>
<th>XML Field Evaluation</th>
<th>MBO Field Evaluation</th>
<th>MBO Set Evaluation</th>
<th>Control Evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outbound</td>
<td>Available</td>
<td>Available</td>
<td>Available</td>
<td>Available</td>
</tr>
<tr>
<td>Inbound (Sub-record)</td>
<td>Available</td>
<td>Not available</td>
<td>Not available</td>
<td>Available</td>
</tr>
<tr>
<td>Inbound (MBO)</td>
<td>Available</td>
<td>Available</td>
<td>Available</td>
<td>Available</td>
</tr>
</tbody>
</table>
The following pages show examples of the user interface for each category of evaluation:

**XML Field Evaluation**

![XML Field Evaluation Diagram]

**MBO Field Evaluation**

![MBO Field Evaluation Diagram]
Defining Conditions and Evaluations

**MBO Set Evaluation**

For XML field and MBO field evaluations, you specify the field that you will evaluate. For a MBO evaluation, if the field is in a sub-record or MBO other than the one specified on the Inbound or Outbound Processing tab, you specify the MBO and the relationship needed to access the field.

**Control Evaluation**

Specifying the Field to Evaluate
Defining Conditions and Evaluations

Specifying the Type of Evaluation

Evaluations generally involve the comparison of two values or a check for the existence of a MBO set or a null value.

The following table lists the possible types of evaluations. Depending on the category of evaluation (XML field, MBO field, MBO set, or control), you will see a subset of these options in the user interface.

**Processing Rule Evaluation Types**

<table>
<thead>
<tr>
<th>Type of Evaluation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EQUALS</td>
<td>The value in the specified field is equal to the value of a second field (the comparison field).</td>
</tr>
<tr>
<td>NOTEQUALS</td>
<td>The value in the specified field is not equal to the value of a second field (the comparison field).</td>
</tr>
<tr>
<td>GREATER</td>
<td>The value in the specified field is greater than the value of a second field (the comparison field).</td>
</tr>
<tr>
<td>GREATEROREQUAL</td>
<td>The value in the specified field is greater than or equal to the value of a second field (the comparison field).</td>
</tr>
<tr>
<td>LESS</td>
<td>The value in the specified field is less than the value of a second field (the comparison field).</td>
</tr>
<tr>
<td>LESSOREQUAL</td>
<td>The value in the specified field is less than or equal to the value of a second field (the comparison field).</td>
</tr>
<tr>
<td>LIKE</td>
<td>The value contains the expected value.</td>
</tr>
<tr>
<td>NOTLIKE</td>
<td>The value does not contain the expected value.</td>
</tr>
<tr>
<td>ISNULL</td>
<td>There is no value or a null value in the specified field.</td>
</tr>
<tr>
<td>ISNOTNULL</td>
<td>There is a value in the specified field.</td>
</tr>
<tr>
<td>NONE</td>
<td>This option is available only if the When to Evaluate is selected as “Changed” or “Not Changed”. If this is selected, it means that no further evaluation needs to be done. The field value is not being evaluated; the evaluation is to see if the field had changed or not changed.</td>
</tr>
<tr>
<td>EXISTS</td>
<td>Records exist in the specified MBO set.</td>
</tr>
<tr>
<td>NOTEXISTS</td>
<td>No records exist in the specified MBO set.</td>
</tr>
</tbody>
</table>
Specifying when to Evaluate the Field

For XML field and MBO field evaluations, the processing rule first determines whether or not it actually needs to evaluate the specified data. It does this by checking the Evaluate When? field, which can have one the following values:

**Processing Rule Evaluation Criteria**

<table>
<thead>
<tr>
<th>Value</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHANGED</td>
<td>Maximo continues with the evaluation only if the specified field was changed by the activity that generated the transaction.</td>
</tr>
<tr>
<td>NOT CHANGED</td>
<td>Maximo continues with the evaluation only if the specified field was not changed by the activity that generated the transaction.</td>
</tr>
<tr>
<td>ALWAYS</td>
<td>Maximo continues with the evaluation whether or not the value of the specified field was changed by the activity that generated the transaction (default). Note: If you specify this option, you cannot specify a comparison type of None (see below).</td>
</tr>
</tbody>
</table>

Under certain conditions, updating a record will result in the changed attribute (changed="1") appearing on the corresponding field in the outbound transaction. Maximo uses this attribute to determine if the field meets the criteria in the Evaluate When? field. For more information about the changed attribute, see Chapter 4, "Maximo XML and Schema," on page 4-1.

This attribute does not appear in transactions generated via the Data Export feature. Evaluations that should be applied only when a value has changed or not changed, might not provide the right output in a Data Export scenario.

**NOTE**

The changed attribute does not apply to inbound transactions.
Specifying the Comparison Field

If a processing rule uses one of the first eight evaluation types (see page 14-14), it must specify the field (comparison field) with which it is making the comparison.

The following table lists the possible types of comparison fields. Depending on the type of evaluation (XML field, MBO field, MBO set, or control), you will see a subset of these options in the user interface.

**NOTE**
Comparison of an alphanumeric source field is case-sensitive.

### Processing Rule Comparison Fields

<table>
<thead>
<tr>
<th>Label</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface Control</td>
<td>Compare the value in the specified field with the value(s) in a list type or value type interface control. In the case of a list type control with multiple values, the evaluation is true if the value of the field equals any value in the list.</td>
</tr>
</tbody>
</table>

Example: Validate the STATUS of a purchase order. The current value in a STATUS field is WAPPR and the possible acceptable values that satisfy the condition are in a list type control called POSEND. The values in POSEND are WAPPR, APPR and CLOSE. If the evaluation type is EQUALS, the evaluation returns a True value.

| Value     | Compare the value in the specified field with a prespecified value. This option is available for user-defined conditions. |

Note: Regardless of the locale setting of the Maximo application server or the database, all decimal fields must use a period (.) as the decimal placeholder. There is no formatting of numbers to the left of the placeholder. This format applies to inbound and outbound data. For example, $1,738,593.64 must be in the following format: 1738593.64.

Example: A processing rule compares the value of the POLIN1 field with the value “SPARE”. If the evaluation type is EQUALS and the two values are the same, the evaluation returns a True value.

| MAXVAR    | Compare the value in the specified field with the value in a system control (a value in the MAXVARS database table). |

Example: Evaluate the OWNERSYSID on any interface to determine if it is the same as MAXVARS.MXSYSID.

<p>| Boolean   | Compare the value in the specified field with a Boolean value (true or false). |</p>
<table>
<thead>
<tr>
<th>Label</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comparison Field</td>
<td>Compare the value in the specified field with another field in the same MBO.</td>
</tr>
<tr>
<td></td>
<td>Example: Compare the GLDEBITACCT and GLCREDITACCT on a PO line or a journal entry to determine if they are equal.</td>
</tr>
<tr>
<td>MBO/Relationship/Field</td>
<td>Compare the value in the specified field with a field in a different MBO.</td>
</tr>
<tr>
<td></td>
<td>Example: Check the OWNERSYSID of inventory in Maximo for the item-storeroom values on a receipt line or a PO Line.</td>
</tr>
</tbody>
</table>
Interface Controls

Interface controls give users the ability to configure the behavior of any interface according to the requirements of individual organizations and sites. Both processing rules and Java classes can access interface controls for evaluation purposes.

Interface controls are defined separately for each adapter. One control may be used by multiple interfaces within the same adapter. The controls are configurable by external system; that is, two external systems that process the same interface (for example, an inbound purchase order interface) can share the same processing logic, class files and processing rules, yet process the data differently due to different control settings.

The controls that apply to each interface defined within the Maximo adapter are documented in Appendix B, "Maximo Adapter Interface Specifications," on page B-1. For the predefined interfaces, controls are evaluated only in cases where processing rules are specified for the interface.

Control Levels

All Maximo master data and documents are stored at the system, organization, or site level. For example, item data is stored at the system level, accounting information at the organization level, and storerooms, inventory, and work orders at the site level. An implied hierarchy exists among these levels, as organizations are defined for a system, and sites are defined within organizations. Accordingly, an interface control can be configured to override values at any of these levels.

**Interface Control Levels**

<table>
<thead>
<tr>
<th>Control Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>System level</td>
<td>A system-level value applies to all Maximo organizations and sites. If the control is not configured for organization- or site-level values, Maximo processing uses the system default. If the control is configured for organization- or site-level values but none exists for a particular organization or site, Maximo processing uses the system level value.</td>
</tr>
<tr>
<td>Organization level</td>
<td>An organization-level value applies to all Maximo sites within an organization. If a control is configured for organization-level values but none exists for a particular organization, Maximo processing uses the system level value.</td>
</tr>
<tr>
<td>Site level</td>
<td>A site-level value applies to a specific site within a Maximo organization. If a control is configured for site-level values but none exists for a particular site, Maximo processing uses the system level value.</td>
</tr>
</tbody>
</table>

**NOTE**

Data processed by interfaces that use a control with an organization or site override must be organization- or site-level data, respectively.
Control Types

Maximo gives you the ability to create four types of interface controls:

- Boolean type controls
- Cross-reference type controls
- List type controls
- Value type controls

Boolean Type Controls

A Boolean type interface control specifies a value of 0 (false) or 1 (true).

List Type Controls

A list type interface control contains a list of values. You can enter multiple values for the control and optionally assign a Maximo domain to the control. Assigning a domain ensures the validation of any value entered for that control, at any level. If a domain is not assigned, there is no validation of the values entered.

Example

Maximo sends work orders to an external system only if the status of the work order is APPR (approved) or COMPLETE. To determine whether to send the work order, Java code or a processing rule can check the status of a work order against a list type control that contains these two values.

Value Type Controls

A value type interface control contains a single value. You can enter a single value for the control and optionally assign a Maximo domain to the control.

Cross-Reference Type Controls

A cross-reference type control replaces one value with another. In an outbound interface, Maximo converts a Maximo value to an external system value and, in an inbound interface, Maximo converts an external system value to a Maximo value. You can optionally assign a Maximo domain to a cross-reference control. If a domain is specified, any Maximo value specified for the control is validated against that domain. If a domain is not specified, there is no validation of the values entered.

In general, cross-reference controls must have a one to one mapping between the Maximo value and the external value. If two Maximo values are associated with an external system value, or two external system values with a Maximo value, a processing error will occur.

One to many mappings can exist if the cross-reference control is being created to function as a multiplication control. A multiplication control is a cross-reference control that copies, or multiples, an inbound message for multiple organizations or site. It has one external value and multiple Maximo values. Multiplication controls are always external system-specific. You identify the control as a multiplication control on the Inbound Integration Points subtab of the Interface tab.
Interface Controls

Examples

Maximo sites correspond to an external system’s business units, but the two systems use different values for these entities. A cross-reference control can perform the translation between the two values. For example, a cross-reference control in an inbound interface can translate business unit EX001 to Maximo site MX001. In an outbound direction, the same control can translate MX001 to EX001.

Multiplication Control

A multiplication control can update the company in every organization in the Maximo database.

Creating New Controls

Unless you create a new interface or modify the processing of an existing interface, you probably will not need to create new controls. Modifying control values at the external system level is generally sufficient to customize predefined interface processing.

If you do need to create a new control, you can do so via the Add/Modify Adapters dialog box in the Integration Interfaces application. Keep the following two points in mind:

- Ensure that the control name is unique across all adapters you have installed.
- You must manually add the control to existing external systems that will use it. You do this on the Interface Controls tab in the External Systems application.
The integration framework provides multiple placeholders within the transaction flow where users can customize transactions via Java programs and XSL. Based on your requirements, you can use one or more of these placeholders, or user exits, for your customizations.

This chapter describes the user exits and, for each one, references a sample Java class or XSL file that has been provided with the Maximo Integration. You can see additional samples in the psdi.iface.samples directory.

**NOTE**
After changing Java class files or XSL files, you must rebuild and redeploy the Maximo EAR file.

For a better understanding of the role of exits in the processing flow, read this chapter in conjunction with Chapter 3, "Outbound and Inbound Processing," on page 3-1.

This chapter is directed to developers and support personnel. It contains the following sections:

- Outbound Customization
- Inbound Customization

Maximo also provides a user interface-based rules processing engine that lets you customize outbound and inbound transactions without writing Java code or using XSL. For more information, see Chapter 14, "Customization with Processing Rules," on page 14-1.
Outbound Customization

This section describes the placeholders, or user exits, within the outbound processing flow of the Maximo Integration.

Outbound Processing Flow

You can customize outbound integration processing in the following ways:

- In an integration point processing class
- With interface processing rules
- In the following Java exits:
  - First user exit (user exit preprocessing)
  - Interface processing class
  - Second user exit (user exit postprocessing)
- With XSL mapping
- In a handler exit
The following figure illustrates the hierarchy of standard Java exit classes for outbound processing.

**Hierarchy of Java Exit Classes for Outbound Processing**

![Hierarchy of Java Exit Classes](image)

If you use an adapter provided by IBM Corporation, the interface processing class will likely contain processing logic, in which case you can implement your code in the first or second user exit.

**NOTE** At various points in the customization process, you can access the integration object XML, the interface XML, or both. This chapter refers to the integration object XML as the IR (internal record) and the interface XML as the ER (external record).
Integration Point Processing

Processing classes exist for some, but not all, predefined integration points. These classes contain business logic to support the processing of the integration object. You can customize the processing class by extending the processing class and overriding specific methods.

The Javadocs list the methods which can be overridden for the integration point processing class.

This class can perform the following processing:

- Validate data
- Allow an inbound transaction to generate an event-based outbound transaction (recurrence); by default, Maximo does not allow this

You must create a new integration point and associate the integration point processing class with it. You identify the integration point processing class in the Integration Objects application, in the Outbound Integration Points data window of the Integration Point tab. You associate the new integration point with the interface in the Interfaces application, on the Outbound Integration Points subtab of the Interface tab.

Interface Processing Rules

For information about the interface processing rules engine that lets you customize outbound transactions without writing Java code or using XSL, see Chapter 14, "Customization with Processing Rules," on page 14-1.

User Exit Preprocessing

In the first user exit, you can change data that will affect the processing logic in the interface processing class. In this exit, only the integration object XML (IR) is available, as the XML has not yet been transformed to external system XML (ER). In the second exit, both the IR and ER are available for processing.

Integration processing for outbound transactions is based on Maximo events or initiated via the Data Export feature. Integration processing listeners subscribe to these events after the Maximo transaction is saved (after SQL statements updating the database) but before the transaction and database are committed. The MBO reference that is available in the outbound processing is also available after database update; for this reason you cannot set data to this MBO to update the database.

To update the database you can use JDBC (Prepare Statements) using the connection available in the exit framework, in which case no business validations will be performed. You also can instantiate a new MBO from the database and call save on the transition again. (You cannot call save on the MboSet because internally it will call save and commit.) If you create your own MBO, the save will be called a second time and all logic associated with this save will be performed twice, once from the Maximo save and second time from the exit save, possibly leading to some duplication. Therefore, use the
Outbound Customization

Maximo APIs rather than the integration framework to change data in Maximo.

Outbound transaction customization can be done in the first user exit using the following method:

```java
public StructureData setUserValueOut(StructureData irData)
```

This method can perform the following processing:

- Validate data
- Change Maximo data by changing the IR record to be sent to the external system
- Stop the transaction from being saved in the database by throwing a Maximo exception
- Stop the transaction from being sent to the external system by throwing a SKIP_TRANSACTION exception
- Log the transaction

You identify the interface user exit class in the Integration Interfaces application, on the Outbound Integration Points subtab of the Interface tab.

**Interface Processing Class**

If you use an adapter provided by IBM Corporation, the interface processing class will likely contain processing logic. If it does, implement your code in the first or second user exit.

If you create a new interface, customization can be done in the interface processing class using the following method:

```java
public StructureData setDataOut(StructureData sData)
```

This method can perform the following processing:

- Validate data
- Change Maximo data by changing the IR record to be sent to the external system
- Stop the transaction from being saved in the database by throwing a Maximo exception
- Stop the transaction from being sent to the external system by throwing a SKIP_TRANSACTION exception
- Log the transaction

You identify the interface processing class in the Integration Interfaces application, on the Outbound Integration Points subtab of the Interface tab.
Outbound Customization

User Exit Postprocessing

In the second exit, both the IR and ER are available for processing. Outbound transaction customization can be done in the second exit using the following method:

```java
public StructureData setUserValueOut(StructureData irData,
                     StructureData erData)
```

This method can perform the following processing:

- Validate data
- Change data by changing the ER record to be sent to the external system
- Map additional data from the IR to the ER
- Stop the transaction from being saved in database by throwing a Maximo exception
- Stop the transaction from being sent to the external system by throwing a SKIP_TRANSACTION exception
- Log the transaction

You identify the interface user exit class in the Integration Interfaces application, on the Outbound Integration Points subtab of the Interface tab.

XSL Mapping

After the Java exit processing is completed, there is a placeholder where you can implement an XSL file and manipulate the data to be sent to external system.

The XSL file is always called with the ER and IR. The XML input into the XSL file looks like the following example:

```xml
<Message>
  <Ir> ...
  ...
  ...
  </Ir>
  <Er> ...
  ...
  ...
  </Er>
</Message>
```

This XSL file can perform the following types of processing:

- Change data in the ER before sending the transaction to the external system
- Map additional data from the IR to the ER
You identify the XSL file in the Integration Interfaces application, on the Outbound Integration Points subtab of the Interface tab.

The XSL file must be part of the EAR and must be under the businessobjects/classes/... folder structure. Also, it must be registered in the dot notation format.

**Example**

If XSL file mapping.xsl is under the businessobjects/classes/psd/iface/xsl folder, it must be registered as follows:

```
psdi.iface.xsl.mapping
```

**NOTE** The .xsl extension is omitted when mapping.xsl is registered.

### Handler Exits

After a transaction is saved to the outbound queue, the polling process retrieves the transaction from the queue and, based on the end point configuration, sends it to an external system. Maximo provides multiple predefined methods (handlers) of sending data to an external system, including HTTP, EJB call, and interface tables. Some of these methods have user exit placeholders to allow for customization.

### The HTTP Processing Exit Class

An exit placeholder allows for customization when sending transactions from Maximo to an external system using HTTP. This exit class is optional and is called as part of the response from the HTTP call.

**HTTP Processing Exit Class**

In the default implementation, psdi.iface.router.DefaultHTTPExit, the response code received from the external system is compared to a range of response codes. The range used by the default implementation is 200 to 299. If the code lies outside that range, then the transaction is assumed to not have been delivered to the external system and an exception is raised.

With some external systems, the HTTP response returned to Maximo from an HTTP call may need to be interpreted. You can apply an exit class that evaluates the response line and determines if the message was accepted by the external system. If it was not, the code must raise an exception so that the XML transaction in the queue is marked in error and not deleted from the outbound queue. If it was accepted, Maximo removes the message from the outbound queue.

This class must implement the psdi.iface.router.HTTPExit interface and implement the following method:

```java
public void processResponseData(int responseCode, String responseMsg, byte[] msgBodyData)
```
Outbound Customization

This class can perform the following processing:

- Interpret the response code and throw an exception.

You enter the fully qualified name of the Java class in the HTTPEXIT property of the end point that implements the HTTP handler. You do this in the Add/Modify End Point dialog box, which you access via the Select Action menu in the External Systems application.

Example

HTTPResponseUser

This class does the following processing to analyze the response from the HTTP post:

1. Checks the response code from the HTTP post

2. If the response code is in the error range, logs the exception on the ERROR level and throws a Maximo exception

3. If the response code is valid, logs the transaction on the DEBUG level

The EJB Processing Exit Class

An exit placeholder allows for customization when sending transactions from Maximo to an external system using EJB. This exit class is optional and is called before the EJB is called.

EJB Processing Exit Class

The implementation of this Java class must resolve the method signature of the EJB that is being invoked through this handler and the parameters that method requires. If no value is specified for this property, Maximo applies a default exit called DefaultEJBExit. This default exit attempts to resolve the EJB’s method signature and parameters.

This class must implement the psdi.iface.router.EJBExit interface and the following methods:

```
public Class[] getClassParams()
```

The getClassParams() method returns the method signature in the form of an array of Java classes.

This method can perform the following processing:

- Create an array of parameter types that correspond with the signature of the external EJB method to be called

```
public Object[] getObjectParams(byte[] data, String interfaceName, String destinationName)
```

The getObjectParams() method returns the parameters of the EJB business method in the form of an array of Java Objects.
Outbound Customization

This method can perform the following processing:

- Create an array of parameters that correspond with the signature of the external EJB method to be called. The number of parameters must match the number of parameter types from getClassParams() method.

```java
public void responseOk(Object response) throws MXException
```

The responseOk() method is called after a successful EJB invocation.

```java
public void responseError(Exception e) throws MXException
```

The responseError() method is called with the originating exception as a parameter if an error is encountered during EJB invocation.

You enter the fully qualified name of the Java class in the EJBEXIT property of the end point that implements the EJB handler. You do this in the Add/Modify End Point dialog box, which you access via the Select Action menu in the External Systems application.

The JMS Processing Exit Class

An exit placeholder allows for customization when sending transactions from Maximo to an external system using JMS. This exit class is optional and is called before the JMS is called.

**JMS Processing Exit Class**

This class must implement the psdi.iface.router.JMSExit class and the following method:

```java
public Map getMessageProperties(Map metaData, byte[] data, Map origProps) throws MXException
```

This method can perform the following processing:

- Change the properties of the JMS message
- Split the data to multiple properties, to match the JMS message

You enter the fully qualified name of the Java class in the JMSEXIT property of the end point that implements the JMS handler. You do this in the Add/Modify End Point dialog box, which you access via the Select Action menu in the External Systems application.
Outbound Customization

The Web Service Processing Exit Class

An exit placeholder allows for customization when sending transactions from Maximo to an external system using a Web service. This exit class is optional and is called before the Web service is invoked.

**Web Service Processing Exit Class**

![Diagram of Web Service Processing Exit Class]

This class must implement the `psdi.iface.router.WSExit` interface and the following methods:

1. `public String getServiceName(Map metaData, String endpointURL, String serviceName, String interfaceName, String targetNameSpace) throws MXException`

   The `getServiceName()` method returns the service name of the Web service to be invoked.

2. `public String getEndpointURL(Map metaData, String endpointURL, String serviceName, String interfaceName, String targetNameSpace) throws MXException`

   The `getEndpointURL()` method returns the endpoint URL of the Web service to be invoked.

3. `public void responseOk(org.w3c.dom.Document response) throws MXException`

   The `responseOk()` method is called after a successful invocation of the external Web service.

4. `public void responseError(Exception e) throws MXException`

   The `responseError()` method is called with the originating exception as a parameter if an error was encountered during the Web service invocation.

5. `public boolean getOneWayWsInfo(Map metaData, String endpointURL, String serviceName, String interfaceName, String targetNameSpace, boolean oneWayWs) throws MXException`

   The `getOneWayWsInfo()` method returns a Boolean value that specifies whether the Web service being invoked is one-way.

6. `public String getSoapAction(Map metaData, String endpointURL, String serviceName, String interfaceName, String targetNameSpace, String soapAction) throws MXException`

   The `getSoapAction()` method returns the SOAPAction HTTP header to be used while invoking the Web service.

You enter the fully qualified name of the Java class in the **WSEXIT** property of the end point that implements the Web service handler. You do this in the Add/Modify End Point dialog box, which you access via the Select Action menu in the External Systems application.
There is a default implementation of the WSExit interface, psdi.iface.router.DefaultWSExit. This class overrides the getEndpointURL() method to concatenate the servicename at the end of endpoint URL, to form the new endpoint URL.

**NOTE**  Use this class name in the WSEXIT property if you communicate with a remote Maximo system via Web services.
Inbound Customization

This section describes the placeholders, or user exits, within the inbound processing flow of Maximo Integration.

**Inbound Processing Flow**

You can customize inbound integration processing in the following ways:

- In an interface table user exit
- In the following Java exits:
  - First user exit (user exit preprocessing)
  - Interface processing class
  - Second user exit (user exit postprocessing)
- With XSL mapping
- With integration object and MBO processing rules
- In an integration point processing class
- In a MBO user exit

The following figure illustrates the hierarchy of standard Java exit classes for inbound processing.

**Hierarchy of Java Exit Classes for Inbound Processing**

If you use an ERP adapter provided by IBM Corporation, the interface processing class will likely contain processing logic, in which case you can implement your code in the first or second user exit.
Interface Table User Exit

When using interface tables to receive transactions from an external system, you can perform customization in the polling program that retrieves the data from the interfaces and sends it to Maximo.

The interface table polling program is run via the Maximo cron task manager. The cron task, IFACETABLECONSUMER, has an optional property called EXITCLASS, where you can place the fully qualified name of a Java exit class.

The IR record from the interface tables will be represented as Java data structure [List], where the first element is always the action of the transaction. The remaining elements of the list are Map data structures with each Map representing a row in the interface table for that transaction. The keys in the Map are the column names and the values are the corresponding column values. All the column values [Integers, Date, Float, Double] are converted to their localized string format before setting them to the Map.

The class must implement the psdi.iface.intertables.IfaceTbExit interface and the following three methods:

```java
public void beforeQueue(long transid, String extSys, String ifaceName, List data, Connection conn)
```

This method is called after the data is pulled from the interface table and before the data is insert into one of the inbound queues.

```java
public void afterCommit(long transid, String extSys, String ifaceName, Connection conn)
```

This method is called after the data has been inserted to an inbound queue and deleted from the interface queue table, and the database commit is done.

```java
public void afterRollback(long transid, String extSys, String ifaceName, Connection conn)
```

This method is similar to the afterCommit method but called if the transaction rolled back.

This class can perform the following processing:

- Validate data
- Change external data by changing the IR record to be saved in Maximo
- Stop the transaction from to be saved in queue by throwing an exception. In this case the transaction will remain in the MXIN_INTER_TRANS table with error message and will be re-tried
Inbound Customization

- Stop the transaction from being sent to external system by throwing a SKIP_TRANSACTION exception. In this case transaction will not be saved in Maximo and removed from the queue.

- Log the transaction

The user exit using the afterCommit or afterRollback method can perform the following processing:

- Perform custom processing and cleanup
- Log the transaction

You identify this class in the Cron Task Setup application in the Configuration module.

XSL Mapping

The XSL file is always called with a combination of ER and IR, and the user can manipulate data to be sent to Maximo. The XML input into the XSL file looks like the following example:

```xml
<Message>
  <IR>
    .
    .
    .
  </IR>
  <ER>
    .
    .
    .
  </ER>
</Message>
```

This XSL file can perform the following processing:

- Change the data to be saved in Maximo by changing the IR
- Map additional data from the ER to the IR

You identify the XSL file in the Integration Interfaces application, on the Inbound Integration Points subtab of the Interface tab.

The output of the XSL map must be in Maximo XML format. The root element of this XML must be the name of the integration point for which this XSL map is intended.

The XSL file must be part of the EAR and must be under the businessobjects/classes/... folder structure. Also, it must be registered in the dot notation format.

**Example**

If XSL file mapping.xsl is under the businessobjects/classes/psd iface/xsl folder, it must be registered as follows:

```
psdi.iface.xsl.mapping
```

**Note** The .xsl extension is omitted when mapping.xsl is registered.
User Exit Preprocessing

In the first user exit, you can change data that will affect the processing logic in the interface processing class. In this exit, only the interface XML (ER) is available, as the XML has not yet been transformed to integration object XML (IR). In the second exit, both the IR and ER are available for processing.

Inbound transaction customization can be done in the first exit using the following method:

```java
public StructureData setUserValueIn(StructureData erData)
```

This method can perform the following processing:

- ▼ Validate data
- ▼ Change external data by changing the ER record before it is mapped to the IR record and saved in Maximo
- ▼ Stop further processing of the transaction throwing an exception. In this case the transaction remains in the queue to be retried
- ▼ Stop the transaction from being sent to the external system by throwing a SKIP_TRANSACTION exception. In this case transaction is not saved in Maximo and is removed from the queue
- ▼ Log the transaction

You identify the interface user exit class in the Integration Interfaces application, on the Inbound Integration Points subtab of the Interface tab.
Interface Processing Class

If you use an adapter provided by IBM Corporation, the interface processing class will likely contain processing logic. If it does, implement your code in the first or second user exit.

If you create a new interface, customization can be done in the interface processing class using the following method:

```java
public StructureData setDataIn(StructureData sData)
```

This method can perform the following processing:

- Validate data
- Change external data by changing the ER record before it is mapped to the IR record and saved in Maximo
- Stop further processing of the transaction throwing an exception. In this case the transaction remains in the queue to be retried
- Stop the transaction from being sent to the external system by throwing ‘SKIP_TRANSACTION’ exception. In this case the transaction is not saved in Maximo and is removed from the queue
- Log the transaction

You identify the interface processing class in the Integration Interfaces application, on the Inbound Integration Points subtab of the Interface tab.

User Exit Postprocessing

In the second exit, both the IR and ER are available for processing. Inbound transaction customization can be done in the second exit using the following method:

```java
public StructureData setUserValueIn(StructureData irData,
StructureData erData)
```

This method can perform the following processing:

- Validate data
- Change external data by changing IR record to be saved in Maximo
- Map additional data from the ER to the IR
- Stop further processing of the transaction throwing an exception. In this case the transaction remains in the queue to be retried
- Stop the transaction from being sent to the external system by throwing a SKIP_TRANSACTION exception. In this case the transaction is not saved in Maximo and is removed from the queue
- Log the transaction
You identify the interface user exit class in the Integration Interfaces application, on the Inbound Integration Points subtab of the Interface tab.

**NOTE** The Java user exits also provide an exit point that is executed just before the MBO is saved in Maximo. For more information, see "MBO User Exit Processing," on page 15-18.

### Integration Object and MBO Processing Rules

For information about the processing rules engine that lets you customize integration objects and MBOs in inbound transactions without writing Java code or using XSL, see Chapter 14, "Customization with Processing Rules," on page 14-1.

### Integration Point Processing

Processing classes exist for some, but not all, predefined integration points. These classes contain business logic to support the processing of the integration object. You can customize the processing class by extending the processing class and overriding specific methods.

The Javadocs list the methods which can be overridden for the integration point processing class.

**CAUTION** If you replace the predefined processing class associated with a predefined integration point, the integration object might not support the same functionality that it supported with the predefined processing class.

This class can perform the following processing:

- Validate data
- Change the data in an integration object
- Create new MBOs
- Call specific methods on a MBO

You must create a new integration point and associate the integration point processing class with it. You identify the integration point processing class in the Integration Objects application, in the Inbound Integration Points data window of the Integration Point tab. You associate the new integration point with the interface in the Interfaces application, on the Inbound Integration Points subtab of the Interface tab.
MBO User Exit Processing

The Java user exits also provide an exit point that is executed just before the MBO is saved in Maximo. This MBO user exit is called after Maximo processing and can process MBO objects created in Maximo by using the following method:

```java
public void setUserMboIn(MboRemote mbo)
```

where the mbo parameter is a reference to a primary MBO in the integration object

This method will be called only once for the primary MBO. For an XML transaction with multiple nouns, the MBO exit will be called once for each noun.

This method can perform the following processing:

- ▼ Validate data
- ▼ Stop the transaction from being saved in Maximo by throwing a Maximo exception. In this case the transaction remains in the queue and is retried
- ▼ Log the transaction

You identify the interface user exit class in the Integration Interfaces application, on the Inbound Integration Points subtab of the Interface tab.
Maximo lets you create new integration components (adapters, integration objects, interfaces, and external systems) and modify existing ones that do not completely meet your business requirements. This chapter explains how to create new components and modify existing ones. It is directed to developers and support personnel.

This chapter contains the following sections:

- Adding or Modifying a Component
- Creating an Integration Object
- Modifying a Predefined Integration Object
- Creating an Integration Point
- Modifying a Predefined Integration Point
- Creating Adapters and Interfaces
- Modifying Predefined Adapters and Interfaces
- Creating an External System
- Modifying a Predefined External System
Adding or Modifying a Component

Depending on the scope and requirements of your implementation, you can create new components or duplicate, modify, or extend the predefined components.

Duplicating an integration component and modifying the copy has several advantages. The copy becomes a user-defined entity, and modification restrictions that apply to the predefined component do not apply to the copy. Duplication also retains the original version of the component, in case you have to revert back to it.

Creating an Integration Object

Maximo provides numerous predefined integration objects, which form the basis for interfaces across multiple functional areas such as purchasing, work management, and inventory. If your business requires an integration for which a predefined object does not exist, you can create an integration object and interface to support this exchange of data to and from Maximo.

To create an integration object, you must complete the following activities:

- Determine the required data objects and fields
- Build the integration object

Determining the Required MBOs

Determining which data to include in an integration object requires some knowledge of the Maximo applications and MBOs. Begin by using the related application and examining the data model to determine which MBOs, database tables, and/or views contain the data you need to transfer between Maximo and an external system.

**NOTE**

An integration object is built from MBOs, so even if you determine the database tables where the required data is maintained, you must find out which MBOs populate these tables. Generally, Maximo has a one to one relationship between a MBO and a database table, though there are cases where multiple MBOs write data to a single table.
Example

Assume that you need to send purchase requisitions to and from Maximo, but Maximo does not provide a predefined integration object with purchase requisition data.* After using the purchasing requisition application, you determine that the data you need resides in the following tables: PR, PRLINE, PRCOST, PRTERM. The tables have the same name as the corresponding MBOs, so you will include the PR, PRLINE, PRCOST, and PRTERM MBOs in the integration object that you create. The resulting interface will contain the data fields in those MBOs.

* Maximo actually does provide a predefined purchase requisition integration object, MXPR.

NOTE  In this example, the Maximo database tables and the corresponding MBOs have the same name. This will not be true in all cases.

Building the Integration Object

After identifying the necessary MBOs, you create the integration object on the Integration Object tab in the Integration Objects application. To build an integration object containing the purchase requisition MBOs in the preceding example, perform the following activities. For step by step procedures, refer to the online help for the Integration Objects application.

CAUTION  Due to XML requirements, the name of the integration object must begin with an alphabetic character.

1  Enter PR as the primary (top-level) source MBO, then PRLINE as the child MBO. PR becomes the parent of PRLINE.

2  Select the predefined MBO relationship that contains the appropriate where clause linking the PR and PRLINE MBOs. If the application displays multiple relationships, examine the where clauses to determine which one to use.

3  Repeat the preceding steps for the remaining required MBOs. PRLINE becomes the parent to PRCOST, and you must indicate the relationship between those two MBOs. Do the same with PR (parent) and PRTERM (child).
Creating an Integration Object

When you finish, your screen will look like the following example:

*Integration Object Tab in Integration Objects Application*

NOTE

The preceding steps describe the process of creating a standard, hierarchical integration object. A merged integration object contains multiple MBOs that are mutually exclusive in a transaction and therefore do not have a hierarchical relationship. For example, the receipts integration object, MXRECEIPT, contains the material receipt MBO (MATRECTRANS) and the service receipt MBO (SERVRECTRANS). These two MBOs do not have a relationship; each is independent of the other, but they are processed by the same integration point and presented to the external system in the same interface. This structure lets the external system use one interface (for material and service receipts), rather than two, to integrate receipts with Maximo.

Alternate Keys

An inbound transaction normally uses the primary key of a MBO to look up and process records in Maximo. However, in some cases a primary key is an internally generated value that is not available to an external system. In such cases, the external system passes an alternate key, and Maximo uses that value to process the MBO.

If you need to specify an alternate key for a MBO in your integration object, do so by identifying the alternate key as a Maximo index, in the Database Configuration application of the Configuration module. That application also identifies the primary keys of the MBOs.
CAUTION  When using alternate keys, do not change or drop associated indexes relevant to the interfaces that you use. Doing so can result in the failure of inbound transactions.

Indexes Tab in Database Configuration Application

To instruct Maximo integration processing to use the alternate key, enter the name of the alternate key on the Integration Object tab in the Integration Objects application.

Integration Object Tab with Alternate Key

Example

The primary key of the PR MBO consists of PRNUM and SITEID. On an inbound transaction, Maximo normally tries to find the PR MBO by using the PRNUM and SITEID. If the integration object indicates an alternate key, Maximo uses that value, instead of the primary key, to access the MBO.
Creating an Integration Object

Required Fields

Maximo and the external system will likely need to exchange only a subset of the data columns in the MBOs within the integration object. Subject to certain restrictions, you can add and exclude integration object columns. You do this via the Persistent Fields, Non-Persistent Fields, and User Fields subtabs on the Integration Object tab.

Persistent Fields

By default, Maximo includes in integration objects all persistent columns in the component MBOs. A persistent column is a data field whose value a MBO writes to a database table after processing. You can exclude persistent columns that you are not mapping. For outbound transactions, Maximo writes only included persistent columns to the XML message. For inbound transactions, Maximo only updates only the MBO columns that are included in the integration object.

CAUTION

Do not exclude any column that is part of a primary or alternate key.

Non-Persistent Fields

By default, Maximo excludes from integration objects most non-persistent columns in the component MBOs. A non-persistent column is a temporary data field whose value a MBO uses for calculations, interim storage, and so on. You can include additional non-persistent columns in the integration object. For example, MBOs that contain the persistent column DESCRIPTION also contain the non-persistent column DESCRIPTION_LONGDESCRIPTION. The integration object automatically includes the former column and excludes the latter.

User Columns

You also can add user fields to an integration object. A user field is a data field that is required by processing rules, non-Maximo adapters, or custom code. It becomes part of the definition of the integration object, but it is not part of any MBO definition and no MBO uses it. You set the value of user fields through interface processing rules or Java exit classes.

Interface Table and Flat File Considerations

If you will use the integration object in interfaces tables or flat files, review the column names within the integration object to determine if duplicates exist. If they do, modify the Alias value for the duplicate columns, to ensure that all column names are unique and Maximo can generate the interface table or flat file without errors. For more information, see “Duplicate Columns and Aliases” on page 5-6.

Interface tables require that all columns included in the corresponding MBO have an Alias name of eighteen or fewer characters. Use the Add/Modify Alias dialog box in the Integration Objects application to change the Alias value for a MBO column. You access this dialog box via the Add/Modify Alias option in the Select Action menu.
Inbound Setting Restrictions

Most MBOs have a unique identifier column whose value is generated when the record is written to the database. A setting restriction on such columns ensures that the MBO will always generate the column value and not set it using a value sent from an external system. The Inbound Setting Restrictions dialog box displays predefined restrictions and lets you enter additional ones if necessary. You access this screen via the Select Action menu in the Integration Objects application.

**Inbound Setting Restrictions Dialog Box**

![Inbound Setting Restrictions Dialog Box](image-url)
## Modifying a Predefined Integration Object

You can modify predefined integration objects, subject to the following restrictions.

### Content MBOs of an Integration Object

You can add MBOs to a predefined integration object, but you cannot delete predefined MBOs from the integration object. You can circumvent this restriction by duplicating the predefined integration object, thereby creating a user-defined integration object, and deleting MBOs from the copy.

### MBO Columns

You can include and exclude persistent, non-persistent, and user columns, within the scope of the standard validations.

Outbound transactions automatically include the columns for MBOs that you add to an integration object. Test inbound transactions to ensure that the added MBO columns are processed successfully. If not, add an integration point processing class to handle the inbound processing.

### Inbound Setting Restrictions

Do not remove any predefined setting restrictions, or inbound processing might fail.

### Interface table and Flat File Considerations

If you will use the integration object in interfaces tables or flat files, see "Interface Table and Flat File Considerations," on page 16-6.

If you use interface tables, regenerate the table for every interface that uses the modified integration object.
Creating an Integration Point

After you create an integration object, you identify one or more integration points for the object. The integration point specifies the type of operation (Notify, Query, or Response) and the processing direction (inbound or outbound) that the integration object will support. You perform this activity on the Integration Point tab in the Integration Objects application.

Integration Point Tab in Integration Objects Application

Notify Operation
Integration points with a Notify operation are used to synchronize data between Maximo and an external system. The Notify operation applies to inbound and outbound transactions.

Query and Response Operations
Integration points with a Query operation are used to execute inbound queries on Maximo. The query result set is returned via an integration point with a Response operation. For more information about query and response operations, see Chapter 17, "Using Integration Queries," on page 17-1.
Creating an Integration Point

Processing Classes

You can optionally associate a custom Java class with an integration point.

Maximo automatically generates outbound transactions in the XML format of the integration object. Unless you are mapping the data to a different XML format, you do not need to provide an outbound processing class. If you do provide one, associate it with the appropriate interface. You do this in the Integration Interfaces application, on the Outbound Integration Points subtab of the Interface tab.

Outbound Integration Points Subtab in Integration Interfaces Application

Inbound transactions sometimes require a processing class to direct a MBO to perform certain processing. For example, changing the status of a purchase requisition (PR) requires that a processing class associated with the integration point call a specific method to do a PR status change. This behavior varies from MBO to MBO, as some inbound transactions require a processing class to direct the MBO to insert, update, or delete data, and some do not.

NOTE If multiple interfaces are associated with the same integration point, the processing class executes for each of those interfaces.
Modifying a Predefined Integration Point

You cannot delete predefined inbound or outbound integration points. To change the behavior of a predefined processing class associated with an integration point, create a new integration point for the corresponding integration object and use a custom processing class in place of the predefined class. For more information about extending a predefined processing class, see Chapter 15, "Customization with User Exits," on page 15-1.

Creating Adapters and Interfaces

After defining an integration object and integration point, you define the interface that will exchange the data in the integration object with the external system. This section details considerations in defining interfaces.

Creating an Adapter

Interfaces are grouped together by adapter type, so first determine whether you need to create an adapter or you can use the predefined Maximo adapter.

The following restrictions apply to your choice of adapter type:

- If you plan to expose your interfaces as Web services, you must use the Maximo adapter. Only interfaces defined within the Maximo adapter can be Web services-enabled.

- If you plan to use interface tables or the Data Import or Data Export feature, you must use an internal type adapter. The Maximo adapter is an internal type adapter.

  The Add/Modify Adapters dialog box contains a check box called Internal? A check in this box indicates an internal type adapter; no check indicates an external type adapter.

- To use the Maximo XML format, you can use the Maximo adapter or define your own internal type adapter.

- To map to a different XML format, create an external type adapter.
You maintain adapters in the Add/Modify Adapters dialog box, which you access via the Select Action menu in the Integration Interfaces application. You can create multiple adapters, if necessary.

**Gateway Properties**

The gateway properties of an adapter identify the location of the interface name and external system name within an XML transaction that is received via the integration gateway. You maintain these properties on the Gateway Properties tab of the Add/Modify Adapters dialog box. For more information about the gateway properties, see Chapter 8, "Integration Gateway," on page 8-1.

**Interface Controls and Processing Rules**

Interface controls are adapter properties that help to implement the data mapping and business rules performed by Java classes and processing rules. Each adapter has its own set of interface controls, and you can configure different values for each external system that uses the adapter. You maintain interface controls on the Interface Controls tab of the Add/Modify Adapters dialog box. For more information about interface controls and processing rules, see Chapter 14, "Customization with Processing Rules," on page 14-1.
Creating Interfaces

After determining which adapter to use, you can create new interfaces for the adapter. You perform most of the activity on the Interface tab in the Integration Interfaces Application.

Interface Tab in Integration Interfaces Application

Integration Objects
For an internal type adapter, specify the integration object that is the basis for the interface. For an external type adapter, do not specify an integration object.

Interface Tables
For an internal type adapter, you can optionally change the interface table name for the interface.

Query and Response Type Interfaces
To create integration queries, you must first create a reply interface (operation = Response). You then create the query type interface (operation = Query) and associate the reply interface with it. For more information, see Chapter 17, "Using Integration Queries," on page 17-1.

Ignore the Reply Interface field when creating a data synchronization (operation = Notify) interface.

Inbound and Outbound Processing
Associate an outbound interface with the outbound integration point whose integration object will provide the content of the interface. By default, Maximo writes outbound transactions in the XML format of the integration object. If the external system requires a different XML format, you can use an interface processing class, a user exit class, and/or an XSL map file to map the XML to that format or to apply business rules.

Associate an inbound interface with an inbound integration point. As with outbound processing, you can provide an interface processing class, a user exit class, and/or an XSL map file, if necessary, to map the interface XML to the integration object XML. If the inbound data is not in the XML format of the integration object following the execution of the interface processing class, a processing error will occur.
Creating Adapters and Interfaces

For more information about XML, see Chapter 4, "Maximo XML and Schema," on page 4-1.

You also can implement processing rules or Java classes to customize inbound or outbound interface processing. For more information, see the following chapters:

- Chapter 14, "Customization with Processing Rules," on page 14-1
- Chapter 15, "Customization with User Exits," on page 15-1

Integration Points

An inbound interface can map to one or more integration points. For example, provided the interface contained all the required data elements, a purchase order interface could update a vendor and a purchase order by mapping to the PO integration point (MXPOIN) and the vendor integration point (MXVENDORIN).

When mapping to multiple integration points, specify the sequence in which Maximo must process the integration points, in the Process Order field.

**NOTE**  Mapping to multiple integration points involves a single database transaction, so an error in one part of the processing will cause the whole transaction to fail.

Multiplication Controls

An inbound interface can use a multiplication control, which is an interface control that multiplies a transaction for multiple sites or organizations within Maximo. For example, a multiplication control can direct inbound processing to multiply a single vendor transaction and make it available to every organization within Maximo. For more information, see Chapter 14, "Customization with Processing Rules," on page 14-1.

Web Services

If the interfaces are defined within the Maximo adapter, you can expose your inbound interface as a Web service. For more information, see Chapter 18, "Maximo Web Services," on page 18-1.
Modifying Predefined Adapters and Interfaces

Rather than create a new adapter and new interfaces, you might prefer to duplicate predefined interfaces and change the copies. You can modify predefined adapters and interfaces, subject to the following restrictions.

Adapters

Within an adapter, you can modify the value of predefined interface controls and create new controls. Changes to an adapter do not filter down to external systems already using the adapter, so if you add new controls to an adapter, you must manually add them to the applicable external systems.

You cannot change the gateway properties of a predefined adapter.

Interfaces

You can assign a multiplication control to multiply an inbound transaction to multiple Maximo organizations or sites.

You can change the interface table name of a predefined interface. After making the change, regenerate the interface table.

You can associate only one integration point with the outbound interfaces provided with the Maximo adapter. You cannot delete predefined outbound integration points.

You can associate additional integration points with the predefined inbound interfaces. If you do, provide an interface processing class, interface user exit class, and/or an XSL map file to map the interface XML to the XML format of the integration object associated with the integration point.

The Maximo adapter does not include any interface processing classes.
Creating an External System

After creating or modifying integration objects, integration points, and interfaces, you create an external system and associate the applicable interfaces with it. You perform these activities in the External Systems application.

### Adapters and External Systems
An external system is associated with a single adapter. The predefined external system, EXTSYS1, uses the Maximo (default) adapter. If you create a new adapter, you must create a new external system to process that adapter.

### Queues
A new external system can use the predefined JMS queues or queues that you create. Assign at least one inbound or outbound queue to the external system.

### End Points and Handlers
Select the end point and handler that Maximo will use to send transactions to the external system, and define the properties for end point and handler combination. For more information, see Chapter 9, "Router," on page 9-1.

### Interfaces
Identify and enable the interfaces that the external system will process.

For outbound interfaces:

> ▼ Decide if an interface will be generated via the Data Export feature or event based transactions (data updated via a Maximo application).

  In the former case, enter the Data Export properties. You do this via the Data Export button on the Outbound Interfaces tab.

  In the latter case, enable the event that will trigger the transaction. You do this via the Enable/Disable Integration Events option on the Select Action menu.

  ▼ Enable the interface for outbound processing.

For inbound interfaces:

> ▼ Indicate whether the interface will be processed through the sequential or continuous JMS queue.

  ▼ Enable the interface for inbound processing.

  ▼ The external system must provide an action attribute to indicate the processing that the MBO must perform. For more information, see Chapter 4, "Maximo XML and Schema," on page 4-1.

**NOTE** If you use interface tables, see Chapter 5, "Interface Tables," on page 5-1 for processing requirements.
Modifying a Predefined External System

You can modify any property of a predefined external system except the name of the adapter. You can create a new external system for an existing adapter by duplicating the predefined system, then updating the copy. Duplicating an external system copies its interfaces and their settings, and its interface controls and their values.
The Maximo XML query framework lets an external system send an XML query to Maximo and receive a synchronous XML response back. A query can be executed using RMI and SOAP request (Web service).

This chapter explains how to use the query framework to send queries to Maximo. Before reading this chapter, read Chapter 4, "Maximo XML and Schema," for an overview of the Maximo XML structure.

This chapter contains the following sections:

- Query Creation
- Header Attributes for Query Type Interfaces
- Header Attributes for Response Type Interfaces
- Content Element for Query Type Interfaces
- Content Element for Response Type Interfaces
- Web Service Queries
Query Creation

Creating a query is a two-part process that consists of defining the query and the corresponding response. Like data synchronization transactions, query and response transactions require integration objects, integration points, and interfaces. They can use existing integration objects, but they cannot use integration points and interfaces defined for data synchronization transactions. Maximo provides a sample query and response type interface, MXINVBALQInterface, that you can use as a reference when building your own interfaces.

Note the following points about the query framework:

- The query framework is intended to provide a mechanism for external systems to query Maximo for information.
- Only interfaces within the Maximo (default) adapter can perform queries.
- The integration object associated with a query type interface must be a standard, not a merged, integration object.
- A query and its corresponding response interface must use the same integration object.

To create a query, perform the following tasks:

1. Identify the MBOs that the query needs to access, and create an integration object containing those MBOs.

2. Define an inbound query (operation = Query) integration point and an outbound response (operation = Response) integration point for the integration object.

3. In the Maximo adapter, define a query interface and a response interface that use these integration points.

4. Associate the query and response type interfaces with an external system, and enable them.
Header Attributes for Query Type Interfaces

The following attributes apply to the Header element in query type interfaces (operation= Query). They provide information about the response that the query expects to receive. All are optional.

**Header Element Attributes for Queries**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
</table>
| uniqueResult | 1 = The query expects to receive a single record as a response. Maximo returns an error if the result set contains multiple records.  
0 = The result set can contain multiple records. This is the default behavior if this attribute is not included in the Header. |
| maxItems    | If the result set can contain multiple records, this attribute identifies the number of records the response type interface will return at a time. This lets you limit the number of records in the response, in case the query's result set contains a large volume of records. If this attribute is not specified, the response type interface will contain the entire result set. |
| rsStart     | Used in conjunction with maxItems to specify the first record to be returned in a response type interface.  
Example: If maxItems=10 and rsStart is not specified, the response type interface returns results 1 through 10. To receive results 11 through 20, resend the query with rsStart=11.  
If rsStart is not specified, the response type interface starts with the first record in the result set. If the number of records in the query's result set is lower than the value of rsStart, the response type interface returns no records. |

**Example**

The response to this query will return records 11 through 20 of the query result set.

```xml
<MXPOQueryInterface xmlns="http://www.mro.com/mx/integration" language="EN">
  <Header operation="Query" uniqueResult="0" maxItems="10" rsStart="11"/>
</MXPOQueryInterface>
```
Header Attributes for Response Type Interfaces

The following attributes apply to the Header element in response type interfaces (operation = Response). They provide information about the content of the response.

**Header Element Attributes for Responses**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rsStart</td>
<td>The first record in the response type interface. This value matches the rsStart value in the corresponding query type interface. If the corresponding query contains a maxItems attribute, its requests for additional records should have the following value in the rsStart attribute: rsStart + rsCount + 1. If not specified, the response type interface starts with the first record in the result set and includes the number of records specified by the rsCount attribute.</td>
</tr>
<tr>
<td>rsCount</td>
<td>The number of records in the response type interface.</td>
</tr>
<tr>
<td>rsTotal</td>
<td>The total number of records in the query result set. If the query does not specify a value for maxItems, the rsTotal value is the same as the rsCount value.</td>
</tr>
</tbody>
</table>

**Example**

This response contains records 11 through 20 of the query result set. The query result set contains 35 records.

```xml
<MXPOQueryInterface xmlns="http://www.mro.com/mx/integration" language="EN">
  <Header operation="Response" rsStart="11" rsCount="10" rsTotal="35">

  **NOTE** The XML in a response type interface looks the same as the XML in a data synchronization interface. The primary difference is that a response uses a different set of attributes for the Header element.
Content Element for Query Type Interfaces

The Content element in a query type interface contains the selection criteria for the query. Queries can select records based on a single value or a range of values.

**NOTE**
The selection criteria can apply only to the top-level MBO in the integration object. However, the response can include data from all the MBOs in the integration object.

For example, a query against the predefined purchase order integration object (MXPO) must select records based on values in the top-level MBO (PO), but the response can include data from the PO, POLINE, POCOST, and POTERM MBOs. For this integration object, you cannot build a query that selects records based on, for example, the item number in the POLINE MBO.

**Operator Attribute**

The operator attribute compares the value of a database field with one or more other values. It has the following format:

\[ \text{operator} = \text{"value"} \]

The value attribute can use the following values.

**Operator Attribute Values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>equal</td>
</tr>
<tr>
<td>!=</td>
<td>not equal</td>
</tr>
<tr>
<td>&lt;</td>
<td>less than</td>
</tr>
<tr>
<td>&lt;=</td>
<td>less than or equal</td>
</tr>
<tr>
<td>&gt;</td>
<td>greater than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>greater than or equal</td>
</tr>
</tbody>
</table>

The operator attribute applies to query type interfaces only. Maximo ignores the operator attribute if it appears in non-query type interfaces. Use the less than and greater than attributes with numeric and date fields only.
Example

To find all POs in a currency other than US dollars, format the query as follows:

```xml
<MXPOQueryInterface xmlns="http://www.mro.com/mx/integration" language="EN">
  <Header operation="Query">
    <SenderID>EXTSYS1</SenderID>
  </Header>
  <Content>
    <MXPO>
      <PO>
        <CURRENCYCODE operator ="!=">USD</CURRENCYCODE>
      </PO>
    </MXPO>
  </Content>
</MXPOQueryIface>
```

Field Selection

A field-based query compares the value in a database field with a prespecified value. The value is not case-sensitive.

Examples

The following query searches for purchase orders where VENDOR is equal to ATI and STATUS is equal to APPR.

```xml
<MXPO>
  <PO>
    < VENDOR operator ="=">ATI</VENDOR>
    < STATUS operator ="=">APPR</STATUS>
  </PO>
</MXPO>
```

The following query searches for purchase orders where VENDOR is like %ATI% and STATUS is like %APPR%.

```xml
<MXPO>
  <PO>
    < VENDOR>ATI</VENDOR>
    < STATUS>APPR</STATUS>
  </PO>
</MXPO>
```

**NOTE**  The preceding query format treats ATI and APPR as if % wildcards exist before and after the hard-coded values. You cannot further restrict the search criteria (that is, you cannot specify %API or API%).
The following queries search for inventory balances that do not have a bin number. The first one uses the operator attribute; the second one does not.

```
<MXINVBAL>
  <INVBALANCES>
    <BINNUM operator ="NULL"></BINNUM>
  </INVBALANCES>
</MXINVBAL>

<MXINVBAL>
  <INVBALANCES>
    <BINNUM >NULL</BINNUM>
  </INVBALANCES>
</MXINVBAL>
```

The following query searches for purchase orders with a status or APPR (approved) or CLOSED, by using the equivalent of an SQL IN clause.

```
<MXPO>
  <PO>
    <STATUS>APPR,CLOSED</STATUS>
  </PO>
</MXPO>
```

### Range Selection

A query can search for database records with a value that falls within a range of values. The format varies, depending on whether the selection criteria is open ended or contains an upper and lower range.

#### Examples

The following query searches for purchase orders where TOTALCOST is greater than $1000.

```
<MXPO>
  <PO>
    < TOTALCOST operator="&gt;=">1000</TOTALCOST>
  </PO>
</MXPO>
```

The following query searches for purchase orders where TOTALCOST is greater than $1000 and less than $20000. It uses two instances of the integration object, the first with the from selection criteria, and the second with the to selection criteria:

```
<Content>
  <MXPO>
    <PO>
      < TOTALCOST operator="&gt;=">1000</TOTALCOST >
    </PO>
  </MXPO>
  <MXPO>
    <PO>
      < TOTALCOST operator="&lt;=">20000</TOTALCOST >
    </PO>
  </MXPO>
</Content>
```
A query can contain a maximum of two instances of the integration object. If others exist, the query use the first two instances in the XML and ignores the others.

**Content Element for Response Type Interfaces**

The content element for response type interfaces is an integration object similar to the data synchronization (operation = Notify) interface.

**Web Service Queries**

A query can be executed using RMI and SOAP request (Web service). For RMI examples, see the following file:

```java
psdi.iface.samples.QueryClient.java
```

A successful response to a query executed via a Web service returns the query result set. If the result set is empty (it contains no records), the XML returned in the SOAP body will contain the Header element and an empty Content element.

If an error occurs, an HTTP response code of 500 is returned, along with a SOAP fault detailing the error message.

Use the following URL for the Query Web service:

```
http://<host>:<port>/meaweb/services/<interfacename>
```

where `<interfacename>` is the name of the query type interface.

**NOTE** You must expose the query type interface as a Web service. Do this via the Select Action menu in the Integration Interfaces application. For more information, see Chapter 18, "Maximo Web Services," on page 18-1.
All data synchronization (operation = Notify) and query (operation = Query) interfaces defined within the Maximo adapter can be exposed as a Web service, and these Web services can be used by external systems to query or send transactions to Maximo.

The Web services provided by Maximo are document-literal style Web services and comply with the Web services Interoperability Organization (WS-I) Basic Profile. Maximo generates XML schema files that contain the definition of each interface and a WSDL file to describe the service; and optionally registers the service in any private UDDI registry. Maximo uses Apache Axis 1.1 as the Web services engine.

NOTE
Any inbound interface defined within the Maximo adapter can be deployed as a Web service.

This chapter discusses the configuration and deployment of Web services. It is directed to the system administrator and it contains the following sections:

- Web Services Configuration
- Web Service Deployment Actions
- XML Schema Generation
- WSDL Generation
- Web Service Deployment
- UDDI Registration
- Invoking a Maximo Web Service
- Web Services Security
To configure Web services, complete the following steps:

1. Before starting the WebLogic or WebSphere server, specify the following system property in the application server JVM:

   ```
   -Daxis.EngineConfigFactory=psdi.iface.webservices.WSEngineConfigFactory
   ```

   This property ensures that the Web services information is persisted in an XML file (server-config.wsdd) and any deployed Web services are automatically made available on a system restart.

   If you use BEA WebLogic and start the server with the startWeblogic script file, specify this property in the JAVA_OPTIONS section of that file.

   If you use IBM WebSphere, use the WebSphere administrative console to specify this property in the Generic JVM arguments section. Go to:

   Process Definition > Additional properties >
   Java Virtual Machine > Generic JVM arguments

   For more information on setting this property, refer to your application server documentation.

2. If you use a WebLogic server and you installed Actuate on the same machine as WebLogic, complete the following steps to prevent a conflict between Actuate and MEA Web services:

   a. Go to the line in startWebLogic.cmd where you set the CLASSPATH variable.

      **Example**

      ```
      set CLASSPATH=%WEBLOGIC_CLASSPATH%;%POINTBASE_CLASSPATH%;%JAVA_HOME%/jre/lib/rt.jar;%WL_HOME%/server/lib/webservices.jar;%CLASSPATH%
      ```

   b. If the line ends with %CLASSPATH%, as in the example, remove %CLASSPATH% from the end of the line.

      **Example**

      ```
      set CLASSPATH=%WEBLOGIC_CLASSPATH%;%POINTBASE_CLASSPATH%;%JAVA_HOME%/jre/lib/rt.jar;%WL_HOME%/server/lib/webservices.jar
      ```

      **NOTE**  Removing %CLASSPATH% does not affect the operation of Maximo or Actuate.

3. Start the WebLogic or WebSphere server.

4. In the Integration Interfaces application, select **Web Service Administration** from the Select Action menu.
Configure the following properties in the Web Service Administration dialog box.

**Web Service Properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Required/Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integration UDDI Registry Inquiry URL</td>
<td>Optional</td>
</tr>
<tr>
<td>Integration UDDI Registry User ID</td>
<td>Optional</td>
</tr>
<tr>
<td>Integration UDDI Registry Password</td>
<td>Optional</td>
</tr>
<tr>
<td>Integration UDDI Registry Publish URL</td>
<td>Optional</td>
</tr>
<tr>
<td>Integration Web Application User ID</td>
<td>Optional</td>
</tr>
<tr>
<td>Integration Web Application Password</td>
<td>Optional</td>
</tr>
<tr>
<td>Integration Web Application URL</td>
<td>Required</td>
</tr>
</tbody>
</table>

If necessary, change the IntegrationPostSize parameter in the web.xml file of the MEA Web application.

This parameter specifies the maximum size message that can be sent to Maximo via a Web service. The default value is 5MB, and anything larger will be rejected. This limitation exists because a very large message can cause an OutOfMemoryError when an XML parser tries to parse it.

After completing these configuration activities, you can deploy a Web service for the applicable interface(s).
Web Service Deployment Actions

You deploy a Web service via the Deploy Web Service option in the Select Action menu in the Integration Interfaces application. Web service deployment is by interface; you must select the interface for which the Web service is to be deployed before you perform the deployment action.

When you deploy a Web service for an interface, the following events occur.

- If they do not already exist, schemas are generated for the interface and the integration object used by the interface.

  **NOTE**  To generate a schema file without deploying a Web service, use the Generate Schema option on the Select Action menu.

- The Web Services Description Language (WSDL) file is generated for the interface.

- The Web service is deployed for the selected interface.

- If UDDI registry properties are configured, the Web service is registered in the UDDI registry.

The following sections of this chapter describe these activities.
XML Schema Generation

The generated schema files are stored in a folder called schema, which is located under the global directory. This folder contains the following subfolders:

- Interfaces
- IntegrationObjects
- MetaData

The subfolders contain the following files.

**XML Schema Subfolders**

<table>
<thead>
<tr>
<th>Subfolder</th>
<th>File(s)</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interfaces</td>
<td>Integration interface schema files</td>
<td>The interface schema definitions. These refer to the integration object schema file via XML schema import mechanism.</td>
</tr>
<tr>
<td>Integration Objects</td>
<td>Integration object schema files</td>
<td>The integration object schema definitions. When multiple interfaces use the same integration object, a single integration object schema is used by all the interfaces.</td>
</tr>
<tr>
<td>MetaData</td>
<td>MXDataTypes.xsd</td>
<td>The common data types used across all the schemas. This file defines mapping between Maximo data types (for example, Amount, GL, YORN, and so on) and XML data types. It contains the operator attribute for all complex data types. This is a static schema file that is copied to the MetaData subfolder the first time a schema is generated for any integration object.</td>
</tr>
<tr>
<td>MetaData</td>
<td>MXIntegrationMeta.xsd</td>
<td>Common components (for example, header definitions and common base types) used across all the schemas. This is a static schema file that is copied to the MetaData subfolder the first time a schema is generated for any integration object.</td>
</tr>
</tbody>
</table>

You can view the generated integration interface schema in the View Schema dialog box in the Integration Interfaces application. You access this dialog box via the View Schema option in the Select Action menu.
WSDL Generation

You also can view the generated schema files at the following URLs.

**URLs of Generated Schema Files**

<table>
<thead>
<tr>
<th>Schema</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Types</td>
<td>http://&lt;host&gt;:&lt;port&gt;/meaweb/schema/MetaData/MXDataTypes.xsd</td>
</tr>
<tr>
<td>Meta Data</td>
<td>http://&lt;host&gt;:&lt;port&gt;/meaweb/schema/MetaData/MXIntegrationMeta.xsd</td>
</tr>
<tr>
<td>Integration Object</td>
<td>http://&lt;host&gt;:&lt;port&gt;/meaweb/schema/IntegrationObjects/IntegrationObjectName.xsd</td>
</tr>
<tr>
<td>Interface</td>
<td>http://&lt;host&gt;:&lt;port&gt;/meaweb/schema/Interfaces/InterfaceName.xsd</td>
</tr>
</tbody>
</table>

where

<host>:<port>/meaweb is the value of the Integration Web Application URL property (see "Web Services Configuration," on page 18-2).

*IntegrationObjectName* is the name of the integration object whose schema you are retrieving.

*InterfaceName* is the name of the interface whose schema you are retrieving.

**WSDL Generation**

The Web service description Language (WSDL) files are generated during the deployment of a Web service. The generated file refers to the XML Schema Interface file via schema import. The WSDL file defines a single operation, `processDocument`, and it uses the schema definitions to specify the XML structure of the input and output messages to this operation. The Web service style is *document* and the binding for the input and output messages is *literal*, as mandated by Web Services Interoperability Organization (WS-I) Basic Profile 1.0. The WSDL also specifies the URL on which to invoke the Web service.

A client program needs just the schema definitions and WSDL to generate client stubs/programs in any programming language (Java, C#, and so on) to invoke the Web service. A client program that generically invokes Maximo Web services is also provided (psdi.iface.webservices.WSCallClient).

The generated WSDL files are stored in a folder called wsdl, which is located under the global directory. You can view the generated WSDL files in the View WSDL dialog box in the Integration Interfaces application. You access this dialog box via the View WSDL option in the Select Action menu.
You also can view the generated WSDL files at the following URL:

http://<host>:<port>/meaweb/wsdl/InterfaceName

where

<host>:<port>/meaweb is the value of the Integration Web Application URL property.

InterfaceName is the name of the interface whose WSDL you are retrieving.

Web Service Deployment

You deploy a Web service for an interface in the Deploy Action dialog box. You access this dialog box via the Deploy Web Service option in the Select Action menu.

If the admin service is protected by HTTP authentication, specify the HTTP user name and password in the Integration Web Application User ID and Integration Web Application Password properties, respectively, in the Web Service Administration dialog box.

The deployed Web service will be available at the following URL:

http://<host>:port/meaweb/services/InterfaceName

where

<host>:<port>/meaweb is the value of the Integration Web Application URL property.

InterfaceName is the name of the interface for which the Web service is deployed.

Information about deployed Web services is written to an XML file called server-config.wsdd, which is stored under the global directory. If the application server is restarted, this information remains in the file and the Web services are automatically redeployed on startup.

To obtain the list of deployed (available) Web services, go to the following URL:

http://<host>:<port>/meaweb/services

where

<host>:<port>/meaweb is the value of the Integration Web Application URL property.
UDDI Registration

You can optionally register deployed Web services in a UDDI registry by configuring the following properties in the Web Service Administration dialog box:

- Integration UDDI Registry Publish URL
- Integration UDDI Registry Inquiry URL
- Integration UDDI Registry User ID
- Integration UDDI Registry Password

**NOTE**
If you use a WebLogic application server, the UDDI registry is set up and enabled out of the box. If you use a WebSphere application server, refer to the WebSphere documentation for instructions on setting up the UDDI registry.

If you specify values for the UDDI registry publish URL and inquiry URL properties, Maximo will register the Web service in the UDDI registry. To bypass UDDI registration, do not specify any values for these properties.

Only the tModel for the WSDL is registered in the UDDI registry; businessEntity, businessService, and bindingTemplate are not registered. The UDDI registration entry contains the URL to the WSDL document.
Invoking a Maximo Web Service

Maximo Web services can be invoked using two different approaches:

- Dynamic Invocation Interface (DII) using the JAX-RPC API. A client program, psdi.iface.webservices.WSCallClient, is provided.

- Generate client stub programs using a Web services tool of your choice (for example, Axis or .NET) by referencing the Maximo generated WSDL and schema files. Axis provides such a tool (WSDL2Java).

When the Web service is invoked, the response is synchronous. A successful response returns an HTTP response code of 200, with a valid SOAP response message consisting of an empty SOAP body (the SOAP envelope only). This applies to data synchronization (operation = Notify) interfaces.

By default, SOAP faults that are returned do not contain the server side stack trace, as it represents a potential security risk. However, you can change this default by changing false to true in the following parameter in the web.xml file:

```xml
<init-param>
  <param-name>axis.development.system</param-name>
  <param-value>false</param-value>
</init-param>
```

When a Web service invokes a notify type interface, the inbound processing flow is the same as that of general integration processing, except that the Web service does not process through the integration gateway or use the JMS queues. For more information, see Chapter 3, "Outbound and Inbound Processing," on page 3-1.

For information about using Web services for query type interfaces, see Chapter 17, "Using Integration Queries," on page 17-1.

Web Services Security

For information about Web services security, see Chapter 12, "Security," on page 12-1.
Maximo can save some elements, such as descriptions and long descriptions, in multiple languages. These translated columns are included in the definition of interfaces and are processed in both directions; that is, Maximo can send and receive non-base language values to and from external systems.

This chapter is directed to the system administrator. As a prerequisite, read the section of the Maximo System Administrator’s Guide that addresses multiple languages.

This chapter contains the following sections:

- Default Processing for Translatable Columns
- Outbound Multi-Language Processing
- Inbound Multi-Language Processing
- Additional Multi-Language Support
When a user logs into Maximo, he or she can choose a language code other than that of the Maximo base language. For any application in Maximo, the user can then enter language-specific values for columns that are designated as translatable.

When outbound transactions are generated, Maximo sends the applicable column values in the language associated with the logged-in user. This is true of transactions triggered via a Maximo activity or the Data Export feature.

Example

The Maximo base language is English. A user logs in as a French user, and updates an item record with a French description. The outbound message that is generated will have the item description in French, even if the description also exists in English or a third language.

Outbound Multi-Language Processing

The following columns and attributes apply to outbound multi-language transactions.

**TRANS_LANGCODE Column**

Outbound transactions contain a column called TRANS_LANGCODE, which contains the language code for translated columns within the transaction. The TRANS_LANGCODE column has been added to all predefined integration objects that have translatable columns. If you want to support translatable columns, add this field to the top-level MBO if you build new integration objects.

Many Maximo MBOs have a persistent column called LANGCODE. This column is used by the Maximo user interface for text searching column and it does not apply to the language of the transaction. By default, this column is excluded from integration objects. Do not include it in integration objects that you create.

**Language Attribute**

The root element of interfaces defined within the Maximo adapter contains a language attribute that identifies the Maximo base language, as shown in the following example:

```xml
<MXGLTXNInterface xmlns="http://www.mro.com/mx/integration" language="EN"/>
```

**Langenabled Attribute**

The XML generated by Maximo includes the langenabled attribute on every translatable column, as shown in the following example:

```xml
<DESCRIPTION langenabled="1">Item 1 description</DESCRIPTION>
```
Inbound Multi-Language Processing

The following column applies to inbound multi-language transactions.

**TRANS_LANGCODE Column**

Inbound transactions containing language-specific values for translated columns must specify that language in the TRANS_LANGCODE column. A transaction can support only one language, so translated columns in a single transaction must be in the same language.

Additional Multi-Language Support

For every Maximo database table that contains translatable columns, there is a related table called L_TABLENAME, where TABLENAME is the name of the corresponding Maximo database table. The L_TABLENAME table stores the non-base language values for every translated column, with the exception of long description. Long descriptions in both the base and non-base languages are stored in the LONGDESCRIPTION table.

If you want your XML transactions to include every translated value (in a single message), include the L_TABLENAME and LONGDESCRIPTION MBOs in the applicable integration objects.

**Example**

If the Maximo base language is English, the ITEM table contains the English description of an item, the L_ITEM table contains the French and German descriptions of the item, and the LONGDESCRIPTION table contains the English, French, and German long descriptions.
Additional Multi-Language Support
This appendix provides an overview of the predefined components provided with the Maximo adapter. It lists each interface and the following information corresponding to each interface:

- Processing direction (I = inbound, O = outbound)
- Inbound and outbound integration points
- Inbound and outbound interface processing classes
- Integration object
- Default interface table name

This appendix has two sections. The first lists application interfaces, and the second lists system interfaces.

For detailed information about each interface, see Appendix B, "Maximo Adapter Interface Specifications," on page B-1.
## Application Interfaces

<table>
<thead>
<tr>
<th>Name of Interface</th>
<th>Direction(s)</th>
<th>Inbound Integration Point</th>
<th>Outbound Integration Point</th>
<th>Inbound Processing Class</th>
<th>Outbound Processing Class</th>
<th>Integration Object</th>
<th>Default Interface Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>MXCOAInterface</td>
<td>I</td>
<td>MXCOAIN</td>
<td>None</td>
<td>psdi.iface.app.coa.MaxCOAProcess</td>
<td>Not applicable</td>
<td>MXCOA</td>
<td>MXCOA_IFACE</td>
</tr>
<tr>
<td>MXEMPACTInterface</td>
<td>I, O</td>
<td>MXEMPACTIN</td>
<td>MXEMPACTOUT</td>
<td>psdi.iface.app.empact.MaxEmpactProcess</td>
<td>Not applicable</td>
<td>MXEMPACT</td>
<td>MXEMPACT_IFACE</td>
</tr>
<tr>
<td>MXGLCOMPInterface</td>
<td>I</td>
<td>MXGLCOMPIN</td>
<td>None</td>
<td>psdi.iface.app.inventory.MaxInvBalancesProcess</td>
<td>psdi.iface.app.inventory.MoutInvBalancesProcess</td>
<td>MXGLCOMP</td>
<td>MXGLCOMP_IFACE</td>
</tr>
<tr>
<td>MXGLTXNInterface</td>
<td>O</td>
<td>None</td>
<td>MXGLTXNOUT</td>
<td>psdi.iface.app.inventory.MaxInvBalancesProcess</td>
<td>psdi.iface.app.inventory.MoutInvBalancesProcess</td>
<td>MXGLTXN</td>
<td>MXGLTXN_IFACE</td>
</tr>
<tr>
<td>MXLABORInterface</td>
<td>I, O</td>
<td>MXLABORIN</td>
<td>MXLABOROUT</td>
<td>psdi.iface.app.inventory.MaxInvBalancesProcess</td>
<td>psdi.iface.app.inventory.MoutInvBalancesProcess</td>
<td>MXLABOR</td>
<td>MXLABOR_IFACE</td>
</tr>
<tr>
<td>Name of Interface</td>
<td>Direction(s)</td>
<td>Inbound Integration Point</td>
<td>Outbound Integration Point</td>
<td>Inbound Processing Class</td>
<td>Outbound Processing Class</td>
<td>Integration Object</td>
<td>Default Interface Table</td>
</tr>
<tr>
<td>-------------------------</td>
<td>--------------</td>
<td>---------------------------</td>
<td>----------------------------</td>
<td>--------------------------</td>
<td>---------------------------</td>
<td>---------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>MXPCInterface</td>
<td>I, O</td>
<td>MXPCIN</td>
<td>MXPCOUT</td>
<td>psdi.iface.app.pc.MaxPurchaseContract</td>
<td>None</td>
<td>MXPC</td>
<td>MXPC_IFACE</td>
</tr>
<tr>
<td>MXPERSONInterface</td>
<td>I, O</td>
<td>MXPERSONIN</td>
<td>MXPERSONOUT</td>
<td>psdi.iface.app.person.MaxPersonProcess</td>
<td>None</td>
<td>MXPERSON</td>
<td>MXPERSON_IFACE</td>
</tr>
<tr>
<td>MXPOInterface</td>
<td>I, O</td>
<td>MXPOIN</td>
<td>MXPOOUT</td>
<td>psdi.iface.app.po.MaxPOProcess</td>
<td>None</td>
<td>MXPO</td>
<td>MXPO_IFACE</td>
</tr>
<tr>
<td>MXPRInterface</td>
<td>I, O</td>
<td>MXPRIN</td>
<td>MXPROUT</td>
<td>psdi.iface.app.pr.MaxPRProcess</td>
<td>None</td>
<td>MXPR</td>
<td>MXPR_IFACE</td>
</tr>
<tr>
<td>MXPROJInterface</td>
<td>I, O</td>
<td>MXPROJIN</td>
<td>MXPROJOUT</td>
<td>None</td>
<td>None</td>
<td>MXPROJ</td>
<td>MXPROJ_IFACE</td>
</tr>
<tr>
<td>MXRCVROTITMInterface</td>
<td>I, O</td>
<td>MXRCVROTITMIN</td>
<td>MXRCVROTITMOUT</td>
<td>psdi.iface.app.rcv.MaxRcvRotItmProcess</td>
<td>None</td>
<td>MXRCVROTITM</td>
<td>MXRCVROTITM_IFACE</td>
</tr>
<tr>
<td>MXSERVITEMInterface</td>
<td>I, O</td>
<td>MXSERVITEMIN</td>
<td>MXSERVITEMOUT</td>
<td>None</td>
<td>None</td>
<td>MXSERVITEM</td>
<td>MXSERVITEM_IFACE</td>
</tr>
<tr>
<td>MXVENDORInterface</td>
<td>I, O</td>
<td>MXVENDORIN</td>
<td>MXVENDOROUT</td>
<td>psdi.iface.app.company.MaxComProcess</td>
<td>None</td>
<td>MXVENDOR</td>
<td>MXVENDOR_IFACE</td>
</tr>
<tr>
<td>MXVENDORMSTRInterface</td>
<td>I, O</td>
<td>MXVENDORMSTRIN</td>
<td>MXVENDORMSTROUT</td>
<td>None</td>
<td>None</td>
<td>MXVENDORMSTR</td>
<td>MXVENDORMSTR_IFACE</td>
</tr>
<tr>
<td>MXWODETAILInterface</td>
<td>O</td>
<td>Not applicable</td>
<td>Not applicable</td>
<td>Not applicable</td>
<td>psdi.iface.app.MoutWORsvProcess</td>
<td>MXWODETAIL</td>
<td>MXWODETAIL_IFACE</td>
</tr>
<tr>
<td>MXWOInterface</td>
<td>I, O</td>
<td>MXWOIN</td>
<td>MXWOOUT</td>
<td>psdi.iface.mic.StatefulMicSetIn</td>
<td>None</td>
<td>MXWO</td>
<td>MXWO_IFACE</td>
</tr>
</tbody>
</table>
## System Interfaces

<table>
<thead>
<tr>
<th>Interface</th>
<th>Inbound (I)</th>
<th>Inbound Integration Point</th>
<th>Inbound Processing Class</th>
<th>Outbound (O)</th>
<th>Outbound Integration Point</th>
<th>Outbound Processing Class</th>
<th>Integration Object</th>
<th>Default Interface Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>MXACTIONInterface</td>
<td>I, O</td>
<td>MXACTIONIN</td>
<td>None</td>
<td>None</td>
<td>MXACTIONOUT</td>
<td>None</td>
<td>MXACTION</td>
<td>MXACTION_IFACE</td>
</tr>
<tr>
<td>MXCTEMPLATEInterface</td>
<td>I, O</td>
<td>MXCOMMTEMPLATEIN</td>
<td>None</td>
<td>None</td>
<td>MXCOMMTEMPLATEOUT</td>
<td>None</td>
<td>MXCTEMPLATE</td>
<td>MXCTEMPLATE_IFACE</td>
</tr>
<tr>
<td>MXDOMAINInterface</td>
<td>I, O</td>
<td>MXDOMAININ</td>
<td>None</td>
<td>None</td>
<td>MXDOMAINOUT</td>
<td>None</td>
<td>MXDOMAIN</td>
<td>None</td>
</tr>
<tr>
<td>MXENDPOINTInterface</td>
<td>I, O</td>
<td>MXENDPOINTIN</td>
<td>None</td>
<td>None</td>
<td>MXENDPOINTOUT</td>
<td>None</td>
<td>MXENDPOINT</td>
<td>None</td>
</tr>
<tr>
<td>MXEXTSYSInterface</td>
<td>I, O</td>
<td>MXEXTSYSIN</td>
<td>None</td>
<td>None</td>
<td>MXEXTSYSOUT</td>
<td>None</td>
<td>MXEXTSYSTEM</td>
<td>None</td>
</tr>
<tr>
<td>MXINTFACEInterface</td>
<td>I, O</td>
<td>MXINTFACEIN</td>
<td>None</td>
<td>None</td>
<td>MXINTFACEOUT</td>
<td>None</td>
<td>MXIFACE</td>
<td>None</td>
</tr>
<tr>
<td>MXINTOBJInterface</td>
<td>I, O</td>
<td>MXINTOBJIN</td>
<td>None</td>
<td>None</td>
<td>MXINTOBJOUT</td>
<td>None</td>
<td>MXINTOBJECT</td>
<td>None</td>
</tr>
<tr>
<td>MXINTTYPEInterface</td>
<td>I, O</td>
<td>MXINTTYPEIN</td>
<td>None</td>
<td>None</td>
<td>MXINTTYPEOUT</td>
<td>None</td>
<td>MXIFACETYPE</td>
<td>None</td>
</tr>
<tr>
<td>MXMAXROLEInterface</td>
<td>I, O</td>
<td>MXMAXROLEIN</td>
<td>None</td>
<td>None</td>
<td>MXMAXROLEOUT</td>
<td>None</td>
<td>MXMAXROLE</td>
<td>MXMAXROLE_IFACE</td>
</tr>
<tr>
<td>MXMESSAGEInterface</td>
<td>I, O</td>
<td>MXMESSAGEIN</td>
<td>None</td>
<td>None</td>
<td>MXMESSAGEOUT</td>
<td>None</td>
<td>MXMESSAGE</td>
<td>None</td>
</tr>
<tr>
<td>MXOBJECTTCFGInterface</td>
<td>I, O</td>
<td>MXOBJECTTCFGIN</td>
<td>psdi.iface.app.configure.MaxObjcfgProcess</td>
<td>None</td>
<td>MXOBJECTTCFGOUT</td>
<td>None</td>
<td>MXOBJECTTCFG</td>
<td>None</td>
</tr>
<tr>
<td>MXWFInterface</td>
<td>I, O</td>
<td>MXWFIN</td>
<td>psdi.iface.app.wf.MaxWFProcess</td>
<td>None</td>
<td>MXWFOUT</td>
<td>None</td>
<td>MXWF</td>
<td>None</td>
</tr>
</tbody>
</table>
This appendix provides details about the predefined interfaces provided with the Maximo adapter.

It contains the following sections:

- Master Data Interfaces
- Item Interfaces
- Document Interfaces
- Transaction Interfaces
- System Interfaces
Master Data Interfaces

This section describes the following types of interfaces:

- Asset
- Chart of Accounts
- Craft
- General Ledger Components
- Labor
- Operating Locations
- Person
- Financial Projects
- Storeroom Locations
- Vendor
- Vendor Master
**MXASSETInterface—Asset Interface**

<table>
<thead>
<tr>
<th>Interface</th>
<th>MXASSETInterface</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Detail Description</strong></td>
<td>Asset Interface</td>
</tr>
<tr>
<td></td>
<td>This interface allows for bidirectional synchronization of asset information, including meters, between Maximo and any external system.</td>
</tr>
<tr>
<td></td>
<td>This interface synchronizes individual assets, but not the asset hierarchy as a whole. It supports the addition and update of meters tied to an asset, but not the update of meter reading values.</td>
</tr>
<tr>
<td></td>
<td>Inbound processing of ASSETMETER is restricted to attributes that define the meter (not meter reading values or other information pertaining to meter readings).</td>
</tr>
</tbody>
</table>

**Outbound Implementation Details**

<table>
<thead>
<tr>
<th>Integration Point Processing</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class Functionality</td>
<td>None</td>
</tr>
<tr>
<td>Interface Processing Rules</td>
<td>None</td>
</tr>
<tr>
<td>Interface Controls</td>
<td>None</td>
</tr>
</tbody>
</table>

**Inbound Implementation Details**

<table>
<thead>
<tr>
<th>Integration Point Processing</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class Functionality</td>
<td>None</td>
</tr>
<tr>
<td>Interface Processing Rules</td>
<td>None</td>
</tr>
<tr>
<td>Interface Controls</td>
<td>None</td>
</tr>
</tbody>
</table>

**Comments**

- Prerequisite: Synchronize applicable operating locations, storerooms, meters, and items before using this interface.
- The MBO relationship used to retrieve the ASSETMETER MBO is different from the one used in the Assets application. Therefore, the changed attribute in the Maximo XML is not set for any values from this MBO.
<table>
<thead>
<tr>
<th>Interface</th>
<th>MXCOAInterface</th>
</tr>
</thead>
</table>

**Detail Description**  
This interface allows for inbound synchronization of chart of accounts data.

**Outbound Implementation Details**

<table>
<thead>
<tr>
<th>Integration Point Processing Class Functionality</th>
<th>Not applicable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface Processing Rules</td>
<td>Not applicable</td>
</tr>
<tr>
<td>Interface Controls</td>
<td>Not applicable</td>
</tr>
</tbody>
</table>

**Inbound Implementation Details**

<table>
<thead>
<tr>
<th>Integration Point Processing Class Functionality</th>
<th>Processes accounts that are specified in the following ways:</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ As a combination of delimited segments</td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>&lt;GLACCOUNT&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;VALUE&gt;6400-2-10&lt;/VALUE&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/GLACCOUNT&gt;</td>
</tr>
<tr>
<td></td>
<td>The interface validates each segment, then creates the COA record in Maximo with:</td>
</tr>
<tr>
<td></td>
<td>GLACCOUNT=6400-2-10</td>
</tr>
<tr>
<td></td>
<td>GLCOMP01=6400</td>
</tr>
<tr>
<td></td>
<td>GLCOMP02=2</td>
</tr>
<tr>
<td></td>
<td>GLCOMP03=10</td>
</tr>
<tr>
<td></td>
<td>This option is available for XML and interface tables. You can specify GL Account this way in all interfaces.</td>
</tr>
<tr>
<td>▼ As individual components that are part of the GLACCOUNT datatype definition</td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>&lt;GLACCOUNT&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;GLCOMP glorder=&quot;0&quot;&gt;6400&lt;/GLCOMP&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;GLCOMP glorder=&quot;1&quot;&gt;2&lt;/GLCOMP&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;GLCOMP glorder=&quot;2&quot;&gt;10&lt;/GLCOMP&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/GLACCOUNT&gt;</td>
</tr>
<tr>
<td></td>
<td>The interface validates the components and creates the account, using the delimiter defined in the GLCONFIGURE table for each segment.</td>
</tr>
<tr>
<td></td>
<td>This option is available for XML only.</td>
</tr>
<tr>
<td>Interface (Continued)</td>
<td>MXCOAIInterface</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>▼ As individual components that are part of the CHARTOFACCOUNTS MBO definition</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>&lt;CHARTOFACCOUNTS&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;GLCOMP01&gt;6400&lt;/GLCOMP01&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;GLCOMP02&gt;2&lt;/GLCOMP01&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;GLCOMP03&gt;10&lt;/GLCOMP01&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;/CHARTOFACCOUNTS&gt;</td>
<td></td>
</tr>
<tr>
<td>This option is available for interface tables and XML.</td>
<td></td>
</tr>
<tr>
<td>The first two options apply to all GLACCOUNT type fields across all interfaces. A GL account can be specified using either option, in all interfaces.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Interface Processing Rules</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comments</td>
<td>Prerequisite: Use the GL Component interface to create components before using this interface.</td>
</tr>
</tbody>
</table>
### MXCRAFTInterface—Craft Interface

<table>
<thead>
<tr>
<th>Interface</th>
<th>MXCRAFTInterface</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detail Description</td>
<td>Craft Interface</td>
</tr>
<tr>
<td></td>
<td>This interface allows for bidirectional synchronization of craft information, including rates and skill levels, between Maximo and any external system.</td>
</tr>
</tbody>
</table>

#### Outbound Implementation Details

<table>
<thead>
<tr>
<th>Integration Point Processing</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class Functionality</td>
<td>None</td>
</tr>
<tr>
<td>Interface Processing Rules</td>
<td>None</td>
</tr>
<tr>
<td>Interface Controls</td>
<td>None</td>
</tr>
</tbody>
</table>

#### Inbound Implementation Details

<table>
<thead>
<tr>
<th>Integration Point Processing</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class Functionality</td>
<td>None</td>
</tr>
<tr>
<td>Interface Processing Rules</td>
<td>None</td>
</tr>
<tr>
<td>Interface Controls</td>
<td>None</td>
</tr>
</tbody>
</table>

#### Comments

Prerequisite: If the craft references a skill or contracts, that information must exist in Maximo.

A craft can have a standard rate; a rate for each skill-level for the craft; or different rates for each contract that provides the craft, with an optional skill-level rate for each contract.

The CRAFT and CRAFTSKILL records have a STANDARDRATE field for the hourly rate for each craft or skill associated with the craft.

The CRAFTRATE record specifies rates for vendors that supply the craft (and, optionally, different rates for each skill-craft-vendor combination). It does not include standard rates for the craft or skills associated with the craft. An inbound CRAFTRATE record must include a vendor.
**MXGLCOMPIface—General Ledger Components Interface**

<table>
<thead>
<tr>
<th>Interface</th>
<th>MXGLCOMPIface</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detail Description</td>
<td>General Ledger Components interface</td>
</tr>
<tr>
<td></td>
<td>This interface allows for inbound synchronization of general ledger component data.</td>
</tr>
</tbody>
</table>

**Outbound Implementation Details**

| Integration Point Processing Class Functionality | Not applicable |
| Interface Processing Rules                  | Not applicable |
| Interface Controls                          | Not applicable |

**Inbound Implementation Details**

| Integration Point Processing Class Functionality | None |
| Interface Processing Rules                  | None |
| Interface Controls                          | None |
| Comments                                    | None |
**MXLABORInterface—Labor Interface**

<table>
<thead>
<tr>
<th>Interface</th>
<th>MXLABORInterface</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Detail Description</strong></td>
<td>Labor Interface</td>
</tr>
<tr>
<td></td>
<td>This interface allows for bidirectional synchronization of labor information between Maximo and any external system.</td>
</tr>
<tr>
<td></td>
<td>If a person record does not exist, the inbound processing automatically creates one.</td>
</tr>
</tbody>
</table>

**Outbound Implementation Details**

| Integration Point Processing Class Functionality | None |
| Interface Processing Rules                     | None |
| Interface Controls                              | None |

**Inbound Implementation Details**

| Integration Point Processing Class Functionality | Sets the PERSONID on the PERSON record, based on the PERSONID in the LABOR record |
|                                                | No business rules |
| Interface Processing Rules                     | None |
| Interface Controls                              | None |
| Comments                                        | Prerequisite: Synchronize any associated craft, work location, or storeroom location before adding or updating the labor |
|                                                | There can be only one Person record for a Labor record. |
**MXOPERLOCInterface—Operating Locations Interface**

<table>
<thead>
<tr>
<th>Interface</th>
<th>MXOPERLOCInterface</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Detail Description</strong></td>
<td>Operating Locations Interface</td>
</tr>
<tr>
<td></td>
<td>This interface allows for bidirectional synchronization of operating location information including meters, between Maximo and any external system.</td>
</tr>
<tr>
<td></td>
<td>This interface synchronize individual operating locations, but not the location hierarchy as a whole. It supports the addition and update of meters tied to an operating location, but not the update of meter reading values.</td>
</tr>
<tr>
<td></td>
<td>Inbound processing of LOCATIONMETER is restricted to attributes that define the meter, but not meter reading values or other information pertaining to meter readings.</td>
</tr>
<tr>
<td><strong>Outbound Implementation Details</strong></td>
<td></td>
</tr>
<tr>
<td>Integration Point Processing</td>
<td>Skips all outbound LOCATION records that are not of type OPERATING or its synonyms.</td>
</tr>
<tr>
<td>Class Functionality</td>
<td></td>
</tr>
<tr>
<td>Interface Processing Rules</td>
<td>None</td>
</tr>
<tr>
<td>Interface Controls</td>
<td>None</td>
</tr>
<tr>
<td><strong>Inbound Implementation Details</strong></td>
<td></td>
</tr>
<tr>
<td>Integration Point Processing</td>
<td>If no LOCTYPE is specified, sets value as default value of the OPERATING domain.</td>
</tr>
<tr>
<td>Class Functionality</td>
<td></td>
</tr>
<tr>
<td>Interface Processing Rules</td>
<td>None</td>
</tr>
<tr>
<td>Interface Controls</td>
<td>None</td>
</tr>
<tr>
<td>Comments</td>
<td>Prerequisite: If applicable, synchronize meters before using this interface.</td>
</tr>
</tbody>
</table>
### MXPERSONInterface—Person Interface

<table>
<thead>
<tr>
<th>Interface</th>
<th>MXPERSONInterface</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detail Description</td>
<td>Person Interface</td>
</tr>
<tr>
<td></td>
<td>This interface allows for bidirectional synchronization of person information, including phone and e-mail details, between Maximo and any external system.</td>
</tr>
<tr>
<td></td>
<td>Note: When updating a person record, the inbound processing always deletes then recreates the phone and e-mail information, based on the information provided in the inbound interface.</td>
</tr>
</tbody>
</table>

### Outbound Implementation Details

<table>
<thead>
<tr>
<th>Integration Point Processing</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class Functionality</td>
<td>None</td>
</tr>
<tr>
<td>Interface Processing Rules</td>
<td>None</td>
</tr>
<tr>
<td>Interface Controls</td>
<td>None</td>
</tr>
</tbody>
</table>

### Inbound Implementation Details

<table>
<thead>
<tr>
<th>Integration Point Processing</th>
<th>Deletes, then re-inserts, phone and e-mail information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class Functionality</td>
<td>Handles status changes</td>
</tr>
<tr>
<td></td>
<td>No business rules</td>
</tr>
<tr>
<td>Interface Processing Rules</td>
<td>None</td>
</tr>
<tr>
<td>Interface Controls</td>
<td>None</td>
</tr>
<tr>
<td>Comments</td>
<td>None</td>
</tr>
</tbody>
</table>
### MXPROJInterface—Financial Projects Interface

<table>
<thead>
<tr>
<th>Interface</th>
<th>MXPROJInterface</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Detail Description</strong></td>
<td>Financial Projects interface</td>
</tr>
<tr>
<td></td>
<td>This interface provides for bidirectional synchronization of financial project and task information between Maximo and any external system.</td>
</tr>
<tr>
<td></td>
<td>This interface synchronizes individual tasks and projects as separate messages. It does not synchronize a project and all its child tasks in a single message.</td>
</tr>
</tbody>
</table>

#### Outbound Implementation Details

| Integration Point Processing Class Functionality | None |
| Interface Processing Rules                      | None |
| Interface Controls                               | None |

#### Inbound Implementation Details

| Integration Point Processing Class Functionality | None |
| Interface Processing Rules                      | None |
| Interface Controls                               | None |
| Comments                                         | None |
# MXSTORELOCInterface—Storeroom Location Interface

<table>
<thead>
<tr>
<th>Interface</th>
<th>MXSTORELOCInterface</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Storeroom Location Interface</td>
</tr>
<tr>
<td></td>
<td>This interface allows bidirectional synchronization of storeroom, labor, and courier location between Maximo and any external system.</td>
</tr>
</tbody>
</table>

### Outbound Implementation Details

| Integration Point Processing Class Functionality | Filters out all locations except Storeroom, Labor, Courier, and any synonyms for these. |
| Interface Processing Rules                     | None |
| Interface Controls                              | None |

### Inbound Implementation Details

| Integration Point Processing Class Functionality | If no location type is specified, sets LOCTYPE to the default value for Storeroom. |
| Interface Processing Rules                     | None |
| Interface Controls                              | None |
| Comments                                       | None |
**MXVENDORInterface—Vendor Interface**

<table>
<thead>
<tr>
<th>Interface</th>
<th>MXVENDORInterface</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Detail Description</strong></td>
<td>This interface allows for bidirectional synchronization of organization level vendor data between Maximo and any external system. Maximo optionally creates the company master record, if necessary. (See Comments.)</td>
</tr>
</tbody>
</table>

**Outbound Implementation Details**

| Integration Point Processing Class Functionality | None |
| Interface Processing Rules | None |
| Interface Controls | None |

**Inbound Implementation Details**

| Integration Point Processing Class Functionality | Creates a company master record in the corresponding organization, if necessary. |
| Interface Processing Rules | None |
| Interface Controls | None |
| Comments | Prerequisite: To enable this interface to create the company master record, select the Automatically Add Companies to Company Master flag for the set associated with the organization where the vendor record is being added. |
### MXVENDORMSTRInterface—Vendor Master Interface

<table>
<thead>
<tr>
<th>Interface</th>
<th>MXVENDORMSTRInterface</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detail Description</td>
<td>Vendor Master interface</td>
</tr>
<tr>
<td></td>
<td>This interface allows for bidirectional synchronization of set level vendor master data between Maximo and any external system.</td>
</tr>
<tr>
<td></td>
<td>This interface does not support synchronization of master contacts.</td>
</tr>
</tbody>
</table>

#### Outbound Implementation Details

| Integration Point Processing Class Functionality | None |
| Interface Processing Rules | None |
| Interface Controls | None |

#### Inbound Implementation Details

| Integration Point Processing Class Functionality | None |
| Interface Processing Rules | None |
| Interface Controls | None |
| Comments | None |
Item Interfaces

This section describes the following types of interfaces:

- Item
- Service item
- Tool item
- Inventory
- Inventory balance
- Item vendor
- Inventory reservations
- Issues
### MXITEMInterface—Item Interface

<table>
<thead>
<tr>
<th>Interface</th>
<th>MXITEMInterface</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detail Description</td>
<td>Material item interface</td>
</tr>
<tr>
<td></td>
<td>This interface provides for the bidirectional synchronization of material item information between Maximo and any external system. It does not process service or tool type items.</td>
</tr>
</tbody>
</table>

#### Outbound Implementation Details

<table>
<thead>
<tr>
<th>Integration Point Processing Class Functionality</th>
<th>Filters out service and tool type items.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface Processing Rules</td>
<td>None</td>
</tr>
<tr>
<td>Interface Controls</td>
<td>None</td>
</tr>
</tbody>
</table>

#### Inbound Implementation Details

<table>
<thead>
<tr>
<th>Integration Point Processing Class Functionality</th>
<th>If the inbound transaction is an item update and the capitalized flag of the item has changed, calls a Maximo method to change the capitalized status of the item.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>If the item type is not ITEM or a valid synonym, reports an error.</td>
</tr>
<tr>
<td>Interface Processing Rules</td>
<td>None</td>
</tr>
<tr>
<td>Interface Controls</td>
<td>None</td>
</tr>
<tr>
<td>Comments</td>
<td>None</td>
</tr>
</tbody>
</table>
### MXSERVITEMInterface—Service Item Interface

<table>
<thead>
<tr>
<th>Interface</th>
<th>MXSERVITEMInterface</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Detail Description</strong></td>
<td>Service Item Interface</td>
</tr>
<tr>
<td></td>
<td>This interface provides for the bidirectional synchronization of service item information between Maximo and any external system. It does not process material or tool type items.</td>
</tr>
</tbody>
</table>

#### Outbound Implementation Details

<table>
<thead>
<tr>
<th>Integration Point Processing Class Functionality</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface Processing Rules</td>
<td>None</td>
</tr>
<tr>
<td>Interface Controls</td>
<td>None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Integration Point Processing Class Functionality</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface Processing Rules</td>
<td>None</td>
</tr>
<tr>
<td>Interface Controls</td>
<td>None</td>
</tr>
</tbody>
</table>

| Comments                                         | None |

#### Inbound Implementation Details

<table>
<thead>
<tr>
<th>Integration Point Processing Class Functionality</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface Processing Rules</td>
<td>None</td>
</tr>
<tr>
<td>Interface Controls</td>
<td>None</td>
</tr>
</tbody>
</table>

| Comments                                         | None |
## MXTOOLITEMInterface—Tool Item Interface

<table>
<thead>
<tr>
<th>Interface</th>
<th>MXTOOLITEMInterface</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Detail Description</strong></td>
<td>Tool Item Interface</td>
</tr>
<tr>
<td></td>
<td>This interface provides for the bidirectional synchronization of tool item information between Maximo and any external system. It does not process material or service type items.</td>
</tr>
<tr>
<td><strong>Outbound Implementation Details</strong></td>
<td></td>
</tr>
<tr>
<td>Integration Point Processing</td>
<td>None</td>
</tr>
<tr>
<td>Class Functionality</td>
<td></td>
</tr>
<tr>
<td>Interface Processing Rules</td>
<td>None</td>
</tr>
<tr>
<td>Interface Controls</td>
<td>None</td>
</tr>
<tr>
<td><strong>Inbound Implementation Details</strong></td>
<td></td>
</tr>
<tr>
<td>Integration Point Processing</td>
<td>If the inbound transaction is an item update and the capitalized flag of the item has changed, calls a Maximo method to change the capitalized status of the item.</td>
</tr>
<tr>
<td>Class Functionality</td>
<td></td>
</tr>
<tr>
<td>Interface Processing Rules</td>
<td>None</td>
</tr>
<tr>
<td>Interface Controls</td>
<td>None</td>
</tr>
<tr>
<td>Comments</td>
<td>Tool items cannot be condition-enabled. They behave like material items in all other respects.</td>
</tr>
</tbody>
</table>
### MXINVENTORYInterface—Inventory Interface

<table>
<thead>
<tr>
<th>Interface</th>
<th>MXINVENTORYInterface</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Detail Description</strong></td>
<td>Inventory Interface</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>This interface provides for the bidirectional synchronization of inventory (item-storeroom) definitions, reorder details, and costing information between Maximo and any external system.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>In the outbound direction, this interface also provides balance information at the storeroom level.</td>
<td></td>
</tr>
<tr>
<td><strong>Outbound Implementation Details</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Integration Point Processing Class Functionality</strong></td>
<td>Sets the value of ITEMTYPE from the Item table.</td>
</tr>
<tr>
<td><strong>Interface Processing Rules</strong></td>
<td>Skips the record if its ITEMTYPE is in the SKIPITEMTYPE interface control.</td>
</tr>
<tr>
<td><strong>Interface Controls</strong></td>
<td>The SKIPITEMTYPE control specifies the item type of inventory records to be skipped.</td>
</tr>
<tr>
<td></td>
<td>Note: This control applies to the inventory (MXINVENTORYInterface) and inventory balance (MXINVBALInterface) interfaces, so using it causes both interfaces to skip transactions for the specified item type(s). It is not possible for only one of those interfaces to skip the item type.</td>
</tr>
<tr>
<td><strong>Inbound Implementation Details</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Integration Point Processing Class Functionality</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Interface Processing Rules</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Interface Controls</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Comments</strong></td>
<td>The user-defined ITEMTYPE field identifies whether the item is a material, service, or tool type item. By default, this interface processes material and tool type items.</td>
</tr>
</tbody>
</table>
### MXINVBALInterface—Inventory Balance Interface

<table>
<thead>
<tr>
<th>Interface</th>
<th>MXINVBALInterface</th>
</tr>
</thead>
</table>
| **Detail Description** | Inventory balance interface  
This interface provides for the bidirectional synchronization of inventory balances between Maximo and any external system.  
The balance change information in this interface is for the lowest level (BIN or LOT) within Maximo. |
| **Outbound Implementation Details** | |
| **Integration Point Processing Class Functionality** | None |
| **Interface Processing Rules** | Skips the record if its ITEMTYPE is in the SKIPITEMTYPE interface control. |
| **Interface Controls** | The SKIPITEMTYPE control specifies the item type of inventory records to be skipped.  
Note: This control applies to the inventory (MXINVENTORYInterface) and inventory balance (MXINVBALInterface) interfaces, so using it causes both interfaces to skip transactions for the specified item type(s). It is not possible for only one of those interfaces to skip the item type. |
| **Inbound Implementation Details** | |
| **Integration Point Processing Class Functionality** | The inbound inventory balance interface internally calls the current balance adjustment method provided by Maximo, and this creates a CURBALADJ financial transaction in INVTRANS. This is not a valid adjustment transaction, and should be ignored for financial reporting. |
| **Interface Processing Rules** | None |
| **Interface Controls** | None |
| **Comments** | The user-defined ITEMTYPE field identifies whether the item is a material, service, or tool type item. By default, this interface processes material and tool type items. |
## MXINNVENDORInterface—Item-Vendor Interface

<table>
<thead>
<tr>
<th>Interface</th>
<th>MXINNVENDORInterface</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Detail Description</strong></td>
<td>Item Vendor Interface&lt;br&gt;This interface provides for the bidirectional synchronization of item-supplier information between Maximo and any external system. Maximo maintains this information at the organization level, and provides an option to specify site-level suppliers for an item.</td>
</tr>
<tr>
<td><strong>Outbound Implementation Details</strong></td>
<td></td>
</tr>
<tr>
<td>Integration Point Processing</td>
<td>Sets the value of COMPANIES.CURRENCYCODE to the user-defined CURRENCYCODE field in the outbound message.</td>
</tr>
<tr>
<td>Class Functionality</td>
<td></td>
</tr>
<tr>
<td>Interface Processing Rules</td>
<td>Does not send out the record if its ITEMTYPE is in the ITEMTYPEFORINV interface control.</td>
</tr>
<tr>
<td>Interface Controls</td>
<td>The ITEMTYPEFORINV control specifies the item type of inventory records to be skipped. Note: This control should contain the same value(s) as the SKIPITEMTYPE control used by the Inventory (MXINVENTORYInterface) and Inventory Balance (MXINVBALInterface) interfaces.</td>
</tr>
<tr>
<td><strong>Inbound Implementation Details</strong></td>
<td></td>
</tr>
<tr>
<td>Integration Point Processing</td>
<td>If the inbound message contains a currency code and that code differs from the vendor’s currency code, reports an error.</td>
</tr>
<tr>
<td>Class Functionality</td>
<td></td>
</tr>
<tr>
<td>Interface Processing Rules</td>
<td>None</td>
</tr>
<tr>
<td>Interface Controls</td>
<td>None</td>
</tr>
<tr>
<td>Comments</td>
<td>The ITEMTYPE field identifies whether the item is a material, service, or tool type item. By default, this interface processes all three types of items.</td>
</tr>
</tbody>
</table>
## MXINVRESInterface—Inventory Reservations Interface

<table>
<thead>
<tr>
<th>Interface</th>
<th>MXINVRESInterface</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Detail Description</strong></td>
<td>Inventory Reservations Interface&lt;br&gt;This interface provides for the bidirectional synchronization of inventory (storeroom) reservations between Maximo and any external system. This interface does not process direct issue reservation records created by Maximo.</td>
</tr>
<tr>
<td><strong>Outbound Implementation Details</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Integration Point Processing Class Functionality</strong></td>
<td>Filters out reservations created for direct issue items.</td>
</tr>
<tr>
<td><strong>Interface Processing Rules</strong></td>
<td>None&lt;br&gt;Note: Unlike several other item interfaces, this one does not use the SKIPITEMTYPE control to skip specific types of items. Therefore, reservations for tool items can be sent to external systems.</td>
</tr>
<tr>
<td><strong>Interface Controls</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Inbound Implementation Details</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Integration Point Processing Class Functionality</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Interface Processing Rules</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Interface Controls</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Comments</strong></td>
<td>The user-defined ITEMTYPE field identifies whether the item is a material, service, or tool type item. By default, this interface processes material and tool type items.</td>
</tr>
</tbody>
</table>
# MXINVISSUEInterface—Issues Interface

<table>
<thead>
<tr>
<th>Interface</th>
<th>MXINVISSUEInterface</th>
</tr>
</thead>
</table>
| **Detail Description** | Issues Interface  
This interface provides for the bidirectional synchronization of issues and returns information between Maximo and any external system. |
| **Outbound Implementation Details** |  |
| **Integration Point Processing Class Functionality** | Filters out direct issues from MATUSETRANS (PONUM is not null and ISSUETYPE is ISSUE or RETURN) because they are handled by the receipts (MXRECEIPTInterface) interface.  
Filters out variance transactions that are written to MATUSETRANS by the invoice approval process. |
| **Interface Processing Rules** | None  
Note: Unlike several other item interfaces, this one does not use the SKIPITEMTYPE control to skip specific types of items. Therefore, issues for tool items can be sent to external systems. |
| **Interface Controls** | None |
| **Inbound Implementation Details** |  |
| **Integration Point Processing Class Functionality** | Validates ISSUETYPE to ensure that it is ISSUE, RETURN, or null. If it is any other value, reports an error. If it is null, defaults the value to ISSUE. |
| **Interface Processing Rules** | None |
| **Interface Controls** | None |
| **Comments** | The user-defined ITEMTYPE field identifies whether the item is a material, service, or tool type item. By default, this interface processes material and tool type items. |
Document Interfaces

This section describes the following types of interfaces:

- Purchase contract (PC)
- Purchase order (PO)
- Purchase requisition (PR)
- Invoice
- Work order (WO)
- Work order detail

The STATUSIFACE Field

All purchasing and work order interfaces have a status, so it is necessary to identify whether related transactions contain new or updated records, or status changes only. The STATUSIFACE field provides this information.

In general, this field applies to all integration object sub-records that are created from stateful MBOs, which are MBOs that have a STATUS field and support status change actions. All the document interfaces have a stateful primary MBO.

By default, the STATUSIFACE field has been added only to the purchasing document interfaces (PR, PO, PC, and invoice), but you can add it to work order interfaces or any other interface that uses an integration object with a stateful primary MBO.

The STATUSIFACE Field and Outbound Processing

While processing an integration object sub-record that has been created from a stateful MBO, the Maximo generic outbound integration point processing looks for a STATUS field in the MBO and a STATUSIFACE field in the corresponding integration object sub-record. If both fields exist, the processing sets the value of the STATUSIFACE field to the value of the changed attribute for the corresponding STATUS field (0 indicates not changed; 1 indicates changed).

This processing applies only to event-generated outbound interfaces, not to interfaces exported via the Data Export feature.

**NOTE** If an outbound processing class does not exist for an interface, the preceding generic processing still applies.
The STATUSIFACE Field and Inbound Processing

When processing an integration object with a stateful primary MBO, the generic inbound processing looks for a STATUSIFACE field on the corresponding integration object sub-record to determine if the inbound message is to be processed as both a document update and a status change, or as a status change only, as follows:

<table>
<thead>
<tr>
<th>Value of STATUSIFACE</th>
<th>Document exists in Maximo</th>
<th>Document does not exist in Maximo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not provided</td>
<td>▼ Updates the document in Maximo</td>
<td>▼ Adds the document to Maximo</td>
</tr>
<tr>
<td>0</td>
<td>▼ If the status of the inbound document is different from the status in Maximo, updates the status in Maximo</td>
<td>▼ Sets the status in Maximo to the status of the inbound message</td>
</tr>
<tr>
<td>1</td>
<td>▼ Updates the status in Maximo</td>
<td>Error</td>
</tr>
</tbody>
</table>

**NOTE** The generic processing described in the preceding table applies only to inbound interfaces using integration objects with a stateful primary MBO. It does not apply to any stateful MBO that is included as a child in the integration object.

The inbound integration point processing class for purchase requisitions and purchase orders overrides the generic processing described here. For more information, see the detailed description of the PR and PO interfaces later in this chapter.
Outbound Processing Rules for Purchasing Interfaces

This section applies to the following types of interfaces:

- Purchase contract
- Purchase order
- Purchase requisition
- Invoice

Purchasing document interfaces are sent out only when the status of the document changes. The processing rules implement the outbound processing described in this section. Users can disable these rules to change the behavior of outbound processing to meet their requirements.

The SEND control associated with each purchasing interface specifies the status(es) at which Maximo sends out the entire document. The interfaces and their corresponding controls are:

<table>
<thead>
<tr>
<th>Interface</th>
<th>SEND Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purchase contract</td>
<td>PCSEND</td>
</tr>
<tr>
<td>Purchase order</td>
<td>POSEND</td>
</tr>
<tr>
<td>Purchase requisition</td>
<td>PRSEND</td>
</tr>
<tr>
<td>Invoice</td>
<td>IVSEND</td>
</tr>
</tbody>
</table>

**NOTE**

This approach allows users to configure when they want the entire document, and to use the STATUSIFACE field to identify if the outbound message is only a status change notification or the entire document.

Any status that you specify in the SEND control is the value, not the MAXVALUE. If multiple synonym values exist for a status, list all applicable synonyms.

The following table describes the conditions for sending outbound purchase documents, and the content of the outbound message.

**NOTE**

All conditions must be met.

<table>
<thead>
<tr>
<th>Conditions</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ The status has changed.</td>
<td>No output.</td>
</tr>
<tr>
<td>▼ The status is not listed in the SEND control.</td>
<td></td>
</tr>
<tr>
<td>▼ The document has not been sent out earlier.</td>
<td></td>
</tr>
<tr>
<td>▼ The status has changed.</td>
<td>The entire document.</td>
</tr>
<tr>
<td>▼ The status is listed in the SEND control.</td>
<td>The STATUSIFACE value is changed to 0.</td>
</tr>
<tr>
<td>▼ The status has changed.</td>
<td>The header only.</td>
</tr>
<tr>
<td>▼ The status is not listed in the SEND control.</td>
<td>The STATUSIFACE value is not changed.</td>
</tr>
<tr>
<td>▼ The document has been sent out earlier.</td>
<td></td>
</tr>
</tbody>
</table>
The following processing rules perform the processing described above. The asterisk (*) in the first two rule names represents the acronym for the interface type (PC, PO, PR, or IV).

<table>
<thead>
<tr>
<th>Rule</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>SKIP*</td>
<td>Skips the invoice if both of the following conditions are true:</td>
</tr>
<tr>
<td></td>
<td>- The document has not been sent out before.</td>
</tr>
<tr>
<td></td>
<td>- The new status is not listed in the SEND control.</td>
</tr>
<tr>
<td>SKIP*UPDATE</td>
<td>Skips the invoice if the status has not changed.</td>
</tr>
<tr>
<td>SETSTATUSIFACE</td>
<td>Sets the value of the STATUSIFACE field to false for all statuses other than the ones in the SEND control.</td>
</tr>
<tr>
<td>SKIPCHILDREN</td>
<td>Sends out the header data only (that is, drops all child sub-records) if the new status is not in the SEND control.</td>
</tr>
</tbody>
</table>
## MXPCInterface—Purchase Contract Interface

<table>
<thead>
<tr>
<th>Interface</th>
<th>MXPCInterface</th>
</tr>
</thead>
</table>
| **Detail Description** | Purchase Contract interface  
This interface allows for bidirectional synchronization of purchase contract information between Maximo and any external system.  
Maximo sends out the entire contract when the contract status changes to APPR. For other statuses, Maximo sends a status change notification.  
Note: This interface currently supports blanket and price type contracts only. |

### Outbound Implementation Details

| Integration Point Processing Class Functionality | None  
For information about the generic outbound processing of the STATUSIFACE field, see ”The STATUSIFACE Field and Outbound Processing,” on page B-24. |
| Interface Processing Rules | See ”Outbound Processing Rules for Purchasing Interfaces,” on page B-26. |
| Interface Controls | The PCSEND control specifies the status values at which Maximo sends out the entire contract. It can have multiple values. The predefined value is APPR. |

### Inbound Implementation Details

| Integration Point Processing Class Functionality | For an overview of the generic inbound processing functionality, see ”The STATUSIFACE Field and Inbound Processing,” on page B-25.  
The purchase contract integration point processing class overrides the generic functionality as follows:  
If the contract revision number exists in Maximo, the following processing takes place:  
> If the message contains an Add action, an error occurs.  
> If the status of the contract in Maximo is APPR, the processing does not update the header, terms, or CONTRACTAUTH. It updates existing lines and adds any lines in the message that do not already exist in Maximo.  
> If the status of an existing line in Maximo is APPR, the processing attempts to unapprove the line (change the status to WAPPR), update the line, then change the status to the status in the inbound message. |

---

Document Interfaces

IBM Maximo Enterprise Adapter: System Administrator's Guide
If the status of the contract in Maximo is not DRAFT, WAPPR, or PNDREV, the processing class skips the processing of CONTRACTAUTH and CONTRACTTERM. If the inbound header or lines have a different status than the Maximo header and lines, the processing class tries to update the Maximo status, first on the header, then on the lines.

If the contract revision number does not exist in Maximo, the following processing takes place:

If the message contains a Change action, an error occurs.

The processing tries to find the current active revision for the contract (a revision with STATUS = synonym of APPR).

If it finds that revision, it attempts to create a new revision with the given revision number, update the header/lines on this revision with data from the inbound message, and change the status on the header/lines as per the statuses in the inbound message.

<table>
<thead>
<tr>
<th>Interface Processing Rules</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface Controls</td>
<td>None</td>
</tr>
<tr>
<td>Comments</td>
<td>This interface is designed to work with the default configuration for blanket and price contracts. Any change to the blanket and price contract defaults in Maximo may result in the interface not functioning correctly. The MXPC integration object used by this interface does not include payment schedules, per the default configuration of Maximo blanket and price contracts. If you change the default settings in Maximo, test this interface to ensure that it functions correctly.</td>
</tr>
</tbody>
</table>
## MXPOInterface—Purchase Order Interface

<table>
<thead>
<tr>
<th>Interface</th>
<th>MXPOInterface</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detail Description</td>
<td>Purchase Order Interface</td>
</tr>
<tr>
<td></td>
<td>This interface allows for bidirectional synchronization of purchase order information between Maximo and any external system.</td>
</tr>
<tr>
<td></td>
<td>Maximo sends out the entire PO or just the PO header, depending on the statuses specified in the POSEND control.</td>
</tr>
</tbody>
</table>

### Outbound Implementation Details

| Integration Point Processing Class Functionality | None |
| Interface Controls                              | POSEND control identifies all statuses at which Maximo will send out the entire PO. It can have multiple values. |

### Inbound Implementation Details

| Integration Point Processing Class Functionality | For an overview of the generic inbound processing functionality, see "The STATUSIFACE Field and Inbound Processing," on page B-25. |
|                                                  | The purchase order integration point processing class overrides the generic functionality as follows: |
|                                                  | If the STATUSIFACE field is not present or set to 0, and the PO exists in Maximo with a status that is a synonym of APPR, the inbound processing attempts to unapprove the PO, update it with new values from the inbound message, and change the status to the status in the inbound message. |
|                                                  | If the inbound PO is a release PO that needs to be added in Maximo, then the PO inbound processing identifies the contract revision details from the PO and attempts to create a release for that PO. |
| Interface Processing Rules                       | None |
| Interface Controls                               | None |
| Comments                                         | None |
## MXPRInterface—Purchase Requisition Interface

<table>
<thead>
<tr>
<th>Interface</th>
<th>MXPRInterface</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Detail Description</strong></td>
<td>Purchase Requisition Interface</td>
</tr>
<tr>
<td>This interface allows for bidirectional synchronization of purchase request information between Maximo and any external system. Maximo sends out the entire PR or just the PR header, depending on the statuses specified in the PRSEND control.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Outbound Implementation Details</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Integration Point Processing</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Class Functionality</strong></td>
<td></td>
</tr>
<tr>
<td>For information about the generic outbound processing of the STATUSIFACE field, see &quot;The STATUSIFACE Field and Outbound Processing,&quot; on page B-24.</td>
<td></td>
</tr>
<tr>
<td><strong>Interface Processing Rules</strong></td>
<td>See &quot;Outbound Processing Rules for Purchasing Interfaces,&quot; on page B-26.</td>
</tr>
<tr>
<td><strong>Interface Controls</strong></td>
<td>The PRSEND control identifies all statuses at which Maximo will send out the entire PO. It can have multiple values.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inbound Implementation Details</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Integration Point Processing</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Class Functionality</strong></td>
<td></td>
</tr>
<tr>
<td>For an overview of the generic inbound processing functionality, see &quot;The STATUSIFACE Field and Inbound Processing,&quot; on page B-25.</td>
<td></td>
</tr>
<tr>
<td>The purchase requisition integration point processing class overrides the generic functionality as follows:</td>
<td></td>
</tr>
<tr>
<td>If the PR currently exists in Maximo, and the inbound message is not a status change but a document update, the following processing takes place:</td>
<td></td>
</tr>
<tr>
<td>If the STATUSIFACE field is not present or set to 0, and the PR currently exists in Maximo with a status that is a synonym of APPR, then the PR inbound processing attempts to unapprove the PR, update it with new values from the inbound message, and change the status to the status specified in the inbound message.</td>
<td></td>
</tr>
<tr>
<td>For all other cases, the generic inbound processing takes place.</td>
<td></td>
</tr>
<tr>
<td><strong>Interface Processing Rules</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Interface Controls</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Comments</strong></td>
<td>None</td>
</tr>
</tbody>
</table>
## MXINVOICEInterface—Invoice Interface

<table>
<thead>
<tr>
<th>Interface</th>
<th>MXINVOICEInterface</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Detail Description</strong></td>
<td>Invoice Interface</td>
</tr>
<tr>
<td>This interface allows for bidirectional synchronization of purchasing contact information between Maximo and any external system.</td>
<td></td>
</tr>
<tr>
<td>Maximo sends out the entire invoice or just the invoice header, depending on the statuses specified in the IVSEND control.</td>
<td></td>
</tr>
<tr>
<td><strong>Outbound Implementation Details</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Integration Point Processing Class Functionality</strong></td>
<td>None</td>
</tr>
<tr>
<td>For information about the generic outbound processing of the STATUSIFACE field, see &quot;The STATUSIFACE Field and Outbound Processing,&quot; on page B-24.</td>
<td></td>
</tr>
<tr>
<td><strong>Interface Processing Rules</strong></td>
<td>See &quot;Outbound Processing Rules for Purchasing Interfaces,&quot; on page B-26.</td>
</tr>
<tr>
<td><strong>Interface Controls</strong></td>
<td>The IVSEND control identifies all statuses at which Maximo will send out the entire invoice. It can have multiple values.</td>
</tr>
<tr>
<td><strong>Inbound Implementation Details</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Integration Point Processing Class Functionality</strong></td>
<td>The inbound processing creates and updates INVOICECOST lines only if one of the following conditions is met:</td>
</tr>
<tr>
<td>▼ The invoice line does not reference a PO line.</td>
<td></td>
</tr>
<tr>
<td>▼ The corresponding PO line does not have a distribution.</td>
<td></td>
</tr>
<tr>
<td>▼ The PO line does not reference a storeroom.</td>
<td></td>
</tr>
<tr>
<td>In all other cases, the inbound processing ignores INVOICECOST information when creating or updating the invoice.</td>
<td></td>
</tr>
<tr>
<td>If the inbound message provides INVOICETERMS, the inbound processing deletes the old terms and adds the new ones.</td>
<td></td>
</tr>
</tbody>
</table>
Users can optionally specify a price variance in the PRICEVAR field on each invoice line, if invoice matching is done in the external system and price variances determined in the external system are sent to Maximo to create variance transactions to update work orders and storerooms. In order for Maximo to process these variances, a value must exist for OWNERSYSID in the inbound invoice, and it cannot be the same as the value of MAXVARS.MXSYSID.

The IVMATCH collaboration switch lets Maximo create and approve an invoice without identifying the matched receipt lines for the invoice line and determining the variances for each line. This switch lets the external system provide the variances at line level so Maximo can take those variances and create the corresponding variance transactions in the INVOICETRANS, MATRECTRANS, or SERVRECTRANS table.

The IVMATCH collaboration switch requires the following values:

- OWNER1SYSID value is always “THISMX”.
- OWNER2SYSID value is the value of INVOICE.OWNERSYSID.

If the evaluation is false, Maximo ignores the line level invoice variances in the invoice processing.

If the evaluation is true, Maximo does not carry out any invoice matching and any variance information provided at the line level is used to create the corresponding variance transactions in INVOICETRANS/ MATRECTRANS/ SERVRECTRANS as applicable.

The table following this one lists possible INVOICE.OWNERSYSID values, the evaluation that Maximo generates, and the default result of the evaluation.

<table>
<thead>
<tr>
<th>Interface Processing Rules</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface Controls</td>
<td>None</td>
</tr>
</tbody>
</table>
Typical scenarios for using the invoice interface are as follows:

Invoice Matching in Maximo (AP Outbound): Invoices received from third parties—either electronically or manually—are processed by Maximo; that is, they are matched against receipts, if applicable, and then approved. Alternately, payment schedules created in Maximo will result in approved invoices being created based on the schedule. These approved invoices result in a payment advice being sent to an external AP system. The process of matching also results in accounting entries being posted to the general ledger.

Invoice Matching in external system (Variances Inbound): Maximo does not do the invoicing, but accepts matched invoices from external systems and applies any variances back to the respective work orders, storerooms, and so on. The accounting entries related to the accounts payable and/or variances must be recorded in the external system; they will not be sent out by Maximo.

The primary intent of this interface is to provide Maximo with any variance information necessary for updating the work order costs.

### IVMATCH Collaboration Switch Evaluation

<table>
<thead>
<tr>
<th>Value of INVOICE.OWNERSYSID</th>
<th>Evaluation</th>
<th>Result (Default)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Null</td>
<td>OWNER1SYSID=&quot;THISMX&quot; and OWNER2SYSID=&quot;THISMX&quot;</td>
<td>False</td>
</tr>
<tr>
<td>MXSYSID</td>
<td>OWNER1SYSID=&quot;THISMX&quot; and OWNER2SYSID=&quot;THISMX&quot;</td>
<td>False</td>
</tr>
<tr>
<td>EXTSYSID</td>
<td>OWNER1SYSID=&quot;THISMX&quot; and OWNER2SYSID=&quot;EXT&quot;</td>
<td>True</td>
</tr>
<tr>
<td>Any other value</td>
<td>OWNER1SYSID=&quot;THISMX&quot; and OWNER2SYSID=&quot;EXT&quot;</td>
<td>True</td>
</tr>
</tbody>
</table>
Outbound Processing Rules for Work Order Interfaces

This section describes the following types of interfaces:

- Work order
- Work order detail

Unlike the purchasing document interfaces, the work order document interfaces do not have a STATUSIFACE field or status change notification functionality. They are first sent out when the work order reaches the status of the WOSTART control and on every update thereafter, regardless of the status of the work order.

Users can configure the start value in the WOSTART control. Any status that you specify in the control is the value, not the MAXVALUE. If multiple synonym values exist for a status, list all applicable synonyms.

The following table describes the conditions for sending outbound work order documents, and the content of the outbound message.

**NOTE** All conditions must be met.

<table>
<thead>
<tr>
<th>Condition(s)</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ The status is not listed in the WOSTART control.</td>
<td>No output</td>
</tr>
<tr>
<td>▼ The document has not been sent out earlier.</td>
<td></td>
</tr>
<tr>
<td>▼ The status is listed in the WOSTART control.</td>
<td>The entire document</td>
</tr>
<tr>
<td>▼ The status is not listed in the WOSTART control.</td>
<td>The entire document</td>
</tr>
<tr>
<td>▼ The document has been sent out earlier.</td>
<td></td>
</tr>
</tbody>
</table>

The following outbound processing rule performs the processes described above:

<table>
<thead>
<tr>
<th>Rule</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>SKIPWO</td>
<td>Skips the work order (does not send it out) if both of the following conditions are true:</td>
</tr>
<tr>
<td></td>
<td>▼ The document has not been sent out before.</td>
</tr>
<tr>
<td></td>
<td>▼ The new status is not listed in the WOSTART control.</td>
</tr>
</tbody>
</table>
MXWOInterface—Work Order Interface

<table>
<thead>
<tr>
<th>Interface</th>
<th>MXWOInterface</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detail Description</td>
<td>Work Order Interface</td>
</tr>
<tr>
<td></td>
<td>This interface allows for bidirectional synchronization of work order information between Maximo and any external system.</td>
</tr>
<tr>
<td></td>
<td>Maximo first sends out a work order when it reaches the status in the WOSTART control, then on all updates thereafter.</td>
</tr>
</tbody>
</table>

**Outbound Implementation Details**

<table>
<thead>
<tr>
<th>Integration Point Processing Class Functionality</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>For information about the generic outbound processing of the STATUSIFACE field, see &quot;The STATUSIFACE Field and Outbound Processing,&quot; on page B-24.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Outbound work orders are sent out on all updates once the work order reaches the start status.</td>
<td></td>
</tr>
</tbody>
</table>

| Interface Controls | The WOSTART control specifies the statuses at which Maximo first sends out the work order. It can have multiple values. |

**Inbound Implementation Details**

<table>
<thead>
<tr>
<th>Integration Point Processing Class Functionality</th>
<th>See &quot;The STATUSIFACE Field and Inbound Processing,&quot; on page B-25.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note: Since the work order interface does not use the STATUSIFACE field, the only STATUSIFACE inbound processing that applies is that which takes place when the STATUSIFACE is not provided or when its value is 0.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Interface Processing Rules</th>
<th>None</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Interface Controls</th>
<th>None</th>
</tr>
</thead>
</table>
### MXWODETAILInterface—Work Order Details Interface

<table>
<thead>
<tr>
<th>Interface</th>
<th>MXWODETAILInterface</th>
</tr>
</thead>
</table>
| **Comments** | The work order interface contains all the information defined by Maximo on the work order, but it does not provide additional information about projects/financial control data, equipment, and locations that is not part of the standard work order in Maximo. If necessary, you can add the additional information via user fields.  
Maximo treats work order tasks as work orders. Both have similar properties and they are stored in the same table. If a user creates a work order, adds tasks/child work orders to the work order, and then approves the work order, the Inherit Status Changes flag on the work order indicates whether approval of the work order also results in approval of all the tasks/child work orders of that particular work order. The default is Y, so when a work order is approved, any child tasks or work orders that inherit the parent’s approval based on this flag are also approved.  
The outbound event listener on the work order MBO receives multiple independent events—one for each work order—and they are processed and sent out independently. Therefore, a work order with three tasks and two child work orders will result in six independent outbound work orders. |

### Outbound Implementation Details

| Integration Point Processing Class Functionality | None |
| Interface Controls | The WOSTART control specifies the statuses at which Maximo first sends out the work order. It can have multiple values. |
The MBO relationship used to retrieve the RESERVATIONS MBO is different from the one used in the Work Order Tracking application. Therefore, the changed attribute in the Maximo XML is not set for any values from this MBO.

This interface does not have any inbound integration points.
Transaction Interfaces

This section describes the following types of interfaces:

- Receipts
- Material receipts
- General ledger transactions
- Labor time reporting
## MXRECEIPTInterface—Receipts Interface

<table>
<thead>
<tr>
<th>Interface</th>
<th>MXRECEIPTInterface</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Detail Description</strong></td>
<td>Receipts interface</td>
</tr>
</tbody>
</table>

This interface allows for bidirectional synchronization of site-level purchase order receipt information between Maximo and any external system. The same interface processes material receipts and service receipts.

In the outbound direction, this interface processes purchase order receipts, transfers (movements against receipts or receipts against internal POs), and returns (returns to vendors after receipt inspection, or returns to vendor after acceptance and goods movement from the inspection holding location).

In the inbound direction, this interface processes receipts and returns. For receipts that require inspection, the user-defined field INSPECTED indicates if the receipt line was inspected in the external system. The interface does not process TRANSFERs independently; each transfer is associated with a receipt.

This interface uses a merged integration object with the following sub-records:

- ▼ MATRECTRANS
- ▼ SERVRETRANS

### Outbound Implementation Details

**Integration Point Processing Class Functionality**

Processes receipts with issue type of RECEIPT or RETURN. If no issue type is specified, it is treated as RECEIPT.

Processing for RECEIPT issue type and Inspection Required = N:

Material and service receipts:

- ▼ Uses inbound RECEIPTQUANTITY (for material receipts) or QTYTORECEIVE (for service receipts) to create the Maximo receipt.

- ▼ Maps inbound REJECTEDQTY to the REJECTEDQTY field in the Maximo receipt.

- ▼ Ignores any other quantities.

- ▼ Does not look at the inbound INSPECTED field.

Material receipts (MATRECTRANS records):

Sends out only RECEIPT and RETURN type records and TRANSFER type records containing a PONUM (not a storeroom transfer). Sends out new receipts only, not updates to existing receipts.
<table>
<thead>
<tr>
<th>Interface</th>
<th>MXRECEIPTInterface (Continued)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integration Point Processing Class Functionality</td>
<td>Service receipts (SERVRECTRANS records):</td>
</tr>
<tr>
<td></td>
<td>Sends out RECEIPT and RETURN type records. Sends out new receipts and updates to existing receipts.</td>
</tr>
<tr>
<td>Interface Processing Rules</td>
<td>Service receipts:</td>
</tr>
<tr>
<td></td>
<td>Sends out records when the status is equal to a value in the SERVRECSTAT control (default COMP). This occurs under the following conditions:</td>
</tr>
<tr>
<td></td>
<td>• A record that does not require inspection is inserted.</td>
</tr>
<tr>
<td></td>
<td>• An existing record is updated in Maximo and the status field is changed to COMP.</td>
</tr>
<tr>
<td>Interface Controls</td>
<td>SERVRECSTAT control identifies all statuses at which Maximo will send out service receipt transactions. It can have multiple values. By default, its value is COMP.</td>
</tr>
<tr>
<td>Inbound Implementation Details</td>
<td></td>
</tr>
<tr>
<td>Integration Point Processing Class Functionality</td>
<td>Processing for RECEIPT issue type and Inspection Required = Y:</td>
</tr>
<tr>
<td></td>
<td>Material receipts:</td>
</tr>
<tr>
<td></td>
<td>INSPECTED = N:</td>
</tr>
<tr>
<td></td>
<td>• Uses only the inbound RECEIPTQUANTITY field to create the Maximo receipt; ignores accepted and rejected quantity values.</td>
</tr>
<tr>
<td></td>
<td>• Creates Maximo receipt with STATUS = WINSP (waiting inspection) and quantity derived from RECEIPTQUANTITY.</td>
</tr>
<tr>
<td></td>
<td>INSPECTED = Y:</td>
</tr>
<tr>
<td></td>
<td>• Uses the inbound RECEIPTQUANTITY, ACCEPTEDQTY, and REJECTEDQTY fields to create the Maximo receipt.</td>
</tr>
<tr>
<td></td>
<td>• Does not allow partial inspections or acceptances. RECEIPTQUANTITY must equal ACCEPTEDQTY + REJECTEDQTY.</td>
</tr>
<tr>
<td></td>
<td>• Creates Maximo receipt with STATUS = WASSET (if rotating item) or COMP (all other items).</td>
</tr>
<tr>
<td></td>
<td>• Depending on quantities specified, can create up to three transactions—one RECEIPT, one TRANSFER, and one RETURN.</td>
</tr>
</tbody>
</table>
## Service receipts:

**INSPECTED = N:**

- Uses the inbound AMTTORECEIVE (if POLINE order quantity is null) or QTYTORECEIVE (in other cases) to create Maximo receipt; ignores all other quantity values.

- Creates Maximo receipt with STATUS = WINSP (waiting inspection) and quantity derived from QTYTORECEIVE.

**INSPECTED = Y:**

- Uses the inbound QTYTORECEIVE, ACCEPTEDQTY, and REJECTEDQTY fields to create Maximo receipt.

- Does not allow partial inspections or acceptances. QTYTORECEIVE must equal ACCEPTEDQTY + REJECTEDQTY.

- Creates a single transaction of type RECEIPT, with STATUS = COMP.

### Processing for RETURN issue type:

**Material and service receipts:**

- Accepts return transactions for a POLINE only if there was an earlier receipt for the same line; if Inspection Required = Y for the POLINE, the receipt must have been approved. Otherwise, reports an error.

- Uses only the inbound RECEIPTQUANTITY (for material receipts) or QTYTORECEIVE (for service receipts) field to create the Maximo receipt; ignores all other quantity values.

- Creates a single transaction with issue type RETURN and the credit GL account as the RBNI account.

## Interface Processing Rules

None

## Interface Controls

None

## Comments

You do not have to specify whether an inbound receipt is a material receipt or a service receipt. The integration processing determines this from the POLINE.

All quantities, including return quantities, must be positive.

You can distinguish between the two types of transfer records by the value in the RECEIPTREFID field, as follows:

- Null: a receipt against an internal PO
- Not null: movement against a receipt

Do not specify a RECEIPTREF value for returns. Returns are processed independently of the corresponding receipt.
## MXRCVROTITMInterface—Material Receipts Interface

<table>
<thead>
<tr>
<th>Interface</th>
<th>MXRCVROTITMInterface</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Detail Description</strong></td>
<td>Material Receipts Interface</td>
</tr>
</tbody>
</table>

This interface allows for inbound synchronization of site-level material (non-service) receipt and transfer information, including rotating items (assets) defined against a receipt.

This interface does not let you specify a status for the receipt; the status is always assumed to be COMP.

This interface differs from the MXRECEIPTInterface in that it processes material receipts exclusively and lets you identify serialized rotating assets to be created in the case of rotating item receipts.

You can receive rotating items with or without asset numbers. If you receive them without asset numbers, you must manually specify the asset numbers via the Receive Rotating Items dialog box in the Purchasing Receiving application.

If you do not process inbound service receipts, you can use this interface for all material receipts.

### Outbound Implementation Details

| Integration Point Processing Class Functionality | Not applicable |
| Interface Processing Rules | Not applicable |
| Interface Controls | Not applicable |

### Inbound Implementation Details

| Integration Point Processing Class Functionality | Reports an error if the PO line being processed has LINETYPE = SERVICE or STDSERVICE. |

Processing is the same as the Inbound Integration Point Processing Class Functionality for material receipts using the MXRECEIPTInterface, with the following additional processing:

Receipts: If the line item is a rotating item and Inspection Required = N, or INSPECTED = Y, checks for asset information corresponding to the item provided in the interface. If the information is available, validates the number of asset records to ensure it equals the RECEIPTQTY or ACCEPTEDQTY (whichever applies). If it does, invokes the Maximo receiving functionality and creates asset as required.

Returns: If the item being returned is a rotating item, ignores any asset information. In Maximo, return of a rotating type item does not affect the assets created by the original receipt. The asset records remain unchanged; only the item balances are updated (if applicable).

<p>| Interface Processing Rules | None |
| Interface Controls | None |</p>
<table>
<thead>
<tr>
<th>Interface</th>
<th>MXRCVROTITMInterface (Continued)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comments</td>
<td>For rotating items, the number of inbound transactions is one more than the number of rotating items. There is one transaction for the total receipt quantity, and one transaction for each rotating item associated with the receipt. For example, for a receipt of ten rotating items, there will be eleven transactions.</td>
</tr>
</tbody>
</table>
## MXGLTXNInterface—General Ledger (Journal Entry) Interface

<table>
<thead>
<tr>
<th>Interface</th>
<th>MXGLTXNInterface</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Detail Description</strong></td>
<td>General Ledger (Journal Entry) Interface</td>
</tr>
<tr>
<td></td>
<td>This interface allows for the posting of site-level transactions generated in Maximo to an external general ledger application for accounting reconciliation.</td>
</tr>
<tr>
<td></td>
<td>This interface is available for outbound processing only.</td>
</tr>
<tr>
<td></td>
<td>This interface uses a merged integration object with the following sub-records:</td>
</tr>
<tr>
<td></td>
<td>▼ SERVRECTRANS</td>
</tr>
<tr>
<td></td>
<td>▼ MATRECTRANS</td>
</tr>
<tr>
<td></td>
<td>▼ INVTRANS</td>
</tr>
<tr>
<td></td>
<td>▼ INVOICETRANS</td>
</tr>
<tr>
<td></td>
<td>▼ MATUSETRANS</td>
</tr>
<tr>
<td></td>
<td>▼ LABTRANS</td>
</tr>
<tr>
<td></td>
<td>▼ TOOLSTRANS</td>
</tr>
<tr>
<td></td>
<td>The SOURCEMBO field identifies the table in which the transaction originated. Its value is derived from the GLSOURCEMBO synonym domain.</td>
</tr>
</tbody>
</table>

### Outbound Implementation Details

<table>
<thead>
<tr>
<th>Integration Point Processing Class Functionality</th>
<th>Service Receipts</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Service receipts include the following:</td>
</tr>
<tr>
<td></td>
<td>▼ Accounting entries for non-distributed service receipts created by the Receiving application for services ordered on POs (ISSUETYPE = RECEIPT, COSTINFO=1)</td>
</tr>
<tr>
<td></td>
<td>▼ Accounting entries for distributed service receipts created by the Receiving application for services ordered on POs (ISSUETYPE = POCOST)</td>
</tr>
<tr>
<td></td>
<td>▼ Accounting entries for PO services with Receipt Required = N that are directly invoiced instead of being received (ISSUETYPE = INVOICE)</td>
</tr>
<tr>
<td></td>
<td>▼ Accounting entries for services that are not against POs and are invoiced directly (ISSUETYPE = INVOICE)</td>
</tr>
<tr>
<td></td>
<td>▼ Invoice variance transactions recorded against service receipts (ISSUETYPE = INVOICE)</td>
</tr>
</tbody>
</table>
### Integration Point Processing

#### Class Functionality

<table>
<thead>
<tr>
<th>Service receipts (SERVRECTRANS) processing:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sets SOURCEMBO to SERVRECTRANS and ISSUETYPE to value listed above.</td>
</tr>
<tr>
<td><strong>INVOICE</strong> type transactions: sends out on insert.</td>
</tr>
<tr>
<td><strong>POCOST</strong> and <strong>RECEIPT</strong> type transactions: if Inspection Required = N for the corresponding POLINE, sends out on insert. If inspection required = Y, sends out when status is changed to COMP.</td>
</tr>
</tbody>
</table>

#### Material receipts:

Material receipts include the following:

- Accounting entries for non-distributed material receipts created by the Receiving application for items/tools ordered on POs (ISSUETYPE = RECEIPT, COSTINFO = 1)

- Accounting entries for distributed material receipts created by the Receiving application for items/tools ordered on POs (ISSUETYPE = POCOST)

- Accounting entries for Item transfers between storerooms (ISSUETYPE = TRANSFER and PONUM=NULL)

- Accounting entries for receipt inspection transfers of items between the receipt inspection storeroom and the PO line storeroom (ISSUETYPE = TRANSFER and RECEIPTREF!=NULL and PONUM!=NULL)

- Accounting entries for receipts against internal POs (ISSUETYPE = TRANSFER and RECEIPTREF=NULL and PONUM!=NULL)

- Accounting entries for receipt inspection goods return of items and materials (ISSUETYPE = RETURN)

- Accounting entries for return to vendor from a storeroom or direct issue PO lines (ISSUETYPE = RETURN)

- Invoice variance transactions recorded against material receipts (ISSUETYPE = INVOICE)

- Accounting transaction for increasing the kit item’s INVENTORY control account value when kits are made (ISSUETYPE = KITMAKE).

- Accounting transactions for increasing the INVENTORY control account for each constituent item of a kit when a kit is disassembled (ISSUETYPE = KITBREAK)
<table>
<thead>
<tr>
<th>Interface</th>
<th>MXGLTXNInterface (Continued)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integration Point Processing Class Functionality</td>
<td>Material receipts (MATRETRANS) processing:</td>
</tr>
<tr>
<td></td>
<td>Sets SOURCEMBO to MATRETRANS and ISSUETYPE to value listed above.</td>
</tr>
<tr>
<td></td>
<td>Sends out INVOICE, RECEIPT, TRANSFER, RETURN, KITMAKE and KITBREAK type transactions on insert.</td>
</tr>
<tr>
<td></td>
<td>For POCOST type transactions on insert if status is COMP (that is, on insert if Inspection Required = N, and when status changed to COMP if Inspection Required = Y).</td>
</tr>
<tr>
<td>Inventory Adjustments</td>
<td>Inventory adjustment transactions include the following:</td>
</tr>
<tr>
<td></td>
<td>▼ Inventory current balance adjustments (ITTYPE = CURBALADJ)</td>
</tr>
<tr>
<td></td>
<td>▼ Inventory standard/average cost adjustments (ITTYPE = STDCOSTADJ/AVCOSTADJ)</td>
</tr>
<tr>
<td></td>
<td>▼ Cost difference when a kit is disassembled and there is a difference between the value of the kit and the sum of the kit component values (ITTYPE = KITCOSTVAR)</td>
</tr>
<tr>
<td></td>
<td>▼ Physical count reconciliation (ITTYPE = RECBALADJ)</td>
</tr>
<tr>
<td></td>
<td>▼ Capitalized cost adjustment (ITTYPE = CAPCSTADJ)</td>
</tr>
<tr>
<td></td>
<td>▼ Standard cost receipt adjustment (ITTYPE = STDRECADJ)</td>
</tr>
<tr>
<td>Inventory adjustment (INVTRANS) processing</td>
<td>Sets SOURCEMBO to INVTRANS and ITTYPE to value listed above.</td>
</tr>
<tr>
<td></td>
<td>Does not send out transactions with ITTYPE = INSERTITEM, CREATEASSET and PHYSCNT, as they are considered audit records rather than accounting transactions.</td>
</tr>
<tr>
<td></td>
<td>Sends out all other transactions on insert.</td>
</tr>
</tbody>
</table>
## Transaction Interfaces

<table>
<thead>
<tr>
<th>Interface</th>
<th>Integration Point Processing Class Functionality</th>
<th>Invoice Transactions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Invoice transactions include the following transactions created by invoice approval:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>▼ Invoice TOTAL transaction (amount payable to invoice vendor, TRANSTYPE = TOTAL)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>▼ Invoice line tax transactions (tax accounting for each TAX code for an invoice line, TRANSTYPE = TAX1:TAX5)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>▼ Invoice currency variance transaction (TRANSTYPE = CURVAR)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>▼ Invoice price variance transactions (TRANSTYPE = INVCEVAR)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Invoice transaction (INVTRANS) processing:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sets SOURCEMBO to INVOICETRANS and TRANSTYPE to the value listed above.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sends out all transactions when they are created.</td>
</tr>
<tr>
<td></td>
<td>Material Issues and Returns</td>
<td>Material issues and returns transactions include the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>▼ Accounting entries for items issued from a storeroom in Maximo (ISSUETYPE = ISSUE)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>▼ Accounting entries for items returned to a storeroom (ISSUETYPE = RETURN)</td>
</tr>
<tr>
<td></td>
<td>Material issue and return (MATUSETRANS) processing:</td>
<td>Sets SOURCEMBO to MATUSETRANS and ISSUETYPE to value listed above.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Does not send out direct issue transactions that are created by PO receiving/ invoice variances in MATUSETRANS, as they are accounted for in MATRECTRANS. Identifies such transactions by their PONUM, so all MATUSETRANS transactions that have a PO reference are not sent out by this interface.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sends out all other transaction at time of creation.</td>
</tr>
<tr>
<td>Interface</td>
<td>MXGLTXNInterface (Continued)</td>
<td></td>
</tr>
<tr>
<td>---------------------------------</td>
<td>------------------------------</td>
<td></td>
</tr>
<tr>
<td>Integration Point Processing</td>
<td>Labor Transaction (LABRTRANS) processing</td>
<td></td>
</tr>
<tr>
<td>Class Functionality</td>
<td>Sets SOURCEMBO to LABTRANS.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>If approval not required, send out labor actuals against work orders on insert.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>If approval required, sends out transaction after it is approved (when GENAPPSERVRECEIPT is Y).</td>
<td></td>
</tr>
<tr>
<td>Tool Transaction (TOOLTRANS)</td>
<td>Sets SOURCEMBO to TOOLTRANS.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sends out usage of tools on work orders on creation of TOOLTRANS.</td>
<td></td>
</tr>
</tbody>
</table>

| Interface Processing Rules      | None |
| Interface Controls              | The GLSOURCE control defines the SOURCEMBO values that the processing skips. By default, this control has no values. |

**Inbound Implementation Details**

| Integration Point Processing   | Not applicable |
| Class Functionality            |                |
| Interface Processing Rules     | Not applicable |
| Interface Controls             | Not applicable |
| Comments                       | None           |
**MXEMPACTInterface—Labor Time Reporting Interface**

<table>
<thead>
<tr>
<th>Interface</th>
<th>MXEMPACTInterface</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detail Description</td>
<td>Labor time reporting interface</td>
</tr>
<tr>
<td></td>
<td>This interface allows for posting site-level actual time reported in Maximo to external applications.</td>
</tr>
<tr>
<td></td>
<td>Maximo sends out approved labor actuals and accepts all inbound labor actuals regardless of status.</td>
</tr>
<tr>
<td></td>
<td>Note: Maximo does not update the status of existing records when processing inbound transactions.</td>
</tr>
</tbody>
</table>

### Outbound Implementation Details

| Integration Point Processing Class Functionality | None |
| Interface Processing Rules | Sends out the labor actual record only if it is approved. |
| Interface Controls | None |

### Inbound Implementation Details

| Integration Point Processing Class Functionality | Treats a transaction with a null action as an Add. LABTRANSID is not allowed; if provided, it causes an error. |
| Interface Processing Rules | None |
| Interface Controls | None |
| Comments | None |
Some MBOs are flagged as *system* objects in Maximo. These MBOs are generally used for application or metadata configuration, and updating data through these MBOs may require specific post processing activities in Maximo (for example, reconfiguring the database). Any interface that uses system MBOs is defined as a system interface. System interfaces are typically used to exchange configuration data between different instances of Maximo, install adapters, or apply updates and patches to Maximo or the adapters.

**CAUTION**

Do not modify these interfaces or their corresponding integration objects and points in any way. Do not disable these interfaces for the default external system, EXTSYS1. Doing so could result in problems with applying product updates and patches from IBM Corporation.

The following restrictions apply to the use of system interfaces and integration objects:

- You cannot enable listeners on outbound system interfaces.
- You cannot process system interfaces via interface tables or flat files.
- External systems that use interface table type end points cannot use system interfaces.
- You must specify an action code on inbound system interfaces.
### MXINTOBJInterface—Integration Object Interface

<table>
<thead>
<tr>
<th>Interface</th>
<th>MXINTOBJInterface</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Detail Description</strong></td>
<td>Integration object interface</td>
</tr>
<tr>
<td></td>
<td>This interface allows for the inbound and outbound synchronization of integration objects, including specific columns and integration points.</td>
</tr>
<tr>
<td></td>
<td>This interface is used internally for loading adapter configurations into Maximo.</td>
</tr>
<tr>
<td></td>
<td>You can use this interface to export new and modified integration objects from one system to another.</td>
</tr>
</tbody>
</table>

#### Outbound Implementation Details

| Integration Point Processing Class Functionality | None |
| Interface Processing Rules                   | None |
| Interface Controls                            | None |

#### Inbound Implementation Details

| Integration Point Processing Class Functionality | None |
| Interface Processing Rules                   | None |
| Interface Controls                            | None |

**Comments**: The validations that apply to users deleting and modifying predefined integration objects also apply to the modification of integration objects by inbound integration transactions.
### MXINTTYPEInterface—Integration Adapter Interface

<table>
<thead>
<tr>
<th>Interface</th>
<th>MXINTTYPEInterface</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Detail Description</strong></td>
<td>Integration adapter interface</td>
</tr>
<tr>
<td></td>
<td>This interface allows for the inbound and outbound synchronization of adapters, adapter configuration data, and interface controls and their default values.</td>
</tr>
<tr>
<td></td>
<td>This interface is used internally for loading adapter configurations into Maximo.</td>
</tr>
<tr>
<td></td>
<td>You can use this interface to export new and modified adapters from one system to another.</td>
</tr>
</tbody>
</table>

#### Outbound Implementation Details

| Integration Point Processing Class Functionality | None |
| Interface Processing Rules                      | None |
| Interface Controls                               | None |

#### Inbound Implementation Details

| Integration Point Processing Class Functionality | None |
| Interface Processing Rules                      | None |
| Interface Controls                               | None |

**Comments**

The validations that apply to users deleting and modifying predefined adapters apply to the modification of adapters by inbound integration transactions.
**MXINTIFACEInterface—Integration Master Interface**

<table>
<thead>
<tr>
<th>Interface</th>
<th>MXINTIFACEInterface</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detail Description</td>
<td>Interface master interface</td>
</tr>
<tr>
<td></td>
<td>This interface allows for the inbound and outbound synchronization of interfaces, their inbound and outbound definitions, and rules and conditions defined for the interfaces.</td>
</tr>
<tr>
<td></td>
<td>This interface is used internally for loading adapter configurations into Maximo.</td>
</tr>
<tr>
<td></td>
<td>You can use this interface to export new and modified interfaces from one system to another.</td>
</tr>
</tbody>
</table>

**Outbound Implementation Details**

| Integration Point Processing | None |
| Class Functionality | None |
| Interface Processing Rules | None |
| Interface Controls | None |

**Inbound Implementation Details**

| Integration Point Processing | None |
| Class Functionality | None |
| Interface Processing Rules | None |
| Interface Controls | None |

**Comments**

| The validations that apply to users deleting and modifying predefined interfaces apply to the modification of interfaces by inbound integration transactions. |
### MXENDPOINTInterface—End Point Master Interface

<table>
<thead>
<tr>
<th>Interface</th>
<th>MXENDPOINTInterface</th>
</tr>
</thead>
</table>
| **Detail Description**     | End point master interface  
This interface allows for the inbound and outbound synchronization of end point information.  
This interface is used internally for loading end points into Maximo.  
You can use this interface to export new and modified end points from one system to another. |

#### Outbound Implementation Details

| Integration Point Processing Class Functionality | None |
| Interface Processing Rules                     | None |
| Interface Controls                              | None |

#### Inbound Implementation Details

| Integration Point Processing Class Functionality | None |
| Interface Processing Rules                     | None |
| Interface Controls                              | None |
| **Comments**                                   | If the end point uses a user-defined handler, that handler must be created first, or errors can occur during inbound processing. |
**MXEXTSYSInterface—External System Interface**

<table>
<thead>
<tr>
<th>Interface</th>
<th>MXEXTSYSInterface</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Detail Description</strong></td>
<td>External system interface</td>
</tr>
<tr>
<td></td>
<td>This interface allows for the inbound and outbound synchronization of the definition of an external system, the interfaces used by the system, and the interface controls values for the system.</td>
</tr>
<tr>
<td></td>
<td>This interface is used internally for loading sample external systems into Maximo.</td>
</tr>
<tr>
<td></td>
<td>You can use this interface to export new and modified external systems from one system to another.</td>
</tr>
<tr>
<td><strong>Outbound Implementation Details</strong></td>
<td></td>
</tr>
<tr>
<td>Integration Point Processing Class Functionality</td>
<td>None</td>
</tr>
<tr>
<td>Interface Processing Rules</td>
<td>None</td>
</tr>
<tr>
<td>Interface Controls</td>
<td>None</td>
</tr>
<tr>
<td><strong>Inbound Implementation Details</strong></td>
<td></td>
</tr>
<tr>
<td>Integration Point Processing Class Functionality</td>
<td>None</td>
</tr>
<tr>
<td>Interface Processing Rules</td>
<td>None</td>
</tr>
<tr>
<td>Interface Controls</td>
<td>None</td>
</tr>
<tr>
<td><strong>Comments</strong></td>
<td>The validations that apply to users adding, deleting and modifying predefined external systems apply to the modification of external systems by inbound integration transactions.</td>
</tr>
<tr>
<td></td>
<td>All interfaces, end points, and controls that the system uses must be added before the external system is added.</td>
</tr>
</tbody>
</table>
### MXMESSAGEInterface—Message Interface

<table>
<thead>
<tr>
<th>Interface</th>
<th>MXMESSAGEInterface</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detail Description</td>
<td>Message interface</td>
</tr>
<tr>
<td></td>
<td>This interface allows for the inbound and outbound synchronization of system error and warning messages.</td>
</tr>
<tr>
<td></td>
<td>This interface is used internally for loading messages into Maximo.</td>
</tr>
<tr>
<td></td>
<td>You can use this interface to export new and modified messages from one system to another.</td>
</tr>
</tbody>
</table>

#### Outbound Implementation Details

<table>
<thead>
<tr>
<th>Integration Point Processing</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class Functionality</td>
<td>None</td>
</tr>
<tr>
<td>Interface Processing Rules</td>
<td>None</td>
</tr>
<tr>
<td>Interface Controls</td>
<td>None</td>
</tr>
</tbody>
</table>

#### Inbound Implementation Details

<table>
<thead>
<tr>
<th>Integration Point Processing</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class Functionality</td>
<td>None</td>
</tr>
<tr>
<td>Interface Processing Rules</td>
<td>None</td>
</tr>
<tr>
<td>Interface Controls</td>
<td>None</td>
</tr>
<tr>
<td>Comments</td>
<td>Manually update the messages.xml file with the messages that you add via this interface.</td>
</tr>
</tbody>
</table>
## MXOBJECTCFGInterface—MBO Configuration Interface

<table>
<thead>
<tr>
<th>Interface</th>
<th>MXOBJECTCFGInterface</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Detail Description</strong></td>
<td>MBO configuration interface</td>
</tr>
<tr>
<td></td>
<td>This interface allows for the inbound and outbound synchronization of MBO definition information, including MBO attributes.</td>
</tr>
<tr>
<td></td>
<td>This interface is used internally for loading sample MBO configurations into Maximo.</td>
</tr>
<tr>
<td></td>
<td>You can use this interface to export new and modified MBO configurations from one system to another.</td>
</tr>
</tbody>
</table>

### Outbound Implementation Details

<table>
<thead>
<tr>
<th>Integration Point Processing Class Functionality</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface Processing Rules</td>
<td>None</td>
</tr>
<tr>
<td>Interface Controls</td>
<td>None</td>
</tr>
</tbody>
</table>

### Inbound Implementation Details

<table>
<thead>
<tr>
<th>Integration Point Processing Class Functionality</th>
<th>Delay the save validation on MAXOBJECTCFG until all the attributes are successfully added into Maximo.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface Processing Rules</td>
<td>None</td>
</tr>
<tr>
<td>Interface Controls</td>
<td>None</td>
</tr>
</tbody>
</table>

**Comments**

Use this interface with caution.

Use only Add, AddChange, or Change action codes when importing data through this interface. Do not use the Replace action unless you completely replace the MAXOBJECTCFG and MAXATTRIBUTECFG data.

The delete action presents the risk of deleting predefined records in these tables.

There is an inbound setting restriction on the CHANGED column, so the XML value is not set to the MBO.

After importing data with this interface, bring down the Maximo application server and run the Maximo Database Configuration application (configdb) in order for the changes to take effect. For more information, refer to the Database Configuration chapter in the Maximo System Administrator’s Guide.
## MXDOMAINInterface—Domain Interface

<table>
<thead>
<tr>
<th>Interface</th>
<th>MXDOMAINInterface</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Detail Description</strong></td>
<td>Domain interface</td>
</tr>
<tr>
<td></td>
<td>This interface allows for the inbound and outbound synchronization of domain information.</td>
</tr>
<tr>
<td></td>
<td>This interface is used internally for loading domains into Maximo.</td>
</tr>
<tr>
<td></td>
<td>You can use this interface to export new and modified domains from one system to another.</td>
</tr>
</tbody>
</table>

### Outbound Implementation Details

| Integration Point Processing Class Functionality | None |
| Interface Processing Rules                      | None |
| Interface Controls                               | None |

### Inbound Implementation Details

| Integration Point Processing Class Functionality | None |
| Interface Processing Rules                      | None |
| Interface Controls                               | None |
| Comments                                          | None |
## MXWFInterface—Workflow Definition Interface

<table>
<thead>
<tr>
<th>Interface</th>
<th>MXWFInterface</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Detail Description</strong></td>
<td>Workflow definition interface</td>
</tr>
<tr>
<td></td>
<td>This interface allows for bidirectional synchronization of workflow definition information, including workflow processes, tasks, nodes, actions and other configuration data required for creating/ maintaining workflow processes in Maximo.</td>
</tr>
<tr>
<td></td>
<td>It is intended primarily as a mechanism to facilitate movement of workflow definitions from test to production systems.</td>
</tr>
<tr>
<td></td>
<td>This interface does not support the deletion or modification of existing workflow processes. If the process exists in the importing system, a new revision of the process is created and must be enabled manually by the user.</td>
</tr>
</tbody>
</table>

### Outbound Implementation Details

<table>
<thead>
<tr>
<th>Integration Point Processing Class Functionality</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface Processing Rules</td>
<td>None</td>
</tr>
<tr>
<td>Interface Controls</td>
<td>None</td>
</tr>
</tbody>
</table>

### Inbound Implementation Details

<table>
<thead>
<tr>
<th>Integration Point Processing Class Functionality</th>
<th>Verifies that action is ADD or Null</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Creates a new process revision.</td>
</tr>
<tr>
<td></td>
<td>For notification templates that are in use by a process, the SENDTO, SUBJECT and MESSAGE attributes of the template are not updated.</td>
</tr>
<tr>
<td></td>
<td>If the communication template is created by the workflow interface, it is created with the data provided in the workflow file.</td>
</tr>
<tr>
<td>Interface Processing Rules</td>
<td>None</td>
</tr>
<tr>
<td>Interface Controls</td>
<td>None</td>
</tr>
<tr>
<td>Interface (Continued)</td>
<td>MXWFInterface</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Comments</td>
<td>Prerequisite: You must process communication templates, actions, and roles before you process workflow transactions.</td>
</tr>
<tr>
<td></td>
<td>All workflow transactions must be in XML format.</td>
</tr>
<tr>
<td></td>
<td>There is an inbound setting restriction on the ENABLED and ACTIVE columns, so the XML value is not set to the MBO. You must manually enable and activate processes. To do so, complete the following steps:</td>
</tr>
<tr>
<td></td>
<td>1 In the Workflow Designer application, display the process that you want to update.</td>
</tr>
<tr>
<td></td>
<td>2 Select <strong>Enable Process</strong> from the Select Action menu.</td>
</tr>
<tr>
<td></td>
<td>3 Select <strong>Activate Process</strong> from the Select Action menu.</td>
</tr>
<tr>
<td></td>
<td>4 Click Save Process.</td>
</tr>
</tbody>
</table>
### MXCTEMPLATEInterface—Communication Template Interface

<table>
<thead>
<tr>
<th>Interface</th>
<th>MXCTEMPLATEInterface</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detail Description</td>
<td>Communication template interface</td>
</tr>
<tr>
<td></td>
<td>This interface allows for bidirectional synchronization of communication templates that are used to create/maintain workflow processes in Maximo.</td>
</tr>
<tr>
<td></td>
<td>A communication template is a definition of a mail message with subject, message, and recipient information that is processed when certain nodes become current, or along specified workflow routing paths between nodes.</td>
</tr>
<tr>
<td></td>
<td>This interface intended to facilitate movement of workflow definitions from test to production systems.</td>
</tr>
</tbody>
</table>

#### Outbound Implementation Details

| Integration Point Processing Class Functionality | None |
| Interface Processing Rules                     | None |
| Interface Controls                              | None |

#### Inbound Implementation Details

| Integration Point Processing Class Functionality | None |
| Interface Processing Rules                      | None |
| Interface Controls                               | None |
| Comments                                         | All workflow transactions must be in XML format. |
MXACTIONInterface—Action Interface

<table>
<thead>
<tr>
<th>Interface</th>
<th>MXACTIONInterface</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detail Description</td>
<td>Action Interface</td>
</tr>
<tr>
<td></td>
<td>This interface allows for bidirectional synchronization of workflow action definitions required to create/maintain workflow processes in Maximo.</td>
</tr>
<tr>
<td></td>
<td>An action is one or more data processing events, such as a change status or set value, that are defined within a workflow process to run along specified workflow routing paths between nodes.</td>
</tr>
<tr>
<td></td>
<td>This interface is intended to facilitate movement of workflow definitions from test to production systems.</td>
</tr>
</tbody>
</table>

Outbound Implementation Details

| Integration Point Processing Class Functionality | None |
| Interface Processing Rules | None |
| Interface Controls | None |

Inbound Implementation Details

| Integration Point Processing Class Functionality | None |
| Interface Processing Rules | None |
| Interface Controls | None |
| Comments | All workflow transactions must be in XML format. |
# MXMAXROLEInterface—Maximo Role Interface

<table>
<thead>
<tr>
<th>Interface</th>
<th>MXMAXROLEInterface</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Detail Description</strong></td>
<td>Maximo Role interface</td>
</tr>
<tr>
<td></td>
<td>This interface allows for bidirectional synchronization of role definitions that are used in configuring workflow processes in Maximo.</td>
</tr>
<tr>
<td></td>
<td>A role is a pointer to a certain table column whose data value will be that of a person ID at run time. Workflow in box assignments and workflow notifications are always made to roles.</td>
</tr>
<tr>
<td></td>
<td>This interface is intended to facilitate movement of workflow definitions from test to production systems.</td>
</tr>
</tbody>
</table>

### Outbound Implementation Details

<table>
<thead>
<tr>
<th>Integration Point Processing Class Functionality</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface Processing Rules</td>
<td>None</td>
</tr>
<tr>
<td>Interface Controls</td>
<td>None</td>
</tr>
</tbody>
</table>

### Inbound Implementation Details

<table>
<thead>
<tr>
<th>Integration Point Processing Class Functionality</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface Processing Rules</td>
<td>None</td>
</tr>
<tr>
<td>Interface Controls</td>
<td>None</td>
</tr>
</tbody>
</table>

| Comments | All workflow transactions must be in XML format. |
Collaboration switches have been designed from an inbound integration point of view to provide users with a way to better manage data synchronization between Maximo and external systems using a concept of ownership. These switches provide the ability to control sub-processes within specific Maximo application functionality based on ownership of different data objects within a transaction.

Most master data and document integration objects in Maximo have an OWNERSYSID attribute present in the primary MBO of the integration object. By default, Maximo's inbound integration processing does not specify any value in this field, and processing of inbound interfaces proceeds as per standard Maximo functionality.

This appendix contains details on the available collaboration switches and their effect on standard Maximo processing.

**Format of Collaboration Switches**

Collaboration switches provide a flexible, user-defined way to control the processing of some inbound transactions, by letting Maximo bypass the default processing for certain types of transactions.

The collaboration switches reside in the Maximo MXCOLLAB table. Each collaboration switch contains four elements, three of which combine to create a unique key. The following table lists these elements. An asterisk (*) indicates the elements that comprise the key.

<table>
<thead>
<tr>
<th>Element</th>
<th>Corresponding MXCOLLAB field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process control ID*</td>
<td>PCID</td>
</tr>
<tr>
<td>System ID 1*</td>
<td>OWNER1SYSID</td>
</tr>
<tr>
<td>System ID 2*</td>
<td>OWNER2SYSID</td>
</tr>
<tr>
<td>Process control value</td>
<td>PCVALUE</td>
</tr>
</tbody>
</table>

**Process Control ID**

The process control ID identifies a business process in a Maximo application, such as the validation of an invoice match, the creation of a blanket PO
release, or the update of a physical inventory count. The prefix of the process control ID indicates the application to which it applies.

**Process Control ID Prefixes**

<table>
<thead>
<tr>
<th>Prefix of Process Control ID</th>
<th>Corresponding Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>INV</td>
<td>Invoice</td>
</tr>
<tr>
<td>ITM</td>
<td>Item</td>
</tr>
<tr>
<td>IV</td>
<td>Inventory</td>
</tr>
<tr>
<td>LT</td>
<td>Labor</td>
</tr>
<tr>
<td>PO</td>
<td>Purchase order</td>
</tr>
<tr>
<td>PR</td>
<td>Purchase requisition</td>
</tr>
<tr>
<td>WO</td>
<td>Work order</td>
</tr>
</tbody>
</table>

For example, the IVRC, IVRCY, and IVWO collaboration switches are all related to inventory processing. For a complete listing of the collaboration switches by application, see page C-9.
System ID 1 and System ID 2

System ID 1 and System ID 2 identify your Maximo system and/or an external system.

The values in these fields vary, depending on the transaction and the objects in the transaction. In general, System ID 1 identifies the system (Maximo or external) that created the object, and System ID 2 identifies the system that created the record being referenced or updated.

Process Control Value

The process control value specifies whether the Maximo business components should bypass default processing for the type of transaction indicated by the process control ID, System ID 1, and the System ID 2. The process control value can be 0 (false) or 1 (true). These values have the following meanings:

<table>
<thead>
<tr>
<th>Value of Process Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Performs default processing</td>
</tr>
<tr>
<td>1</td>
<td>Bypasses default Maximo processing</td>
</tr>
</tbody>
</table>

Default Collaboration Switches

Maximo creates three default collaboration switches, with different combinations of system ID values, for each process control ID. Authorized users can create additional switches as needed.

The default switches use various combinations of the following values in the System ID 1 and System ID 2 fields.

System ID Values

<table>
<thead>
<tr>
<th>System ID Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>THISMX</td>
<td>The Maximo system identified in MAXVARS.MXSYSID*</td>
</tr>
<tr>
<td>EXT</td>
<td>Any system other than the one identified in MAXVARS.MXSYSID</td>
</tr>
</tbody>
</table>

* The collaboration switches do not use the actual value in MXSYSID.

Authorized users can update only the process control value in the default collaboration switches.

CAUTION

Deleting a default collaboration switch or modifying any field other than the process control value can result in system failure.
Retrieving a Collaboration Switch

Each process control ID has at least three related collaboration switches (the defaults and any that the user adds). Maximo uses the following logic to determine which system ID values to set when retrieving a collaboration switch from the MXCOLLAB table:

1. Maximo uses the rules for deriving the System ID 1 and System ID 2 values for the process control ID in question.

   Example: For process control ID PRDEL, System ID 1 is the literal THISMX and System ID 2 is the system that owns the PR.

2. If the value in System ID 1 is now blank, null, or equal to the value in the MXSYSID row of the Maximo MAXVARS table, Maximo uses THISMX for System ID 1.

3. If the value in System ID 2 is now blank, null, or is equal to the value in the MXSYSID row of the Maximo MAXVARS table, and the process control ID is not PRPAB, Maximo uses THISMX for System ID 2.

4. If the process control ID is PRPAB, the value in System ID 2 will be null after step 1, and Maximo uses EXT for System ID 2. (if the blanket PO does not exist).

5. If both System ID 1 and System ID 2 now equal THISMX, Maximo performs the default Maximo processing (that is, it acts as if it retrieved an MXCOLLAB record with a process control value of 0).

   If both System ID 1 and System ID 2 equal THISMX, ignore the remaining steps.

6. Maximo tries to find a record with the modified key in the MXCOLLAB table.

   If the record exists, Maximo uses the record’s process control value to determine whether or not to bypass Maximo processing.

   If the record exists, ignore the remaining steps.

7. If the record does not exist, Maximo modifies the key as follows:

   If System ID 1 now equals THISMX and System ID 2 does not equal THISMX, Maximo uses EXT as the System ID 2 value.

   If System ID 1 value does not equal THISMX and System ID 2 equals THISMX, Maximo uses EXT as the System ID 1 value.

8. Maximo tries to find a record with the modified key in the MXCOLLAB table.

   If the record exists, Maximo uses the record’s process control value to determine whether or not to bypass Maximo processing.

   If the record exists, ignore the remaining step.
Maximo uses EXT for both System ID 1 and System ID 2.

Maximo finds the record with the modified key in the MXCOLLAB table.

This record always exists, because every process control value has a default collaboration switch with both system IDs equal to EXT.

Maximo uses the record’s process control value to determine whether or not to bypass Maximo processing.

Viewing Collaboration Switches

You can use any database tool to display the values in the MXCOLLAB table. You can also create a report with the report-writing tools available in Maximo.

To display the collaboration switches for a single process control ID, use the following SQL query:

```sql
select pcid, owner1sysid, owner2sysid, pcvalue
from mxcollab
where pcid = 'PCID'
order by pcid, owner1sysid, owner2sysid;
```

To display all collaboration switches, use the following SQL query:

```sql
select pcid, owner1sysid, owner2sysid, pcvalue
from mxcollab
order by pcid, owner1sysid, owner2sysid;
```

To display a short description of the process control IDs, use the following SQL query:

```sql
select * from mxcollabref order by pcid;
```
Modifying a Collaboration Switch

Authorized users can use any database tool to modify the process control value of a collaboration switch.

**CAUTION**
Do not change the value of PCID, OWNER1SYSID, or OWNER2SYSID on existing collaboration switches.

To change the process control value in a collaboration switch, use the following SQL statement:

```sql
update mxcollab
set pcvalue = PCVALUE
where pcid = 'PCID'
and owner1sysid = 'OWNER1SYSID'
and owner2sysid = 'OWNER2SYSID';
```

**NOTE**
The values in the MXCOLLAB table are case-sensitive.

Adding a Collaboration Switch

Authorized users can add new collaboration switches to the MXCOLLAB table. New switches must use an existing process control ID, but they can use new system IDs.

**NOTE**
Only the default collaboration switches can use the values THISMX and EXT in the system ID fields.

To add a collaboration switch, use the following SQL statement:

```sql
insert into mxcollab
(pcid, owner1sysid, owner2sysid, pcvalue)
values ('PCID', 'OWNER1SYSID', 'OWNER2SYSID', PCVALUE);
```

**Example**

You the Maximo Integration to integrate Maximo with an Oracle Financials® system and other systems. When Oracle Financials issues Maximo-owned inventory, you want Maximo to accept the issue and update inventory balances and costs. However, when other systems issue Maximo-owned inventory, you want to accept the issue, but you do not want to update inventory balances or costs.

Before you modify the MXCOLLAB table to reflect these conditions, the INV collaboration switches have the following values.
**INV Collaboration Switch Default Values**

<table>
<thead>
<tr>
<th>Process Control ID</th>
<th>System ID 1</th>
<th>System ID 2</th>
<th>Process Control Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>INV</td>
<td>THISMX</td>
<td>EXT</td>
<td>1</td>
</tr>
<tr>
<td>INV</td>
<td>EXT</td>
<td>THISMX</td>
<td>0</td>
</tr>
<tr>
<td>INV</td>
<td>EXT</td>
<td>EXT</td>
<td>1</td>
</tr>
</tbody>
</table>

**To accomplish the task, complete the following steps:**

1. Change the value of the INV/EXT/THISMX collaboration switch to 1 to bypass normal update processing.

   To update the collaboration switch, use the following SQL statement:

   ```sql
   update mxcollab
   set pcvalue = 1
   where pcid = 'INV'
   and owner1sysid = 'EXT'
   and owner2sysid = 'THISMX';
   ```

2. Select a system ID to represent Oracle Financials.

   This example uses ORC as the system ID.

3. Add a new collaboration switch—INV/ORC/THISMX—to the MXCOLLAB table.

   The single SQL statement following step 5 performs the processing for steps 3, 4, and 5.

4. Set the new collaboration switch’s process control value to 0.

   This value directs Maximo to perform the normal balance and cost updates when it receives issues from Oracle Financials.

5. Set the OWNERSYSID attribute on the inbound transaction to the system ID you chose in step 2—in this case, ORC.

   **NOTE** If OWNERSYSID is blank, the value in the DEFEXTSYS integration control is used.

   To perform the processing described in steps 3, 4 and 5, use the following SQL statement:

   ```sql
   insert into mxcollab
   (pcid, owner1sysid, owner2sysid, pcvalue)
   values ('INV', 'ORC', 'THISMX', 0);
   ```

   After you perform this procedure, the INV collaboration switches will have the following values.
Adding a Collaboration Switch

INV Collaboration Switch Updated Values

<table>
<thead>
<tr>
<th>Process Control ID</th>
<th>System ID 1</th>
<th>System ID 2</th>
<th>Process Control Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>INV</td>
<td>THISMX</td>
<td>EXT</td>
<td>1</td>
</tr>
<tr>
<td>INV</td>
<td>EXT</td>
<td>THISMX</td>
<td>1</td>
</tr>
<tr>
<td>INV</td>
<td>EXT</td>
<td>EXT</td>
<td>1</td>
</tr>
<tr>
<td>INV</td>
<td>ORC</td>
<td>THISMX</td>
<td>0</td>
</tr>
</tbody>
</table>

Example

By setting the value of the ISUIN integration control to 1, you tell Maximo to accept issue transactions from an external system. The INV collaboration switch controls the update of inventory balance and cost related to issues. You can adjust the setting of this switch, if necessary, so that Maximo bypasses that update process.

The INV/EXT/THISMX collaboration switch controls the processing of inventory (Process Control ID = INV) that is issued in the external system (System ID 1 = EXT) and owned by Maximo (System ID 2 = THISMX).

If the value of the INV/EXT/THISMX collaboration switch were 0, Maximo would update the inventory balance and cost. This is the default processing.

If the value of the INV/EXT/THISMX collaboration switch were 1, Maximo would bypass the default processing and not update the inventory balance and cost.

In this example, ISUIN accepts any issues into Maximo. The INV/EXT/THISMX collaboration switch then tells the inventory business component how to process a specific type of issue.
## Inventory Collaboration Switches

<table>
<thead>
<tr>
<th>Process Control ID</th>
<th>Description</th>
<th>Value and Action</th>
<th>Derivation of System ID 1</th>
<th>Derivation of System ID 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>INV</td>
<td>Update inventory. Used when creating issues, returns, or miscellaneous receipts or adjustments. Inventory must exist in this system.</td>
<td>0: Update inventory. 1: Do not update inventory.</td>
<td>INVTRANS or MATUSETRANS</td>
<td>INVENTORY</td>
</tr>
<tr>
<td>INVDEL</td>
<td>Delete inventory. Used when deleting externally owned inventory. Caution: If value is 1, the item-storeroom will still exist on related open PRs, P0s, RFQs, work orders, and so on. This might result in problems receiving/approving these lines.</td>
<td>0: Delete item if it passes normal Maximo validations. 1: Delete inventory without any validations and delete INVBALANCES record for the item.</td>
<td>“THISMX”</td>
<td>INVENTORY</td>
</tr>
<tr>
<td>INVISS</td>
<td>Enter item issues. Used when issuing material.</td>
<td>0: Allow material issues for the inventory. 1: Do not allow material issues for the inventory.</td>
<td>MATUSETRANS</td>
<td>INVENTORY</td>
</tr>
<tr>
<td>INVISSR</td>
<td>Enter item issue returns. Used when returning material.</td>
<td>0: Allow material returns. 1: Do not allow issue returns for the material.</td>
<td>MATUSETRANS</td>
<td>INVENTORY</td>
</tr>
<tr>
<td>INVISSWO</td>
<td>Update work order actual cost, equipment INVCOSTs. Used when processing issues or returns. Meant to handle Maximo to Maximo cases where these updates will be done separately.</td>
<td>0: Update work order actual material cost, equipment INVCOST. 1: Do not update work order actual material cost, equipment INVCOST.</td>
<td>MATUSETRANS</td>
<td>WORKORDER</td>
</tr>
</tbody>
</table>
## Inventory Collaboration Switches

<table>
<thead>
<tr>
<th>Process Control ID</th>
<th>Description</th>
<th>Value and Action</th>
<th>Derivation of System ID 1</th>
<th>Derivation of System ID 2</th>
</tr>
</thead>
</table>
| INVPHY             | Enter external physical counts. Used when creating physical counts.         | 0  
|                    |                                                                             | Allow physical count for the inventory.                                         | INVTRANS                  | INVENTORY                 |
|                    |                                                                             | 1  
|                    |                                                                             | Do not allow physical count for the inventory.                                  |                           |                           |
| INVTR              | Update the From storeroom on a transfer or the receipt of internal PO.      | 0  
|                    |                                                                             | Update INVBALANCES in the From storeroom.                                       | MATRECTRANS               | LOCATIONS (storeroom for transfer; vendor for internal PO) |
|                    |                                                                             | 1  
|                    |                                                                             | Do not update INVBALANCES in the From storeroom.                                |                           |                           |
| ITMDEL             | Delete items. Used when deleting items that this system does not own.       | 0  
|                    |                                                                             | Delete item if it passes normal Maximo validations.                             | “THISMX”                  | ITEM                      |
|                    |                                                                             | 1  
|                    |                                                                             | Delete item without any validation. Also delete INVENTORY, INVBALANCES, and INNVENDOR records for the item. |                           |                           |
|                    | Caution: If value is 1, the item will still exist on related open PRs, POs, RFQs, work orders, and so on. This might result in problems receiving/approving these lines. |                                                                                   |                           |                           |
# Invoice Collaboration Switches

<table>
<thead>
<tr>
<th>Process Control ID</th>
<th>Description</th>
<th>Value and Action</th>
<th>Derivation of System ID 1</th>
<th>Derivation of System ID 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>IVILC</td>
<td>Update inventory last cost.</td>
<td>0 Update inventory last cost.</td>
<td>INVOICE</td>
<td>INVENTORY</td>
</tr>
<tr>
<td></td>
<td>Used when approving invoices.</td>
<td>1 Do not update inventory last cost.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IVINV</td>
<td>Update inventory average cost.</td>
<td>0 Update inventory average cost.</td>
<td>INVOICE</td>
<td>INVENTORY</td>
</tr>
<tr>
<td></td>
<td>Used when approving invoices.</td>
<td>1 Do not update inventory average cost.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IVMATCH</td>
<td>Use and validate invoice match.</td>
<td>0 Validate match.</td>
<td>“THISMX”</td>
<td>INVOICE</td>
</tr>
<tr>
<td></td>
<td>Used when approving invoices.</td>
<td>1 Do not validate any match provided.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>If set to 1, IVPO should also be set to 1.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IVPO</td>
<td>Update POs and receipts.</td>
<td>0 Update PO status and receipts.</td>
<td>INVOICE</td>
<td>PO</td>
</tr>
<tr>
<td></td>
<td>Used when approving invoices.</td>
<td>1 Do not update PO status or receipts.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IVPRO</td>
<td>Check and prorate differences between invoice headers and lines.</td>
<td>0 Prorate the difference between the header and the line total.</td>
<td>“THISMX”</td>
<td>INVOICE</td>
</tr>
<tr>
<td></td>
<td>Used when approving invoices.</td>
<td>1 Do not prorate the difference between the header and line total.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IVRC</td>
<td>Create service receipts for invoice lines without a PO reference.</td>
<td>0 Generate a service receipts for the invoice lines that do not have a PO reference.</td>
<td>“THISMX”</td>
<td>INVOICE</td>
</tr>
<tr>
<td></td>
<td>Used when approving invoices that contain a line without a PO reference.</td>
<td>1 Do not generate a service receipt for the invoice lines that do not have a PO reference.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Invoice Collaboration Switches

<table>
<thead>
<tr>
<th>Process Control ID</th>
<th>Description</th>
<th>Value and Action</th>
<th>Derivation of System ID 1</th>
<th>Derivation of System ID 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>IVRCY</td>
<td>Create service receipts for invoice lines with a PO reference and RECEIPTREQD = N.</td>
<td>0 Generate a service receipt for the invoice line.</td>
<td>INVOICE</td>
<td>PO</td>
</tr>
<tr>
<td></td>
<td>Used when approving invoices containing a line with a PO reference, when the corresponding POLINE is a service and RECEIPTREQD = N.</td>
<td>1 Do not generate a service receipt for the invoice line.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IVTOL</td>
<td>Perform invoice tolerance checking validation.</td>
<td>0 Perform all tolerance checks on invoice.</td>
<td>“THISMX”</td>
<td>INVOICE</td>
</tr>
<tr>
<td></td>
<td>Used when approving invoices.</td>
<td>1 Do not perform tolerance checks on invoice.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IVVLC</td>
<td>Update vendor last cost.</td>
<td>0 Update vendor last cost.</td>
<td>INVOICE</td>
<td>INVVENDOR</td>
</tr>
<tr>
<td></td>
<td>Used when approving invoices.</td>
<td>1 Do not update vendor last cost.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IVWO</td>
<td>Update work orders.</td>
<td>0 Update work order.</td>
<td>INVOICE</td>
<td>WORKORDER</td>
</tr>
<tr>
<td></td>
<td>Used when approving invoice.</td>
<td>1 Do not update work order.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Labor Transaction Collaboration Switches

<table>
<thead>
<tr>
<th>Process Control ID</th>
<th>Description</th>
<th>Value and Action</th>
<th>Derivation of System ID 1</th>
<th>Derivation of System ID 2</th>
</tr>
</thead>
</table>
| LTSRC              | Generate service receipts for POs.                  | 0  Allow setting value of LABTRANS.GENAPPRSERVRECEIPT to Y; configurable in Maximo Application Setup.  
                        | Used when creating labor transactions or changing status. PO must exist in this system. | 1  Leave value of LABTRANS.GENAPPRSERVRECEIPT as N.                                      | LABTRANS                  | PO                        |

## Purchase Order Collaboration Switches

<table>
<thead>
<tr>
<th>Process Control ID</th>
<th>Description</th>
<th>Value and Action</th>
<th>Derivation of System ID 1</th>
<th>Derivation of System ID 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>PODEL</td>
<td>Delete POs.</td>
<td>0  Do not delete PO.</td>
<td>“THISMX”</td>
<td>PO</td>
</tr>
</tbody>
</table>
<pre><code>                    | Used when deleting POs. Use only when deleting then subsequently re-adding a PO due to changes in the PO. | 1  Delete PO and PRLINEs; do not delete POSTATUS.                                 |                           |                           |
                    | If any PRLINES contain a reference to the PO, clear them. If necessary, reopen the PR. When the PO is re-added, the PRLINES will be established again. |                                                                  |                           |                           |
</code></pre>
<p>| POINV              | Do not allow unreferenced external inventory for internal POs. | 0  If the item-vendor combination not in INVENTORY table, error.                  | PO                                                                                         | LOCATIONS, where vendor is the storeroom |</p>
<table>
<thead>
<tr>
<th>Process Control ID</th>
<th>Description</th>
<th>Value and Action</th>
<th>Derivation of System ID 1</th>
<th>Derivation of System ID 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>POIVM</td>
<td>Create inventory vendor information for inventory. Used when approving POs.</td>
<td>0 Update or create INVVENDOR record.</td>
<td>PO</td>
<td>ITEM of POLINE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 Do not create INVVENDOR record.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>POPR</td>
<td>Update status of PRs. Used when copying PR lines to POs, creating POs from PRs, reopening PRs.</td>
<td>0 Change status of PR (auto close based on MAXVAR setting) or reopen when POLINE containing PR reference is deleted, or other instances of reopen.</td>
<td>PO</td>
<td>PR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 Do not change PR status.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>POREL</td>
<td>Create releases for blanket POs. Used when approving PR and the PR lines contain a blanket references, and when a release is created directly from a PO without a PR.</td>
<td>0 Generate PO release. (If PRLINE.AGREEMENTPONUM not in PO, do not generate PO release).</td>
<td>PR</td>
<td>PO of the blanket</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 Do not regenerate PO releases.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PORES</td>
<td>Process material reservations. Used when changing the status of internal POs.</td>
<td>0 Generate inventory reservations. If item-vendor combination (where vendor is internal storeroom) not in INVENTORY table, do not generate PO reservations. This might happen if POINV is 0.</td>
<td>PO</td>
<td>INVENTORY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 Do not generate inventory reservations.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Purchase Requisition Collaboration Switches

<table>
<thead>
<tr>
<th>Process Control ID</th>
<th>Description</th>
<th>Value and Action</th>
<th>Derivation of System ID 1</th>
<th>Derivation of System ID 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRDEL</td>
<td>Delete PRs. Used when deleting PRs. Use only when deleting then subsequently re-adding a PR due to changes in the PR.</td>
<td>0 Do not delete PR. 1 Delete the PR and PRLINES; do not delete PRSTATUS.</td>
<td>“THISMX”</td>
<td>PR</td>
</tr>
<tr>
<td></td>
<td>If WPMATERIAL or MRLINE contain references to the PR, clear them. They will be reestablished when you read PR.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRINV</td>
<td>Do not allow unreferenced external inventory on internal PRs. Used when storerooms are maintained in an external system. Items are in ITEM master in Maximo; storeroom is defined as a LOCATION; INVENTORY is not defined for item-storeroom combination because it is not owned by Maximo. The owner of the PR is the MXSYSID of the system that creates the PR. Validation occurs when an item-storeroom (INVENTORY) is validated on the PRLINE. The OWNERSYSID of the storeroom is compared with the OWNERSYSID of the PR, and the flag determines if the combination is allowed.</td>
<td>0 If the item-vendor combination is not in INVENTORY table, error. 1 If the item-vendor combination (vendor is the internal storeroom) is not in INVENTORY table, where PR.VENDOR = LOCATIONS.LOCATION, ignore error. LOCATIONS must exist; that is, pass standard validation for the location.</td>
<td>PR</td>
<td>LOCATIONS, where vendor is the internal storeroom</td>
</tr>
</tbody>
</table>
### Purchase Requisition Collaboration Switches

<table>
<thead>
<tr>
<th>Process Control ID</th>
<th>Description</th>
<th>Value and Action</th>
<th>Derivation of System ID 1</th>
<th>Derivation of System ID 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRPAB</td>
<td>Do not allow unreferenced external purchase agreements / blankets.</td>
<td>0 If PRLINE.AGREEMENTPONUM is not in PO, error.</td>
<td>PR</td>
<td>“EXT”</td>
</tr>
<tr>
<td></td>
<td>Used when adding or updating PR lines and changing the status of PRs.</td>
<td>1 If PRLINE.AGREEMENTPONUM is not in PO, ignore error.</td>
<td></td>
<td>Normally this would be from PO of the blanket, but in this case the blanket PO does not exist</td>
</tr>
</tbody>
</table>
## Receipt Collaboration Switches

<table>
<thead>
<tr>
<th>Process Control ID</th>
<th>Description</th>
<th>Value and Action</th>
<th>Derivation of System ID 1</th>
<th>Derivation of System ID 2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>RC</td>
<td>Enter PO receipts.</td>
<td>0 Allow receiving against the PO.</td>
<td>MATRECTRANS or SERVRECTRANS</td>
<td>PO</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Used when creating receipts.</td>
<td>1 Do not allow receiving against the PO.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RCILC</td>
<td>Update inventory last cost.</td>
<td>0 Update inventory last cost.</td>
<td>MATRECTRANS</td>
<td>INVENTORY</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Used when approving receipts.</td>
<td>1 Do not update inventory last cost.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RCINV</td>
<td>Update inventory.</td>
<td>0 Update inventory if it exists.</td>
<td>MATRECTRANS</td>
<td>INVENTORY</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Used when receiving, or approving receipts.</td>
<td>1 Do not update inventory.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RCIV</td>
<td>Generate invoices for PO receipts.</td>
<td>0 Generate invoice if value of PayOnReceipt is set.</td>
<td>MATRECTRANS or SERVRECTRANS</td>
<td>PO</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Used when approving receipts.</td>
<td>1 Do not generate invoice, even if value of PayOnReceipt is set.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RCPO</td>
<td>Update external PO.</td>
<td>0 Update PO.</td>
<td>MATRECTRANS or SERVRECTRAN</td>
<td>PO</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Used when approving receipt.</td>
<td>1 Do not update PO.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RCR</td>
<td>Enter PO receipt return.</td>
<td>0 Allow receipt returns for the PO.</td>
<td>MATRECTRANS or SERVRECTRAN</td>
<td>PO</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Used when creating receipt return.</td>
<td>1 Do not allow receipt returns for the PO.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RCVLC</td>
<td>Update vendor last cost.</td>
<td>0 Update vendor last cost.</td>
<td>MATRECTRANS</td>
<td>INVVENDOR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Used when approving receipt or receiving PO line.</td>
<td>1 Do not update vendor last cost.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RCWO</td>
<td>Update work orders.</td>
<td>0 Update work order.</td>
<td>MATRECTRANS or SERVRECTRAN</td>
<td>WORKORDER</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Used when approving receipts.</td>
<td>1 Do not update work order.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Work Order Collaboration Switches

<table>
<thead>
<tr>
<th>Process Control ID</th>
<th>Description</th>
<th>Value and Action</th>
<th>Derivation of System ID 1</th>
<th>Derivation of System ID 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORES</td>
<td>Process material reservations.</td>
<td>0 Generate inventory reservation.</td>
<td>WORKORDER</td>
<td>INVENTORY</td>
</tr>
<tr>
<td></td>
<td>Used when changing the status of a work order.</td>
<td>1 Do not generate inventory reservation.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Inventory must exist in this system.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
This appendix explains how to create the JMS queues and perform other configuration activities on your WebLogic and WebSphere application servers.

**CAUTION** Perform these activities only if you did not already do them during Maximo installation.

It includes the following sections:

- Creating the JMS Queues in WebLogic
- Configuring the MEA Connection Factory in WebLogic
- Configuring the Redelivery Delay in WebLogic
- Configuring Maximum JMS Queue Space in WebLogic
- Creating the JMS Bus (Queues) in WebSphere
- Creating the MEA Connection Factory and Queues in WebSphere

After you perform the activities described in this appendix, continue from where you left off in Chapter 6, "Basic Configuration."
Creating the JMS Queues in WebLogic

To configure the JMS queues, complete the following steps:

1. Access the Configuration Wizard by navigating to:
   Start > Programs > BEA WebLogic > Configuration Wizard
   The Create or Extend a Configuration screen appears.

2. Select **Extend an existing WebLogic configuration**, then click **Next**.

---

Create or Extend a Configuration Screen

- **Create a new WebLogic configuration**
  - Start here to create a WebLogic configuration in your projects directory.

- **Extend an existing WebLogic configuration**
  - Start here to extend an existing WebLogic configuration.
  - Use this option to add applications and services, including Database access (JDBC) and Messaging (JMS).
  - This option also enables you to extend functionality by enabling WebLogic Workshop.
Creating the JMS Queues in WebLogic

The Choose a WebLogic Configuration Directory screen appears.

**Choose a WebLogic Configuration Directory Screen**

3. Select the domain where Maximo is installed, then click **Next**.

The Select a Configuration Extension Template screen appears.

**Select a Configuration Extension Template Screen**
Creating the JMS Queues in WebLogic

4. Select the DefaultWebApp used for Maximo configuration, then click Next.

The Database (JDBC) Options screen appears.

**Database (JDBC) Options Screen**

5. Select No, then click Next.

The Messaging (JMS) Options screen appears.

**Messaging (JMS) Options Screen**
6 Select **Yes**, then click **Next**.

The Configure JMS Connection Factories screen appears.

**Configure JMS Connection Factories Screen**

7 Click **Add**.

8 Enter or select the following values. Enter the values in the Name and JNDI name columns exactly as shown (case-sensitive).

<table>
<thead>
<tr>
<th>Column</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>MEA connectionfactory</td>
</tr>
<tr>
<td>JNDI name</td>
<td>jms/mro/int/qcf/intqcf</td>
</tr>
<tr>
<td>Default delivery mode</td>
<td>Persistent</td>
</tr>
<tr>
<td>Default priority</td>
<td>4</td>
</tr>
<tr>
<td>Acknowledge policy</td>
<td>All</td>
</tr>
</tbody>
</table>
9 Click Next.

The Configure JMS Destination Key(s) screen appears.

**Configure JMS Destination Key(s) Screen**

10 Click Next.

The Configure JMS Template(s) screen appears.

**Configure JMS Template(s) Screen**
11 Click Next.

The Configure JMS File Stores screen appears.

**Configure JMS File Stores Screen**

![Configure JMS File Stores Screen]

12 Click Add.

13 Enter or select the following values. Enter the value in the Name column exactly as shown (case-sensitive).

   **NOTE** The directory names shown in the figure (c:\bea\jmsstore) are arbitrary. The directory can be any directory located in your file system. Create a folder named jmsstore in the root of the BEA installation location before continuing with this step. Do not use c:\temp.

<table>
<thead>
<tr>
<th>Column</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>mxintsqinfile</td>
</tr>
<tr>
<td>Directory</td>
<td>c:\bea\jmsstore</td>
</tr>
<tr>
<td></td>
<td>or</td>
</tr>
<tr>
<td></td>
<td>any existing directory located in your file system (for storing persistent messages)</td>
</tr>
<tr>
<td>Synchronous write policy</td>
<td>Disabled</td>
</tr>
</tbody>
</table>

14 Click Add.
15 Enter or select the following values. Enter the value in the Name column exactly as shown (case-sensitive).

<table>
<thead>
<tr>
<th>Column</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>mxintsqoutfile</td>
</tr>
<tr>
<td>Directory</td>
<td>c:\bea\jmsstore</td>
</tr>
<tr>
<td></td>
<td>or</td>
</tr>
<tr>
<td></td>
<td>any existing directory located in your file system (for storing persistent messages)</td>
</tr>
<tr>
<td>Synchronous write policy</td>
<td>Disabled</td>
</tr>
</tbody>
</table>

16 Click Add.

17 Enter or select the following values. Enter the value in the Name column exactly as shown (case-sensitive).

<table>
<thead>
<tr>
<th>Column</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>mxintcqinfile</td>
</tr>
<tr>
<td>Directory</td>
<td>c:\bea\jmsstore</td>
</tr>
<tr>
<td></td>
<td>or</td>
</tr>
<tr>
<td></td>
<td>any existing directory located in your file system (for storing persistent messages)</td>
</tr>
<tr>
<td>Synchronous write policy</td>
<td>Disabled</td>
</tr>
</tbody>
</table>
18 Click Next.

The Configure JMS Servers screen appears.

**Configure JMS Servers Screen**

19 Click Add.

20 Enter or select the following values. Enter the value in the Name column exactly as shown (case-sensitive).

<table>
<thead>
<tr>
<th>Column</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>mxintsqinserver</td>
</tr>
<tr>
<td>Store</td>
<td>mxintsqinfile</td>
</tr>
<tr>
<td>Paging store</td>
<td>Unspecified</td>
</tr>
<tr>
<td>Temporary template</td>
<td>Unspecified</td>
</tr>
<tr>
<td>Expiration scan interval</td>
<td>30</td>
</tr>
</tbody>
</table>

21 Click Add.
Creating the JMS Queues in WebLogic

22 Enter or select the following values. Enter the value in the Name column exactly as shown (case-sensitive).

<table>
<thead>
<tr>
<th>Column</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>mxintsqoutserver</td>
</tr>
<tr>
<td>Store</td>
<td>mxintsqoutfile</td>
</tr>
<tr>
<td>Paging store</td>
<td>Unspecified</td>
</tr>
<tr>
<td>Temporary template</td>
<td>Unspecified</td>
</tr>
<tr>
<td>Expiration scan interval</td>
<td>30</td>
</tr>
</tbody>
</table>

23 Click Add.

24 Enter or select the following values. Enter the value in the Name column exactly as shown (case-sensitive).

<table>
<thead>
<tr>
<th>Column</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>mxintcqinserver</td>
</tr>
<tr>
<td>Store</td>
<td>mxintcqinfile</td>
</tr>
<tr>
<td>Paging store</td>
<td>Unspecified</td>
</tr>
<tr>
<td>Temporary template</td>
<td>Unspecified</td>
</tr>
<tr>
<td>Expiration scan interval</td>
<td>30</td>
</tr>
</tbody>
</table>

25 Click Next.

The Configure JMS Servers to WebLogic Servers screen appears.

Configure JMS Servers to WebLogic Servers Screen
26 Select all the JMS servers that appear in the JMS Server column, and click the right arrow button to assign them to the WebLogic server.

27 Click **Next**.

The Configure JMS Topics screen appears.

**Configure JMS Topics Screen**

28 Click **Next**.

The Configure JMS Queues screen appears.
29 Select the mqintsqinserver tab.

**Configure JMS Queues Screen (mxintsqinserver Tab)**

30 Click **Add**.

31 Enter or select the following values. Enter the values in the Name and JNDI name columns exactly as shown (case-sensitive).

<table>
<thead>
<tr>
<th>Column</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>mxintsqin</td>
</tr>
<tr>
<td>JNDI name</td>
<td>jms/mro/int/queues/sqin</td>
</tr>
<tr>
<td>Store enabled</td>
<td>true</td>
</tr>
<tr>
<td>Template</td>
<td>Unspecified</td>
</tr>
</tbody>
</table>
32 Select the mqintsqoutserver tab.

*Configure JMS Queues Screen (mxintsqoutserver Tab)*

33 Click **Add**.

34 Enter or select the following values. Enter the values in the Name and JNDI name columns exactly as shown (case-sensitive).

<table>
<thead>
<tr>
<th>Column</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>mxintsqout</td>
</tr>
<tr>
<td>JNDI name</td>
<td>jms/mro/int/queues/sqout</td>
</tr>
<tr>
<td>Store enabled</td>
<td>true</td>
</tr>
<tr>
<td>Template</td>
<td>Unspecified</td>
</tr>
</tbody>
</table>
35 Select the mqintcqinserver tab.

**Configure JMS Queues Screen (mxintcqinserver Tab)**

![Configure JMS Queues Screen](image)

36 Click **Add**.

37 Enter or select the following values. Enter the values in the Name and JNDI name columns exactly as shown (case-sensitive).

<table>
<thead>
<tr>
<th>Column</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>mxintcqin</td>
</tr>
<tr>
<td>JNDI name</td>
<td>jms/mro/int/queues/cqin</td>
</tr>
<tr>
<td>Store enabled</td>
<td>true</td>
</tr>
<tr>
<td>Template</td>
<td>Unspecified</td>
</tr>
</tbody>
</table>
38 Click Next.

The Applications and Services Targeting Options screen appears.

**Applications and Services Targeting Options Screen**

Do you want to target servers and clusters onto which Applications, JMS component services, JDBC component services, and other services are deployed?

You can specify the servers and clusters onto which you want to deploy Applications, Messaging (JMS) component services, Database (JDBC) component services, and other services (Startup/Shutdown Class, etc.).

If you want to skip this section, select "No" and click "Next." This wizard uses settings for your applications and services targeting that are identical to the settings in the configuration source that you selected earlier.

If you want to customize these settings, select "Yes" and click "Next" to go through the configuration setup.

39 Select Yes, then click Next.

The Target Applications to Servers or Clusters screen appears.

**Target Applications to Servers or Clusters Screen**
40 Click Select All, then click Next.

The Target Services to Servers or Clusters screen appears.

**Target Services to Servers or Clusters Screen**

41 Click Select All, then click Next.

The Security Configuration Options screen appears.

**Security Configuration Options Screen**
42 Select No, then click Next.

The Create WebLogic Configuration screen appears.

Create WebLogic Configuration Screen

43 Select the WebLogic domain (the default is mydomain), then click Import.

The Creating Configuration screen appears.

Creating Configuration Screen
Creating the JMS Queues in WebLogic

44 When the configuration is complete, click **Done**.
Configuring the MEA Connection Factory in WebLogic

After configuring the JMS queue, configure the messages maximum and enable the connection factory. If you do not enable the connection factory, you will receive warning messages.

To configure the connection factory, complete the following steps:

1. Start the WebLogic server.
2. Log on to the WebLogic console.
3. In the WebLogic server console, navigate to <WebLogic domain>/Services/JMS/Connection Factories/MEA connectionfactory.
4. Select the General subtab on the Configuration tab.
5. Enter –1 in the Messages Maximum field.

Messages Maximum Field

6. Select the Transactions subtab on the Configuration tab.
7 On the Transactions subtab of the Configuration tab, select the **XA Connection Factory Enabled** check box.

**XA Connection Factory Enabled Check Box**

8 Click **Apply**.

**NOTE** The preceding changes will take effect the next time you restart the WebLogic server.
Configuring the Redelivery Delay in WebLogic

Redelivery delay is the time that elapses between a message erroring out and being reprocessed by the queue. The message is not visible in the queue for the period of time specified for redelivery delay. This delay improves performance by processing other messages instead of immediately processing the error message. The default delay is 30 seconds.

**NOTE** This redelivery delay applies to the continuous inbound queue only.

To override the default redelivery delay, complete the following steps:

1. In the WebLogic server console, navigate to `<WebLogic domain>/Services/JMS/Servers/mxintcqinserver/Destination/mxintcqin`.

2. On the Redelivery subtab of the Configuration tab, enter the new redelivery delay value in the **Redelivery Delay Override** field.

3. Click **Apply**.
Configuring Maximum JMS Queue Space in WebLogic

The Bytes Maximum value indicates the maximum space the JMS queues can occupy. If this field does not contain a value, the only limit is the available free heap size. Exceeding the limit results in Out Of Memory errors and the suspension of processing. To avoid this, set the Bytes Maximum property. The optimal setting is 42-43% of the maximum heap size for the server.

**NOTE**

This limit is the limit for the sum of all the JMS queues in a Weblogic server. If a Weblogic server hosts all three default queues, in three JMS servers, Bytes Maximum is distributed among the three queues, based on their loads.

To set the Bytes Maximum value, complete the following steps:

1. In the WebLogic server console, navigate to `<WebLogic domain>/Services/JMS/Servers/mxintcqinserver/Destination/mxintcqin`.
2. Select the Thresholds & Quotas General subtab on the Configuration tab.
3. Enter the maximum JMS queue space in the **Bytes Maximum** field.

**Bytes Maximum Field**

If you enter a Bytes Maximum value, disable the Flow Control Enabled property.

To disable the Flow Control Enabled property, complete the following steps:

1. In the WebLogic server console, navigate to `<WebLogic domain>/Services/JMS/Connection Factories/MEA connectionfactory`.
2. Select the Flow Control subtab on the Configuration tab.
3 Clear the **Flow Control Enabled** check box.

**Flow Control Enabled Check Box**

With flow control, you can enable a JMS server or destination to slow down message producers when it determines that it is becoming overloaded. Specifically, when a JMS server/destination exceeds its specified bytes or messages thresholds, it becomes armed and instructs producers to limit their message flow (messages per second). This page allows you to define the flow control configuration for this JMS connection factory.

**Flow Maximum:**

The maximum number of messages-per-second (between 0 and a positive 32-bit integer) allowed for a producer that is experiencing a threshold condition on the JMS server or queue/topic destination. When a producer is flow controlled, it will never be allowed to go faster than this number of messages per second.

**Flow Minimum:**

The minimum number of messages per-second (between 0 and a positive 32-bit integer) allowed for a producer that is experiencing a threshold condition. That is, WebLogic JMS will not further slow down a producer whose message flow limit is at its Flow Minimum.

**Flow Interval:**

The number of seconds (between 0 and a positive 32-bit integer) when a producer adjusts its flow from the Flow Maximum number of messages to the Flow Minimum amount, or vice versa.

**Flow Steps:**

The number of steps (between 1 and a positive 32-bit integer) used when a producer is adjusting its flow from the Flow Maximum amount of messages to the Flow Minimum amount, or vice versa.

**Flow Control Enabled**

Specifies whether flow control is enabled for a producer created using this JMS connection factory allows flow control.

**Send Timeout:**

4 Click **Apply**.

You can now continue from where you left off in Chapter 6, "Basic Configuration."
Creating the JMS Bus (Queues) in WebSphere

To configure the JMS queues, complete the following steps:

1. Start the WebSphere application server.

2. Launch Internet Explorer and open the WebSphere Administrative Console by typing the following URL:

   http://<machine_name>:<port_number>/ibm/console

   You might, for example, enter a URL similar to the following:

   http://localhost:9060/ibm/console

3. At the “Welcome, please enter your information” login screen, enter your User ID, then click Log in.

   This action opens the Welcome screen for the WebSphere Administrative Console.

4. Click Service Integration > Buses to open the Buses pane as shown below:

   **The Buses Pane**

   ![Buses Pane](image)

   **NOTE**

   A bus is a group of interconnected servers and clusters that have been added as members of the bus.

   A service integration bus supports applications using message-based and service-oriented architectures. A bus is a group of interconnected servers and clusters that have been added as members of the bus. Applications connect to a bus at one of the messaging engines associated with its bus members.

   **Preferences**

   - New
   - Delete

   **Select**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

   **Total**
5 Click **New** to open the Buses > New pane where you can add a new service integration bus.

**The Buses > New Pane**

<table>
<thead>
<tr>
<th>General Properties</th>
<th>Additional Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Bus members</td>
</tr>
<tr>
<td>Description</td>
<td>Messaging engines</td>
</tr>
<tr>
<td>UUID</td>
<td>Destination</td>
</tr>
<tr>
<td></td>
<td>Mediations</td>
</tr>
<tr>
<td></td>
<td>Foreign buses</td>
</tr>
<tr>
<td></td>
<td>Custom properties</td>
</tr>
</tbody>
</table>

6 To add a new service integration bus, enter the following information:

- **Text description of the new bus in the** **Name** **field, for example, meajmsbus.**

- **Deselect the Secure check box. If you leave this box checked, meajmsbus inherits the Global Security setting of the cell.**

- **Change the value of the High message threshold field to a minimum value of 500,000 messages.**

- **Accept all other default settings.**

If the number of messages awaiting processing exceeds the High Message Threshold you set, the application server will take action to limit the addition of new messages in the processing queues.

Depending on your message requirements, you may want to enter a higher message threshold value. You can determine an optimal message threshold setting by monitoring the messaging in/out queues and the impact of the message threshold setting on system performance. You might, for example, lower the threshold value if a higher value is degrading system performance.

**NOTE** If you decide to change the **High message threshold** setting after the initial configuration, you must open the Additional Properties menu in the
Creating the JMS Bus (Queues) in WebSphere

Administrative Console and change the threshold value for each child configuration.

7 Click **Apply** to create the new service integration bus. This step propagates the JMS bus setup to the cluster configuration. Confirm that build completed screen displays the following:

- Bus name, for example, meajmsbus.
- Auto-generated, unique ID (UUID), for example, 4BCAC78E15820FED.
- The Secure field is unchecked.
- High Message Threshold field has a minimum value of 500,000.

Adding Servers to the JMS Bus

To add servers to the JMS bus, complete the following steps:

1 From the WebSphere Administrative Console, click **Service Integration > Buses** to open the Buses pane as shown below:

*The Buses Pane - meajmsbus*

2 Click **meajmsbus** to open the Buses > meajms pane.

3 Under Additional Properties, click **Bus members**.
4 In the Buses > meajmsbus > Bus members pane, click **Add** to open the Add a new bus member pane as shown in the following figure:

**The Buses - Add a New Bus Member Pane**

![Add a new bus member pane](image)

5 Click the Server drop-down arrow, and select a server name, for example "QAERP7Node01.server1" to add to the bus.

6 Click **Next**.

7 Click **Finish**.

8 Click **Save**.

9 Click the "Synchronize changes with Nodes" box.
10 Click Save.

The Configuration > General Properties pane for the new bus member, server QAERP7, appears similar to the following:

**The Buses - General Properties for New Bus Member**

<table>
<thead>
<tr>
<th>General Properties</th>
<th>Message points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td></td>
</tr>
<tr>
<td>QAERP7Node01.server</td>
<td>Mediation points</td>
</tr>
<tr>
<td>UUID</td>
<td>Queue points</td>
</tr>
<tr>
<td>03572AFB5E880CC3B</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>Publication points</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Initial state</td>
<td>Additional Properties</td>
</tr>
<tr>
<td>Started</td>
<td>Data store</td>
</tr>
<tr>
<td></td>
<td>Service integration bus link</td>
</tr>
<tr>
<td>High message threshold</td>
<td>WebSphere MQ client links</td>
</tr>
<tr>
<td>500000</td>
<td>WebSphere MQ links</td>
</tr>
<tr>
<td>Target groups</td>
<td>Custom properties</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Bus name</td>
<td></td>
</tr>
<tr>
<td>maxmibus</td>
<td></td>
</tr>
<tr>
<td>Bus UUID</td>
<td></td>
</tr>
<tr>
<td>4BCC79E15820FEB</td>
<td></td>
</tr>
</tbody>
</table>
Creating the JMS Bus Destination for the Continuous Inbound (cqinbd) Queue

To add a logical address for the continuous inbound bus destination queue (cqinbd) within the JMS bus, complete the following steps:

1. From the WebSphere Administrative Console, click Service Integration > Buses to open the Buses pane.

2. Click meajmsbus to open the Buses > meajms pane.

3. Click Destinations under Additional Properties to open the Buses > meajmsbus > Destinations pane as shown in the following figure:

   **The Buses - Add a New Bus Destination**

   ![Buses - Add a New Bus Destination](image)

   **NOTE** A bus destination, for example cqinbd, is a virtual place within a service integration bus where applications can attach and exchange messages.

4. Click New to open the “Create new destination” pane.

**The Buses - Create Queue Destination Type**

![Create Queue Destination Type](image)
Creating the JMS Bus (Queues) in WebSphere

5 Leave Queue checked as the destination type, and click **Next** to open the “Create new queue” pane.

**The Buses - Specify Queue Attributes**

---

6 Enter “cqinbd” in the Identifier field and “Continuous Queue Inbound” in the Description field, then click **Next** to open the “Create a new queue for point-to-point messaging” pane.

**The Buses - Add the cqinbd Bus Destination Queue to a Bus Member**

---

7 Select the Bus Member pull-down and choose a server, such as “QAERP7...” as the bus member that will store and process messages for the cqinbd bus destination queue.
8. Click **Next** to open the “Confirm queue creation” pane.

*The Buses - Confirm the Creation of the cqinbd Bus Destination Queue*

9. Review your selections, then click **Finish** to complete the creation of the cqinbd bus destination queue.

10. Navigate the path Buses > meajmsbus > Destinations, then click **cqinbd** to open the configuration pane where you must make the following changes:

   ▼ Change the “Maximum failed deliveries” value to 1.

   This is the maximum number of times you want the system to process a failed messaging attempt before forwarding the message to the exception destination.

   ▼ Click None as the “Exception destination” value.

   The following figure is a partial snapshot of the cqinbd configuration pane.

*The Buses - Configuration Changes to the cqinbd Bus Destination Queue*
Creating the JMS Bus (Queues) in WebSphere

11 Click Apply.

12 Click Save.

13 Click the "Synchronize changes with Nodes" box.

14 Click Save.
Creating the JMS Bus Destination for the Sequential Inbound (sqinbd) Queue

To add a logical address for the sequential inbound bus destination queue (sqinbd) within the JMS bus, complete the following steps:

1. From the WebSphere Administrative Console, click Service Integration > Buses to open the Buses pane.
2. Click meajmsbus to open the Buses > meajms pane.
3. Click Destinations under Additional Properties to open the Buses > meajmsbus > Destinations pane.

**NOTE**
A bus destination, for example sqinbd, is a virtual place within a service integration bus where applications can attach and exchange messages.

4. Click New to open the “Create new destination” pane.
5. Leave Queue checked as the destination type, and click Next to open the “Create new queue” pane.
6. Enter “sqinbd” in the Identifier field and “Sequential Queue Inbound” in the Description field, then click Next to open the “Create a new queue for point-to-point messaging” pane.
7. Select the Bus Member pull-down and choose a server, such as “QAERP7...” as the bus member that will store and process messages for the sqinbd bus destination queue.
8. Click Next to open the “Confirm queue creation” pane.
9. Review your selections, then click Finish to complete the creation of the sqinbd bus destination queue.

10. Navigate the path Buses > meajmsbus > Destinations, then click sqinbd to open the configuration pane where you must make the following changes:

    ▼ Change the “Maximum failed deliveries” value to 1.
    
    This is the maximum number of times you want the system to process a failed messaging attempt before forwarding the message to the exception destination.

    ▼ Click None as the “Exception destination” value.

11. Click Apply.
12. Click Save.
13. Click the "Synchronize changes with Nodes" box.
14. Click Save.
Creating the JMS Bus Destination for the Sequential Outbound (sqoutbd) Queue

To add a logical address for the sequential outbound bus destination queue (sqoutbd) within the JMS bus, complete the following steps:

1. From the WebSphere Administrative Console, click Service Integration > Buses to open the Buses pane.
2. Click meajmsbus to open the Buses > meajms pane.
3. Click Destinations under Additional Properties to open the Buses > meajmsbus > Destinations pane.

**NOTE** A bus destination, for example sqoutbd, is a virtual place within a service integration bus where applications can attach and exchange messages.

4. Click New to open the “Create new destination” pane.
5. Leave Queue checked as the destination type, and click Next to open the “Create new queue” pane.
6. Enter “sqoutbd” in the Identifier field and “Sequential Queue Outbound” in the Description field, then click Next to open the “Create a new queue for point-to-point messaging” pane.
7. Select the Bus Member pull-down and choose a server, such as “QAERP7...” as the bus member that will store and process messages for the sqoutbd bus destination queue.
8. Click Next to open the “Confirm queue creation” pane.
9. Review your selections, then click Finish to complete the creation of the sqoutbd queue.
10. Navigate the path Buses > meajmsbus > Destinations, then click sqoutbd to open the configuration pane where you must make the following changes:

   ▼ Change the “Maximum failed deliveries” value to 1.
   This is the maximum number of times you want the system to process a failed messaging attempt before forwarding the message to the exception destination.

   ▼ Click None as the “Exception destination” value.
11. Click Apply.
12. Click Save.
13. Click the "Synchronize changes with Nodes" box.
14. Click Save.
Creating the MEA Connection Factory and Queues in WebSphere

You add a connection factory for creating connections to the associated JMS provider of point-to-point messaging queues.

1. From the WebSphere Administrative Console, click Resources > JMS Providers > Default Messaging to open the “Default messaging provider” pane.

2. Click Browse Nodes to open the Select a Node Scope pane as shown in the following figure:

**Default Messaging Provider - Select a Node Scope**

<table>
<thead>
<tr>
<th>Select</th>
<th>Node</th>
<th>Version</th>
<th>Cell</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>QAERP7CellManager01</td>
<td>6.0.0.2</td>
<td>QAERP7Cell01</td>
</tr>
<tr>
<td></td>
<td>QAERP7Node01</td>
<td>6.0.0.2</td>
<td>QAERP7Cell01</td>
</tr>
</tbody>
</table>

3. Click the Select box for "QAERP7Node01," then click OK.

4. Under Connection Factories, click JMS queue connection factory to open the following figure:

**JMS Providers - Add New JMS Queue Connection Factory**
Creating the MEA Connection Factory and Queues in WebSphere

5 Click **New** to complete the General Properties section for the new connection factory. At a minimum, enter the following information.

<table>
<thead>
<tr>
<th>Column</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>meaconfact</td>
</tr>
<tr>
<td>JNDI name</td>
<td>jms/mro/int/qcf/intqcf</td>
</tr>
<tr>
<td>Bus name</td>
<td>meajmsbus</td>
</tr>
</tbody>
</table>

**JMS Providers - Add General Properties for New Connection Factory**

6 Click **OK** to add “meaconfact” as a new JMS queue connection factory.

7 Click **Save**.

8 Click the “Synchronize changes with Nodes” box.

9 Click **Save**.

Creating the Continuous Inbound (cqin) JMS Queue

You must create a JMS queue (cqin) as the destination for continuous inbound point-to-point messages.

1 From the WebSphere Administrative Console, click Resources > JMS Providers > Default Messaging to open the “Default messaging provider” pane.

2 Under Destinations, click **JMS queue** to open the JMS queue pane where you can create a new queue for continuous inbound messages.
3 Click New to complete the General Properties section for the new continuous inbound messaging queue. At a minimum, enter the following information.

<table>
<thead>
<tr>
<th>Column</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>cqin</td>
</tr>
<tr>
<td>JNDI name</td>
<td>jms/mro/int/queues/cqin</td>
</tr>
<tr>
<td>Bus name</td>
<td>meajmsbus</td>
</tr>
<tr>
<td>Queue name</td>
<td>cqinbd</td>
</tr>
</tbody>
</table>

**JMS Providers - Add General Properties for Continuous Inbound Queue (cqin)**

4 Click OK.

5 Click Save.

6 Click the "Synchronize changes with Nodes" box.

7 Click Save.

**Creating the Sequential Inbound (squin) JMS Queue**

You must create a JMS queue (squin) as the destination for sequential inbound point-to-point messages.

1 From the WebSphere Administrative Console, click Resources > JMS Providers > Default Messaging to open the “Default messaging provider” pane.

2 Under Destinations, click JMS queue to open the JMS queue pane where you can create a new queue for sequential inbound messages.
3 Click **New** to complete the General Properties section for the new sequential inbound messaging queue. At a minimum, enter the following information.

<table>
<thead>
<tr>
<th>Column</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>sqin</td>
</tr>
<tr>
<td>JNDI name</td>
<td>jms/mro/int/queues/sqin</td>
</tr>
<tr>
<td>Bus name</td>
<td>meajmsbus</td>
</tr>
<tr>
<td>Queue name</td>
<td>sqinbd</td>
</tr>
</tbody>
</table>

4 Click **OK**.

5 Click **Save**.

6 Click the "Synchronize changes with Nodes" box.

7 Click **Save**.

### Creating the Sequential Outbound (sqout) JMS Queue

You must create a JMS queue (sqout) as the destination for sequential outbound point-to-point messages.

1 From the WebSphere Administrative Console, click Resources > JMS Providers > Default Messaging to open the “Default messaging provider” pane.

2 Under Destinations, click **JMS queue** to open the JMS queue pane where you can create a new queue for sequential outbound messages.

3 Click **New** to complete the General Properties section for the new sequential outbound messaging queue. At a minimum, enter the following information.

<table>
<thead>
<tr>
<th>Column</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>sqout</td>
</tr>
<tr>
<td>JNDI name</td>
<td>jms/mro/int/queues/sqout</td>
</tr>
<tr>
<td>Bus name</td>
<td>meajmsbus</td>
</tr>
<tr>
<td>Queue name</td>
<td>sqoutbd</td>
</tr>
</tbody>
</table>

4 Click **OK**.

5 Click **Save**.

6 Click the "Synchronize changes with Nodes" box.

7 Click **Save**.
Creating JMS Activation for the Continuous Inbound Queue (cqin)

You must activate the continuous inbound queue (cqin) before it can receive messages. Complete the following steps to activate the cqin queue:

1. From the WebSphere Administrative Console, click Resources > JMS Providers > Default Messaging to open the “Default messaging provider” pane.

2. Under Activation Specifications, click JMS activation specification to open the JMS activation specification pane where you can create an activation configuration that will enable cqin to receive inbound messages.

3. Click New to complete the General Properties section for the new JMS activation specification. At a minimum, enter the following information.

<table>
<thead>
<tr>
<th>Column</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>meajmsact</td>
</tr>
<tr>
<td>JNDI name</td>
<td>meajmsact</td>
</tr>
<tr>
<td>Destination type</td>
<td>Queue (default)</td>
</tr>
<tr>
<td>Destination JNDI name</td>
<td>jms/mro/int/queues/cqin</td>
</tr>
<tr>
<td>Bus name</td>
<td>meajmsbus</td>
</tr>
<tr>
<td>Maximum batch size</td>
<td>10</td>
</tr>
<tr>
<td>Maximum concurrent endpoints</td>
<td>5</td>
</tr>
</tbody>
</table>

The following figure shows the General Properties pane (note that you have to scroll down in the pane to view Maximum batch size and Maximum concurrent endpoints):

![General Properties Pane](image)

4. Click OK.

5. Click Save.
Creating the MEA Connection Factory and Queues in WebSphere

6 Click the "Synchronize changes with Nodes" box.

7 Click Save.

8 Stop the application server.

9 Restart the Node Agent.

10 Start the application server.

If Maximo is already installed on your application server, you must rebuild and deploy the Maximo.ear file. Refer to the IBM Maximo System Administrator’s Guide for detailed instructions.

NOTE Unlike WebLogic, WebSphere JMS does not have a Bytes Maximum parameter, so you might encounter an Out of Memory error. However, WebSphere provides a Max Messages parameter to limit the number of records in a JMS queue.
This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those
Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
2Z4A/101
11400 Burnet Road
Austin, TX 78758 U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

## Trademarks

AIX, IBM, IBM Corporation, the IBM logo, Tivoli, and WebSphere are registered trademarks of the International Business Machines Corporation in the United States, other countries, or both.

Intel, the Intel logo, and Pentium are registered trademarks of the Intel Corporation in the United States, other countries, or both.

Microsoft, Windows, and Internet Explorer are registered trademarks of Microsoft Corporation in the United States, other countries, or both.
Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Sun, Sun Microsystems, the Sun logo, and Solaris are trademarks of Sun Microsystems, Inc.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.
A

ACTION column, interface tables 5-11
actions, processing rule 14-4
adapters
- creating 16-11
- definition 2-9
- external type 2-10
- internal type 2-9
- modifying 16-15
- modifying predefined 16-15
Add action, Maximo XML 4-9
AddChange action, Maximo XML 4-10
Administrator E-mail Address 6-7
alias column names 5-6
alternate keys 16-4

B

batch processing initiation 3-4
Boolean columns, Maximo XML 4-18
Boolean type interface controls 14-19

C

Change action, Maximo XML 4-10
changed attribute, Maximo XML 4-15
checklist, configuration 6-3
cluster configuration
- cron tasks 13-2
- EJB 13-3
- gateway Web service 13-3
- global directory 13-3
- HTTP servlet 13-3
- integration gateway 13-3
- Maximo Web services 13-4
- queues 13-2
collaboration switches
- adding C-6
- default C-3
- format C-1
- inventory C-9
- invoice C-11
- labor transaction C-13
- modifying C-6
- overview C-1
- purchase order C-13
- purchase requisition C-15
- receipt C-17
- retrieving C-4
collaboration switches (continued)
- viewing C-5
- work order C-18
column name length 5-6
Combine action, processing rules 14-8
comparison field, processing rules 14-16
conditions, processing rules 14-10
configuration
- Administrator E-mail Address 6-7
- checklist 6-3
- cron task 5-12, 10-2, 14-3
- Data Export feature 6-4, 6-36
- Data Import feature 6-4, 6-38
- Domain values setting 6-7
- EJB 13-3
- end points 6-15, 6-17
- error management 7-4
- event listeners 6-17
- external systems 6-13, 6-32
- gateway Web service 13-3
- global directory 6-7, 6-32, 13-3
- HTTP servlet 13-3
- IFACETABLECONSUMER cron task 6-35
- inbound queues 6-21
- Integration Administration properties 6-6
- integration gateway 13-3
- interface controls 6-23, 6-28
- interface tables 6-33
- interfaces 6-20
- mail server host 6-7, 6-32
- Maximo Web services 13-4, 18-2
- MEA connection factory D-19
- message data beans 6-12
- prerequisites 6-2, 6-6
- queues 6-8, 6-14, 13-2
- redelivery delay D-21
- Sender E-mail Address 6-7
- Update SENDERSYSID setting 6-7
- Web services 6-4
content element
- Maximo XML 4-6
- query interfaces 17-5
- query response interfaces 17-8
continuous queue
- definition 2-17
- message data beans 6-12, 11-3
- overview 11-3
- pipeline size 11-4
- redelivery delay 11-4
conversion, data format 2-13
CreationDateTime field 4-5
Critical Errors 7-9
cron tasks
- cluster configuration 13-2
- configuration 6-10
- IFACETABLECONSUMER 5-12, 10-2
- JMSQSEQCONSUMER 11-2
- selectors 10-2, 11-5
cross-reference type interface controls 14-19

© Copyright IBM Corp. 2007
Index

customer support iv-xii

customization
  EJB handler 15-8
  handlers 9-15, 15-7
  HTTP handler 15-7
  inbound processing 15-12
  integration points 15-4, 15-17
  interface tables 15-13
  interfaces 15-5, 15-16
  JMS handler 15-9
  MBO user exit processing 15-18
  outbound processing 15-2
  processing rules 14-1
  user exit postprocessing 15-6, 15-16
  user exit preprocessing 15-4, 15-15
  Web Service handler 15-10
  XSL mapping 15-6, 15-14

D

Data Export feature
  configuration 6-4, 6-36
  restrictions 3-17, 6-36

data formats
  conversion 2-13
  date format 4-18
  flat file 2-13
  interface tables 2-13, 5-5
  Maximo XML 2-13, 4-2
  numbers 4-18
  overview 2-13

Data Import feature
  configuration 6-4, 6-38
  inbound processing 3-20
  restrictions 6-38

data synchronization, Web services 3-37

date format 4-18

Delete action 4-9

deletequeue utility 11-6

deploying Maximo Web services 18-4, 18-7

document interfaces B-24

Domain values setting 6-7

duplicate column names 16-6

E

EAR file, rebuilding 6-7, 6-12, 6-32, 7-5, 9-5, 15-1

EJB
  cluster configuration 13-3
  customization 15-8
  handler 9-4
  integration gateway 8-5

echo
  configuration 6-15, 6-17
  definition 2-16
  MXFLATFILE 9-2
  MXIFACETABLE 9-2
  MXXMLFILE 9-2
  predefined 9-2

error management
  cause of errors 7-9
  configuration 7-4
  correcting errors 7-9
  critical errors 7-9
  deleting errors 7-13
  error notification 7-6
  external systems properties 7-5
  folders 7-7
  Integration Administration properties 7-4
  interface tables 7-3
  maximo.properties configuration 7-5
  overview 7-2
  queues 7-2
  researching errors 7-10
  updating message data 7-12

evaluation category, processing rule 14-11

evaluation field, processing rule 14-13, 14-15

evaluation type, processing rule 14-14

event attribute, Maximo XML 4-4

event listeners
  configuration 6-17
  integration point 2-8

eexternal adapters 2-10

eexternal systems
  configuration 6-13, 6-32
  creating 16-16
  definition 2-15
  enabling 6-32
  error management 7-5
  identifier 2-15
  interface table requirements 5-13
  modifying predefined 16-17
  outbound processing 3-15
  processing overview 2-3
  restrictions 16-17

EXTSYSNAME column, interface tables 5-10

F

field attributes, Maximo XML 4-15

flat files
  alias column names 5-6
  definition 2-13
  duplicate columns 5-6
  formatting 9-7

FLATFILE handler 9-6

G

Gateway Properties tab 8-2

gateway Web service
  cluster configuration 13-3
  security 12-6
  transactions 8-7

global directory
  cluster configuration 13-3
  configuration 6-7, 6-32
  integration gateway 13-3
  glorder attribute, Maximo XML 4-16
Index

H

handlers
  customization 9-15, 15-7
  EJB 9-4
  FLATFILE 9-6
  HTTP 9-8
  IFACETABLE 9-9
  JMS 9-10
  predefined 9-3
  WEBSERVICE 9-12
  XMLFILE 9-14
HASLD column 5-7
header attributes
  Maximo XML 4-4
  queries 17-3
  query responses 17-4
header element, Maximo XML 4-4
header fields, Maximo XML 4-5
HTTP handler 9-8, 15-7
HTTP servlet
  cluster configuration 13-3
  security 12-7
HTTP/HTTPS transactions 8-6

I

IFACENaMe column, interface tables 5-7
IFACETABLE handler 9-9
IFACETABLECONSUMER cron task 5-12, 6-35, 10-2
inbound processing
  alternate keys 16-4
  Boolean columns 4-18
  customization 15-12
  Data Import feature 3-20
  inbound queues 3-19, 3-22
  initiating 3-18
  integration gateway 3-19
  integration object processing rules 3-30
  integration objects 3-28
  integration points 3-23
  interface classes 3-25
  interface tables 3-21
  MBO processing rules 3-32
  MBOs 3-31, 3-35
  multiple languages 19-3
  null columns 4-18
  overview 2-18, 3-16
  processing classes 3-33
  queue-based processing 3-17
  STATUSIFACE field B-25
  user exit MBO processing 3-34
  user exit postprocessing 3-26
  user exit preprocessing 3-24
  Web services processing 3-36
  XSL mapping 3-27
inbound queues
  configuration 6-21
  inbound processing 3-19, 3-22
  inbound setting restrictions 16-7
Integration Administration properties 6-6
  7-4
integration gateway
  bypassing interpretation 8-4
  EJB transactions 8-5
  Gateway Properties tab 8-2
  gateway Web service transactions 8-7
  HTTP/HTTPS transactions 8-6
  inbound processing 3-19
  interpretation layer 8-2
  interpreter 8-3
  overview 8-2
  security 12-4
integration objects
  creating 16-2
  inbound processing 3-28
  inbound setting restrictions 16-7
  modifying predefined 16-8
  outbound processing 3-5, 3-8
  overview 2-5
  restrictions 16-8
integration points
  creating 16-9
  customization 15-4, 15-17
  definition 2-7
  direction property 2-7
  event listeners 2-8
  inbound processing 3-23
  modifying predefined 16-11
  operation property 2-7
  outbound processing 3-7
  overview 2-7
  processing class 2-8
  properties 2-7
  restrictions 2-7, 16-11
interface classes
  inbound processing 3-25
  outbound processing 3-12
interface controls
  Boolean type 14-19
  configuration 6-23
  creating 14-20
  cross-reference type 14-19
  default values 2-16
  levels 14-18
  list type 14-19
  organization/site level values 6-28
  organization-level controls 14-18
  overview 2-14, 14-18
  restrictions 6-28
  site-level controls 14-18
  system level values 6-23
  system-level controls 14-18
  types 14-19
  value type 14-19
interface element, Maximo XML 4-3
interface queue tables 5-3
interface tables
  ACTION column 5-11
  alias column names 5-6
  column name lengths 5-6
  creating 5-4, 6-33
  customization 15-13
  data format 2-13
  deleting 5-5
  duplicate columns 5-6, 16-6
  error management 7-3
  external system requirements 5-13
  EXTSYSNAME column 5-10
  format 5-5
  HASLD field 5-7
Index

interface tables (continued)
  IFACENAME column 5-7
  IFACETABLECONSUMER cron task 6-35, 10-2
  inbound processing 3-21
  JMSQSEQCONSUMER cron task 11-2
  key columns 5-5
  location 5-2
  long description column 5-11
  multiple queue tables 10-3
  naming 5-2
  outbound processing 5-13
  polling 5-12, 10-1
  processing columns 5-7
  queue tables 5-3
  regenerating 5-4
  restrictions 2-13, 3-21, 5-4
  security 12-12
  selectors 10-2
  TRANSID column 5-8
  TRANSSEQ column 5-10

interfaces
  configuration 6-20
  creating 16-13
  customization 15-5, 15-16
  direction 2-11
  modifying predefined 16-15
  operation property 2-12
  overview 2-9
  processing classes 2-12
  processing overview 2-3
  processing rules 2-14
  restrictions 16-15
  user exit classes 2-12

internal adapters 2-9
interpretation layer 8-2
interpretation, bypassing 8-4
interpreter, integration gateway 8-3
inventory collaboration switches C-9
invoice collaboration switches C-11
invoking Maximo Web service 18-9
item interfaces B-15
IVMATCH collaboration switch B-33

J

JMS handler 9-10, 15-9
JMSQSEQCONSUMER cron task
  configuration 6-10
  properties 11-2

K

key columns, interface table 5-5

L

labor transaction collaboration switches C-13
langenabled attribute 4-17, 19-2

language attribute 4-3, 19-2
list type interface controls 14-19
long description column, interface table 5-11

M

mail server host 6-7, 6-32
master data interfaces B-2
MAXIMO adapter 1-1
Maximo application server, restarting 6-7, 6-12, 6-32, 7-4, 13-2, 18-7
Maximo Web services
  cluster configuration 13-4
  configuration 18-2
  deploying 18-4
  invoking 18-9
  restrictions 2-9
  security 12-9
  UDDI registration 18-8
  WSDL generation 18-6
Maximo XML
  action attributes 4-8, 4-14
  Add action 4-9
  AddChange action 4-10
  Boolean columns 4-18
  Change action 4-10
  changed attribute 4-15
  content element 4-6
  CreationDateTime field 4-5
  date format 4-18
  definition 2-13
Delete action 4-9
  encoding 4-2
  event attribute 4-4
  field attributes 4-15
  glorder attribute 4-16
  header attributes 4-4
  header element 4-4
  header fields 4-5
  interface element 4-3
  langenabled attribute 4-17
  language attribute 4-3
  maxItems attribute 17-3
  maxvalue attribute 4-17
  merged integration objects 4-7
  MessageID field 4-5
  null columns 4-18
  number format 4-18
  operation attribute 4-4
  RecipientID field 4-5
  Replace action 4-10
  rsCount attribute 17-4
  rsStart attribute 17-3, 17-4
  rsTotal attribute 17-4
  SenderID field 4-5
  standard integration objects 4-6
  structure 4-2
  uniqueResult attribute 17-3
  xmlns attribute 4-3
  maximo.properties 7-5
  Maximum Try Count property 7-5
maxItems attribute, Maximo XML 17-3
maxvalue attribute, Maximo XML 4-17
MBOs
  definition 2-5
  inbound processing 3-31, 3-35
MBOs (continued)
  integration objects 16-2
  processing rules 3-32, 14-2, 14-3
  user exit processing 15-18
MEA connection factory D-19
merged integration objects 4-7
message data beans 6-12, 11-3
MessageID field 4-5
multiple languages 19-1
multiple queue tables 10-3
multiplication controls 6-27
MXACTIONInterface A-4, B-63
MXASSETInterface A-2, B-3
MXCOAInterface A-2, B-4
MXCRAFTInterface A-2, B-6
MXCTEMPLATEInterface A-4, B-62
MXDOMAINInterface A-4, B-59
MXEMPACTInterface A-2, B-50
MXENDPOINTInterface A-4, B-55
MXEXTSYSInterface A-4
MXEXTSYSTEMInterface B-56
MXFLATFILE end point 9-2
MXGLCOMPInterface A-2, B-7
MXGLTXNInterface A-2, B-45
MXIFACetable end point 9-2
MXINTERFACEInterface A-4, B-54
MXINTOBJInterface A-4, B-52
MXINTTYPEInterface A-4, B-53
MXINVIRALInterface A-2, B-20
MXINVENTORYInterface A-2, B-19
MXINVISSUEInterface A-2, B-23
MXINVOICEInterface A-2, B-32
MXINVVENDEORInterface A-2, B-32
MXITEMInterface A-2, B-16
MXLABORInterface A-2, B-8
MXMAXROLEInterface A-4, B-64
MXMESSAGEInterface A-4, B-57
MXOBJECTCFGInterface A-4, B-58
MXOPERLOCInterface A-3, B-9
MXPCInterface A-3, B-28
MXPERSONInterface A-3, B-10
MXPOLInterface A-3, B-30
MXPRInterface A-3, B-31
MXPROInterface A-3, B-11
MXRCVROTTIMInterface A-3, B-43
MXRECEIPTInterface A-3, B-40
MXSERVITEMInterface A-3, B-17
MXSTORELOCInterface A-3, B-12
MXTOOLITEMInterface A-3, B-18
MXVENDORInterface A-3, B-13
MXVENDORMSTRInterface A-3, B-14
MXXFInterface A-4, B-60
MXXFDETAILInterface A-3, B-37
MXXFInterface A-3, B-36
MXXMFILE end point 9-2

N

non-persistent fields 2-6, 16-6
null columns, Maximo XML 4-18
number format, Maximo XML 4-18

O

online support iv-xii
operation attribute, Maximo XML 4-4
operation property, integration point 2-7
Oracle databases and long description column 5-11
organization-level interface controls 14-18
outbound processing
  batch processing 3-4
  customization 15-2
  external system processing 3-15
  initiation 3-3
  integration objects 3-5, 3-8
  integration points 3-7
  interface processing class 3-12
  interface tables 5-13
  multiple languages 19-2
  overview 2-19
  prerequisites 3-2
  processing rules 3-10
  purchasing interfaces B-26
  real time processing 3-3
  STATUSIFACE field B-24
  user exit postprocessing 3-13
  user exit preprocessing 3-11
  work order interfaces B-35
  XSL mapping 3-14

P

persistent fields 2-6, 16-6
pipeline size 11-4
polling, interface tables 5-12, 10-1
predefined components 6-2
predefined interfaces
  document data B-24
  item data B-15
  master data B-2
  system data B-51
  transaction data B-39
prerequisites
  configuration 6-2, 6-6
  outbound processing 3-2
  queue-based inbound processing 3-17
  process sequence, processing rule 14-10
  processing classes
    inbound processing 3-33
    integration point 2-8
    interfaces 2-12
    user exit 2-12
  processing rules
    actions 14-4
      Combine action 14-8
      comparison field 14-16
      conditions 14-10
      defining 14-3
      evaluation category 14-11
      evaluation field 14-13, 14-15
      evaluation type 14-14
      inbound processing 3-30
      interfaces 2-14
      MBOs 14-2, 14-3
      outbound processing 3-10
      processing sequence 14-10
processing rules (continued)
  purchasing interfaces B-26
  Replace action 14-7
  restrictions 14-2
  Set action 14-6
  Skip action 14-4
  Skipchildren action 14-5
  Split action 14-9
  Stop action 14-5
  sub-records 14-2, 14-3
  triggers 14-4
  work order interfaces B-35

properties
  integration point 2-7
  interface operation 2-12
  queues 6-8

purchase order collaboration switches C-13
purchase requisition collaboration switches C-15

restrictions (continued)
  integration objects 16-8
  integration points 2-7, 16-11
  interface controls 6-28
  interface tables 2-13, 3-21, 5-4
  interfaces 16-15
  Maximo Web services 2-9
  Maximum Try Count property 7-5
  multiple languages 19-3
  multiplication controls 6-27
  predefined components 6-2
  processing rules 14-2
  queries 2-12, 17-2, 17-5
  query responses 2-12
  queue security 12-3
  rsCount attribute, Maximo XML 17-4
  rsStart attribute, Maximo XML 17-3, 17-4
  rsTotal attribute, Maximo XML 17-4

Q

queries
  content element 17-5
  creating 17-2
  header attributes 17-3
  restrictions 2-7, 2-12, 17-2, 17-5
  Web service processing 3-38, 17-8

query responses
  content element 17-8
  header attributes 17-4
  restrictions 2-7, 2-12

queues
  cluster configuration 13-2
  configuration 6-14
  creating 11-7, D-1
  definition 2-17
  deletequeue utility 11-6
  error management 7-2
  properties 6-8
  security 12-2
  viewqueue utility 11-6

S

schemas, XML
  attributes 4-19
  content 4-19
  directories 4-20
  generating 4-20
  key fields 4-19
  overview 4-19
  validation 4-19

security
  gateway Web service 12-6
  HTTP servlet 12-7
  integration gateway 12-4
  interface tables 12-12
  Maximo Web services 12-9
  queues 12-2
  remote Integration APIs 12-12
  selectors, cron task 10-2, 11-5
  Sender E-mail Address 6-7
  SenderID field 4-5
  sequential queues
    cron task, configuring 6-10
    definition 2-17
    overview 11-1

Set action 14-6
site-level interface controls 14-18
Skip action 14-4
Skipchildren action 14-5
Split action 14-9
standard integration objects 4-6
STATUSIFACE field B-24
Stop action 14-5
sub-records
  definition 2-5
  processing rules 14-2, 14-3
support, online iv-xii
system interfaces B-51
system-level interface controls 14-18
T

TRANS_LANGCODE field 4-18, 19-2, 19-3
transaction interfaces B-39
TRANSID column 5-8
translatable columns 19-2
TRANSSEQ column 5-10
triggers, processing rule 14-4

U

UDDI registration 18-8
uniqueResult attribute 17-3
Update SENDER SYSID setting 6-7
user defined fields 16-6
user exit MBO processing 3-34
user exit postprocessing 15-16
  customization 15-6
  inbound processing 3-26
  outbound processing 3-13
user exit preprocessing
  customization 15-4, 15-15
  inbound processing 3-24
  outbound processing 3-11

V

value type interface controls 14-19
viewqueue utility 11-6

W

Web services
  configuration 6-4
  customization 15-10
  data synchronization 3-37
  deploying 18-7
  overview 3-36
  queries 3-38, 17-8
WEBSERVICE handler 9-12
work order collaboration switches C-18
WSDL generation 18-6

X

XML schemas
  attributes 4-19
  content 4-19
  directories 4-20
  generating 4-20, 18-5
  key fields 4-19
  overview 4-19
  validation 4-19
XMLFILE handler 9-14
xmlns attribute 4-3
XSL mapping
  customization 15-6, 15-14
  inbound processing 3-27
  outbound processing 3-14