MICRO FOCUS | SERENA

ChangeMan® ZMF

ERO XML Services User's Guide

Product version: 8.1

Publication date: September 2018 (rebranded only)

Table of Contents

Welcome to ChangeMan® ZMF

ChangeMan ZMF is a comprehensive and fully integrated solution for Software Change Management systems in z/OS environments. It provides reliable and streamlined implementation of software changes from development into production. ChangeMan ZMF manages and automates the application life cycle, protects the integrity of the code migration process, and results in higher quality delivered code to any test environment and to the production environment.

Before You Begin

See the Readme for the latest updates and corrections for this manual.

Objective

The *ChangeMan ZMF ERO XML Services User's Guide* documents the Enterprise Release Option (ERO) functions and services available for general customer use. These services are also known as the "green" services and provide mostly search and query functions.

Audience

This document is intended for ChangeMan ZMF administrators, release managers, and developers who use the ChangeMan ZMF ERO Option. It is assumed that you are familiar with your mainframe operating system and security system, and that you understand the operation and administration of ChangeMan ZMF. Some familiarity with basic XML syntax and schemas is helpful.

It is also assumed that you are familiar with XML Services concepts, architecture, and syntax; if you are not, then refer to chapters 1 and 2 of the *ChangeMan ZMF XML Services User's Guide* for a comprehensive overview.

Navigating this book

- Chapter 1 provides an overview of the ERO XML services.
- Chapter 2 describes the XML syntax, data structures and values, code examples, and usage tips for the ERO "green" services.

Guide to ChangeMan ZMF Documentation

The following sections provide basic information about ChangeMan ZMF documentation.

ChangeMan ZMF Documentation Suite

You can download all ChangeMan ZMF documentation in Adobe Portable Document Format (PDF) format from the following URL:

https://www.microfocus.com/support-and-services/documentation/

The ChangeMan ZMF documentation set includes the following manuals in PDF format.

Manual	Description
Administrator's Guide	Describes ChangeMan ZMF features and functions with instructions for choosing options and configuring global and application administration parameters.
ChangeMan ZMF Quick Reference	Provides a summary of the commands you use to perform the major functions in the ChangeMan ZMF package life cycle.

Manual	Description
Customization Guide	Provides information about ChangeMan ZMF skeletons, exits, and utility programs that will help you to customize the base product to fit your needs.
Db2 Option Getting Started Guide	Describes how to install and use the Db2 Option of ChangeMan ZMF to manage changes to Db2 components.
ERO Concepts	Discusses the concepts of the ERO Option of ChangeMan ZMF for managing releases containing change packages.
ERO Getting Started Guide	Explains how to install and use the ERO Option of ChangeMan ZMF to manage releases containing change packages.
IMS Option Getting Started Guide	Provides instructions for implementing and using the IMS Option of ChangeMan ZMF to manage changes to IMS components.
INFO Option Getting Started Guide	Describes two methods by which ChangeMan ZMF can communicate with other applications: ■ Through a VSAM interface file. ■ Through the Tivoli Information Management for z/OS product from IBM.
Installation Guide	Provides step-by-step instructions for initial installation of ChangeMan ZMF. Assumes that no prior version is installed or that the installation will overlay the existing version.
Java / zFS Getting Started Guide	Provides information about using ZMF to manage application components stored in USS file systems, especially Java application components.
Load Balancing Option Getting Started Guide	Explains how to install and use the Load Balancing Option of ChangeMan ZMF to connect to a ChangeMan ZMF instance from another CPU or MVS image.
M+R Getting Started Guide	Explains how to install and use the M+R Option of ChangeMan ZMF to consolidate multiple versions of source code and other text components.
M+R Quick Reference	Provides a summary of M+R Option commands in a handy pamphlet format.
Messages	Explains messages issued by ChangeMan ZMF, SERNET, and System Software Manager (SSM) used for the Staging Versions feature of ChangeMan ZMF.
Migration Guide	Gives guidance for upgrading ChangeMan ZMF
OFM Getting Started Guide	Explains how to install and use the Online Forms Manager (OFM) option of ChangeMan ZMF.
SER10TY User's Guide	Gives instructions for applying licenses to enable ChangeMan ZMF and its selectable options.
User's Guide	Describes how to use ChangeMan ZMF features and functions to manage changes to application components.

Manual	Description
XML Services User's Guide	Documents the most commonly used features of the XML Services application programming interface to ChangeMan ZMF.
Web Services Getting Started Guide	Documents the Web Services application programming interface to ChangeMan ZMF.

Using the Manuals

To view PDF files, use Adobe® Reader®, which is freely available from www.adobe.com.

TIP Be sure to download the *full version* of Reader. The more basic version does not include the search feature.

This section highlights some of the main Reader features. For more detailed information, see the Adobe Reader online help system.

The PDF manuals include the following features:

- **Bookmarks.** All of the manuals contain predefined bookmarks that make it easy for you to quickly jump to a specific topic. By default, the bookmarks appear to the left of each online manual.
- **Links.** Cross-reference links within a manual enable you to jump to other sections within the manual and to other manuals with a single mouse click. These links appear in blue.
- **Printing.** While viewing a manual, you can print the current page, a range of pages, or the entire manual.
- **Comments.** All PDF documentation files that are delivered with ChangeMan ZMF have enabled commenting with Adobe Reader. Adobe Reader version 7 and higher has commenting features that enable you to post comments to and modify the contents of PDF documents. You access these features through the Comments item on the menu bar of the Adobe Reader.
- **Advanced search.** Starting with version 6, Adobe Reader includes an advanced search feature that enables you to search across multiple PDF files in a specified directory. (This is in addition to using any search index created by Adobe Catalog—see step 3 below.)

To search within multiple PDF documents at once, perform the following steps (requires Adobe Reader version 6 or higher):

1. In Adobe Reader, select Edit | Search (or press CTRL+F).
2. In the text box, enter the word or phrase for which you want to search.
3. Select the **All PDF Documents in** option, and browse to select the folder in which you want to search.
4. Optionally, select one or more of the additional search options, such as **Whole words only** and **Case-Sensitive**.

5 Click the **Search** button.

NOTE Optionally, you can click the **Use Advanced Search Options** link near the lower right corner of the application window to enable additional, more powerful search options. (If this link says **Use Basic Search Options** instead, the advanced options are already enabled.) For details, see Adobe Reader's online help.

Typographical Conventions

The following typographical conventions are used in the online manuals and online help. These typographical conventions are used to assist you when using the documentation; they are not meant to contradict or change any standard use of typographical conventions in the various product components or the host operating system.

Convention	Explanation
italics	Introduces new terms that you may not be familiar with and occasionally indicates emphasis.
bold	Emphasizes important information and field names.
UPPERCASE	Indicates keys or key combinations that you can use. For example, press the ENTER key.
`monospace`	Indicates syntax examples, values that you specify, or results that you receive.
`monospaced italics`	Indicates names that are placeholders for values you specify; for example, *`filename`*.
`monospace bold`	Indicates the results of an executed command.
vertical rule |	Separates menus and their associated commands. For example, select File | Copy means to select Copy from the File menu. Also, indicates mutually exclusive choices in a command syntax line.

Chapter 1
Introduction

This chapter provides an overview of the "green" XML services for the Enterprise Release Option (ERO).

For a comprehensive overview of XML Services concepts, architecture, and syntax, refer to chapters 1 and 2 of the *ChangeMan ZMF XML Services User's Guide*.

For more information about ERO, refer to the following ChangeMan ZMF documents:

ERO Concepts
ERO Getting Started Guide

Services Summary

There are 45 ERO XML services available for general customer use. These are also known as the "green" services and provide mostly search and query functions.

The following table provides a summary of these services in alphabetical order, with links to the page where the service is documented.

Service Name	Scope Name	Message Name	Description of Function	COBOL Copybook	Page #
package	service	• attach • detach	• Attaches a package to a release • Detaches a package from a release	• XMLCPGPM • XMLCPGPM	16 17
rlsmappl	promote	• list	• Lists release management promotion data	• XMLCRPRM	19
	service	• list • release	• Lists application release status • Lists release data for each release to which an application is joined	• XMLCRAPL • XMLCRARL	21 24
	syslib	• list	• Lists SYSLIB data for release applications	• XMLCRASY	27
rlsmappr	area	• list	• Lists release area approver data	• XMLCRAAP	30
	ascapprv	• list	• Lists the items that are associated with an approval entity	• XMLCRASC	34
	global	• list	• Lists global release approval entity data	• XMLCRGAP	36
	release	• list	• Lists data for install approval entities	• XMLCRAAP	39
rlsmarea	all_chk	• syslib	• Lists the COPYLIB, LOADLIB, and source concatenation lists for libraries that are allocated	• XMLCRSYL	41
	all_noc	• syslib	• Lists all of the COPYLIB, LOADLIB, and source concatenation lists, including libraries that are not yet allocated	• XMLCRSYL	45
	cim	• list	• Lists release area component in motion (CIM) information from the ERO Db2 CIM table	• XMLCRCIM	47
	cmp_lock	• list	• Lists the holder of a release component lock	• XMLCRCLK	53
	cpy	• syslib	• Lists the COPYLIB concatenation for a release application	• XMLCRSYL	55
	detail	• cmp_rlse • integrty • test	• Lists all components in a release concatenation and shows all locations where each component resides • Checks the integrity of the component-in-motion (CIM) table against physical members in area libraries. Checks all versions of all components in the release concatenation. • Tests the contents of a release area against all of the packages that may place a component in that area. Lists information for failing components and packages.	• XMLCRCML • XMLCRCHK • XMLCRTST	57 60 62
	hst	• list	• Lists history from the ERO component history table	• XMLCRHST	64
	imp	• list	• Lists impact data from the ERO Db2 impact table	• XMLCRIMP	72

Service Name	Scope Name	Message Name	Description of Function	COBOL Copybook	Page #
	load	• syslib	• Lists the LOADLIB concatenation for a release application	• XMLCRSYL	75
	scan	• cmp_rlse	• Scans the latest version of components in a release concatenation to find those with content matching a search string	• XMLCRCML	77
	scanall	• cmp_rlse	• Scans all components in a release concatenation to find those with content matching a search string	• XMLCRCML	80
	service	• list • test	• Lists release area definitions • Tests the contents of a release against all of the packages that may place a component in that release. Displays a message describing the status of packages and components in the release.	• XMLCRARE • XMLCRTST	81 86
	source	• syslib	• Lists the source SYSLIB information for a library type	• XMLCRSYL	88
	start	• list	• Lists the release area definitions for a starting area	• XMLCRARE	90
	summary	• cmp_rlse • integrty	• Lists information for the latest version of each component in a release concatenation • Checks integrity of the component-in-motion (CIM) table against physical members in area libraries.	• XMLCRCML • XMLCRCHK	92 95
	syslib	• list	• Lists SYSLIB data for an application	• XMLCRASL	97
	ver_regr	• list	• Performs a version regression check on components. If a version regression situation exists between the current release and a prior release, lists information for the current and prior versions.	• XMLCRVER	100
rlsmltyp	bun	• list	• Lists information from the release BUN library-type table	• XMLCRBUN	104
	service	• list	• Lists library security and format information	• XMLCRLTP	107
rlsmrlse	cim	• list	• Lists release area component in motion (CIM) information from the ERO Db2 CIM table	• XMLCRLCM	112
	detail	• test	• Tests the contents of a release against all of the packages that may place a component in that release	• XMLCRTSC	114
	hst	• list	• Lists release component history from the ERO component history table	• XMLCRLHT	117
	imp	• list	• Lists impact data from the ERO Db2 impact table	• XMLCRLMP	118
	library	• list	• Lists release area libraries	• XMLCRLLT	120
	prior	• list	• Lists prior release information	• XMLCRLPR	123
	reasons	• list	• List Backout and Revert reasons for a release	• XMLCRRBR	126

Service Name	Scope Name	Message Name	Description of Function	COBOL Copybook	Page #
	rls_link	• list	• Lists release management data across a TCP/IP link	• XMLCRLLK	129
	service	• list • search • test	• Lists scheduler dates, times, and status for a release • Searches for releases and lists information • Tests the contents of a release against all of the packages that may place a component in that release. Displays a status message.	• XMLCRLSM • XMLCRSRC • XMLCRTSC	132 139 149
	sites	• list	• Lists release/site dates and contacts	• XMLCRSTE	152

Special Syntax Conventions

XML adopts certain syntax conventions that apply in some search, summary, and analysis contexts.

Semicolon-Delimited Lists

The allowed contents of commonly used tags may be different in search contexts than in non-search contexts. For example, user ID and security entity tags, which elsewhere permit only a single value, may accept a semicolon-delimited list of TSO user IDs or security entities in a search context. When a tag allows multiple values to be entered, it is noted in the tag description.

Yes/No Flag Tags

All search, summary, and analysis services in XML follow common default value conventions for yes/no flag tags. They also share common conventions for Boolean relationships among flag tag values.

The key to these conventions is the flag tag *group*. An example of a group is the release status flag tags in the RLSMRLSE SERVICE SEARCH service on page 139. The values of all flag tags within a group are considered together; in fact, such mutual processing is the basis for identifying such tags as a group. Different flag tag groups may be supported by a single service; however, each group is evaluated independently of the other groups.

Default Values Within a Group

All yes/no flag tags in a group default to the value "Y" if no tag in the group has an explicitly assigned value. But if *any* flag tag in the group is explicitly assigned a value, all other tags in the group change their default values to "N."

For example, if your XML request says nothing about which release statuses to include, the release status flag tags all default to "Y" values, and you will get *all* release statuses in your reply. But if your XML request explicitly states you want all releases in DEV status, the assumption is made that you *don't* want releases in any other status. The flag tag value that you have set to "Y" retains that value, but the defaults for all other flag tags in

the group default to "N". To request multiple release statuses, you would set the yes/no flags for each desired release status to an explicit "Y" value.

IMPORTANT! If you explicitly set flag tag values, always include at least one "Y" value in each flag tag group. This is necessary because XML does not fully support Boolean NOT operations. You can *include* an item in a given state, and you can *exclude* an item in a given state, but you cannot include in your search results an item that is *not* in a given state. If you explicitly set just one flag tag in a group to a value of "N" and take the defaults for the rest, no results will be returned.

Boolean Relationships Within a Group

The "Y" values of all tags within a yes/no flag tag group are related by Boolean OR in search, summary, or analysis contexts. Any item in any state requested by a "Y" flag is returned in the results.

Tags with a value of "N" in a group are related by Boolean AND to the other tags in the group. Any item in any state identified by a flag with a value of "N" is excluded from the results.

Asterisk (*) Wildcard

Several tags allow the use of an asterisk (*) wildcard. Where this is allowed, it is indicated in the description for that tag.

Chapter 2

ERO General Services

XML supports the following ERO services for general use:

PACKAGE SERVICE ATTACH

The PACKAGE SERVICE ATTACH message attaches a package to a release.

The XML service/scope/message tags and attributes for this message are:

```
<service name="PACKAGE">
<scope name="SERVICE">
<message name="ATTACH">
```

These tags appear in both requests and replies.

PACKAGE SERVICE ATTACH — Request

The following example shows how you might code a request to attach a package to a release. Data structure details for the `<request>` tag follow the example.

Example XML — PACKAGE SERVICE ATTACH Request

```
<?xml version="1.0"?>
<service name="PACKAGE">
 <scope name="SERVICE">
  <message name="ATTACH">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
    <package>ACTP000094</package>
    <release>S4711010</release>
    <releaseArea>ACCTPAY </releaseArea>
   </request>
  </message>
 </scope>
</service>
```

PACKAGE SERVICE ATTACH <request> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<package>	Required	1	String (10)	Name of package to be attached.
<release>	Required	1	String (8), variable	Name of release to which package will be attached.
<releaseArea>	Required	1	String (8), variable	Name of starting subsystem area for package check-in to the release.

PACKAGE SERVICE ATTACH — Reply

The XML reply to a PACKAGE SERVICE ATTACH request does not return a `<result>` data structure. It does, however, return a standard `<response>` data structure to indicate the success or failure of the request. Successful requests have a return code of 00. Unsuccessful requests have a return code of 04 or higher.

Example XML — PACKAGE SERVICE ATTACH Reply

```
<?xml version="1.0"?>
<service name="PACKAGE">
 <scope name="SERVICE">
  <message name="ATTACH">
   <response>
    <statusMessage>CMR7508I - Package ACTP000094 is now attached to release
     S4711010.</statusMessage>
    <statusReturnCode>00</statusReturnCode>
    <statusReasonCode>7508</statusReasonCode>
   </response>
  </message>
 </scope>
</service>
```

PACKAGE SERVICE DETACH

The PACKAGE SERVICE DETACH message detaches a package from a release.

The XML service/scope/message tags and attributes for this message are:

```
<service name="PACKAGE">
<scope name="SERVICE">
<message name="DETACH">
```

These tags appear in both requests and replies.

PACKAGE SERVICE DETACH — Request

The following example shows how you might code a request to detach a package from a release. Data structure details for the `<request>` tag follow the example.

Example XML — PACKAGE SERVICE DETACH Request

```
<?xml version="1.0"?>
<service name="PACKAGE">
 <scope name="SERVICE">
  <message name="DETACH">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
    <package>ACTP000094</package>
   </request>
  </message>
 </scope>
</service>
```

PACKAGE SERVICE DETACH <request> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<package>	Required	1	String (10)	Name of package to be detached.

PACKAGE SERVICE DETACH — Reply

The XML reply to a PACKAGE SERVICE DETACH request does not return a `<result>` data structure. It does, however, return a standard `<response>` data structure to indicate the success or failure of the request. Successful requests have a return code of 00. Unsuccessful requests have a return code of 04 or higher.

Example XML — PACKAGE SERVICE DETACH Reply

```
<?xml version="1.0"?>
<service name="PACKAGE">
 <scope name="SERVICE">
  <message name="DETACH">
   <response>
    <statusMessage>CMR7507I - Package ACTP000094 is now detached from release
     S4711010.</statusMessage>
    <statusReturnCode>00</statusReturnCode>
    <statusReasonCode>7507</statusReasonCode>
   </response>
  </message>
 </scope>
</service>
```

RLSMAPPL PROMOTE LIST

The RLSMAPPL PROMOTE LIST message lists release management promotion data for a named release.

The XML service/scope/message tags and attributes for this message are:

```
<service name="RLSMAPPL">
<scope name="PROMOTE">
<message name="LIST">
```

These tags appear in both requests and replies.

RLSMAPPL PROMOTE LIST — Request

The following example shows how you might code a request to list promotion data for a release. The only required tag is `<release>`, however, additional tags may be specified to list specific release areas, promotion sites, and so on. Data structure details for the `<request>` tag follow the example.

Example XML — RLSMAPPL PROMOTE LIST Request

```
<?xml version="1.0"?>
<service name="RLSMAPPL">
 <scope name="PROMOTE">
  <message name="LIST">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
    <release>S4711010</release>
   </request>
  </message>
 </scope>
</service>
```

RLSMAPPL PROMOTE LIST <request> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<applName>	Optional	0 - 1	String (4), variable	Release application name.
<promotionLevel>	Optional	0 - 1	Integer (2)	Release promotion level.
<promotionName>	Optional	0 - 1	String (8), variable	Release promotion name.
<promotionSiteName>	Optional	0 - 1	String (8), variable	Release promotion site name.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<release>	Required	1	String (8), variable	Release name.
<releaseArea>	Optional	0 - 1	String (8), variable	Release area name.

RLSMAPPL PROMOTE LIST — Reply

The XML reply to a RLSMAPPL PROMOTE LIST request returns zero to many `<result>` data elements. Each `<result>` tag contains promotion data and status flags for a release area.

The standard `<response>` data element follows any `<result>` tags in the reply and indicates the success or failure of the request. Successful requests have a return code of 00. Unsuccessful requests have a return code of 04 or higher. Because it is the final data element in the XML reply message, the `<response>` tag serves as an end-of-list marker.

The following example shows what the reply message might look like. Data structure details for the `<result>` tag follow the example.

Example XML — RLSMAPPL PROMOTE LIST Reply

```
<?xml version="1.0"?>
<service name="RLSMAPPL">
 <scope name="PROMOTE">
  <message name="LIST">
   <result>
    <release>S4711010</release>
    <releaseArea>ACCTPAY</releaseArea>
    <applName>ACTP</applName>
    <promotionSiteName>SERT4P1</promotionSiteName>
    <promotionName>S4P1UT</promotionName>
    <promotionLevel>10</promotionLevel>
    <areaBlockedRequired>N</areaBlockedRequired>
    <areaCheckoffApprovedRequired>N</areaCheckoffApprovedRequired>
    <areaDemoteRequiredOnGet>N</areaDemoteRequiredOnGet>
    <areaCheckinApprovedRequired>N</areaCheckinApprovedRequired>
   </result>
   <result>
    <release>S4711010</release>
    <releaseArea>FINANCE</releaseArea>
    <applName>ACTP</applName>
    <promotionSiteName>SERT4P2</promotionSiteName>
    <promotionName>S4P2AT</promotionName>
    <promotionLevel>30</promotionLevel>
    <areaBlockedRequired>N</areaBlockedRequired>
    <areaCheckoffApprovedRequired>N</areaCheckoffApprovedRequired>
    <areaDemoteRequiredOnGet>N</areaDemoteRequiredOnGet>
    <areaCheckinApprovedRequired>N</areaCheckinApprovedRequired>
   </result>
   .
   .
   .
   <response>
    <statusMessage>CMR8700I - Area Promotion service completed</statusMessage>
```

```
    <statusReturnCode>00</statusReturnCode>
    <statusReasonCode>8700</statusReasonCode>
   </response>
  </message>
 </scope>
</service>
```

RLSMAPPL PROMOTE LIST <result> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<applName>	Optional	0 - 1	String (4), variable	Release application name.
<areaBlockedRequired>	Optional	0 - 1	String (1)	**Y** = Release area block required. **N** = Release area block not required.
<areaCheckinApprovedRequired>	Optional	0 - 1	String (1)	**Y** = Release area checkin approval required. **N** = Release area checkin approval not required.
<areaCheckoffApprovedRequired>	Optional	0 - 1	String (1)	**Y** = Release area checkoff approval required. **N** = Release area checkoff approval not required.
<areaDemoteRequiredOnGet>	Optional	0 - 1	String (1)	**Y** = Demotion required for release area retrieve. **N** = Demotion not required for release area retrieve.
<promotionLevel>	Optional	0 - 1	Integer (2)	Release promotion level.
<promotionName>	Optional	0 - 1	String (8), variable	Release promotion name.
<promotionSiteName>	Optional	0 - 1	String (8), variable	Release promotion site name.
<release>	Optional	0 - 1	String (8), variable	Release name.
<releaseArea>	Optional	0 - 1	String (8), variable	Release area name.

RLSMAPPL SERVICE LIST

The RLSMAPPL SERVICE LIST message lists the application release status for a named release.

The XML service/scope/message tags and attributes for this message are:

```
<service name="RLSMAPPL">
<scope name="SERVICE">
<message name="LIST">
```

These tags appear in both requests and replies.

RLSMAPPL SERVICE LIST — Request

The following example shows how you might code a request to list the release status for all applications in a named release. Data structure details for the <request> tag follow the example.

Example XML — RLSMAPPL SERVICE LIST Request

```
<?xml version="1.0"?>
<service name="RLSMAPPL">
 <scope name="SERVICE">
  <message name="LIST">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
    <release>S4711010</release>
   </request>
  </message>
 </scope>
</service>
```

RLSMAPPL SERVICE LIST <request> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<applName>	Optional	0 - 1	String (4), variable	Release application name.
<release>	Required	1	String (8), variable	Release name.

RLSMAPPL SERVICE LIST — Reply

The XML reply to a RLSMAPPL SERVICE LIST request returns zero to many <result> data elements. Each <result> contains status flags and other information for an application in the release.

The standard <response> data element follows any <result> tags in the reply and indicates the success or failure of the request. Successful requests have a return code of 00. Unsuccessful requests have a return code of 04 or higher. Because it is the final data element in the XML reply message, the <response> tag serves as an end-of-list marker.

The following example shows what the reply message might look like. Data structure details for the <result> tag follow the example.

Example XML — RLSMAPPL SERVICE LIST Reply

```
<?xml version="1.0"?>
<service name="RLSMAPPL">
 <scope name="SERVICE">
  <message name="LIST">
   <result>
```

```
    <release>S4711010</release>
    <applName>ACTP</applName>
    <applDesc>ACTP Accounts Payable (Base ZMF)</applDesc>
    <applDateJoined>20111005</applDateJoined>
    <applTimeJoined>173838</applTimeJoined>
    <isApplInstalled>N</isApplInstalled>
    <isApplBackedOut>N</isApplBackedOut>
    <allApplsRelated>N</allApplsRelated>
    <allReleaseApplsRelated>N</allReleaseApplsRelated>
    <allBaselinesShared>N</allBaselinesShared>
    <isApplLibsDefined>Y</isApplLibsDefined>
    <isApplSyslibsDefined>Y</isApplSyslibsDefined>
    <isApplPromotionDefined>Y</isApplPromotionDefined>
    <relatedApplCount>00001</relatedApplCount>
    <relatedApplName>COMM</relatedApplName>
   </result>
   <response>
    <statusMessage>CMR8700I - Release Application service completed
     </statusMessage>
    <statusReturnCode>00</statusReturnCode>
    <statusReasonCode>8700</statusReasonCode>
   </response>
  </message>
 </scope>
</service>
```

RLSMAPPL SERVICE LIST <result> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<allApplsRelated>	Optional	0 - 1	String (1)	**Y** = All base applications are related. **N** = All base applications are not related.
<allBaselinesShared>	Optional	0 - 1	String (1)	**Y** = All applications share baselines. **N** = All applications do not share baselines.
<allReleaseApplsRelated>	Optional	0 - 1	String (1)	**Y** = All ERO applications are related. **N** = All ERO applications are not related.
<applDateJoined>	Optional	0 - 1	Date, yyyymmdd	The date that the application was joined to the release.
<applDesc>	Optional	0 - 1	String (44), variable	Application description.
<applName>	Optional	0 - 1	String (4), variable	Release application name.
<applTimeJoined>	Optional	0 - 1	Time, hhmmss	The time that the application was joined to the release.
<isApplBackedOut>	Optional	0 - 1	String (1)	**Y** = Application is backed out. **N** = Application is not backed out.
<isApplInstalled>	Optional	0 - 1	String (1)	**Y** = Application is installed. **N** = Application is not installed.
<isApplLibsDefined>	Optional	0 - 1	String (1)	**Y** = Application libraries are defined. **N** = Application libraries are not defined.
<isApplPromotionDefined>	Optional	0 - 1	String (1)	**Y** = Application promotion is defined. **N** = Application promotion is not defined.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<isApplSyslibsDefined>	Optional	0 - 1	String (1)	**Y** = Application SYSLIBS are defined. **N** = Application SYSLIBS are not defined.
<relatedApplCount>	Optional	0 - 1	Integer (5)	Count of related applications.
<relatedApplName>	Optional	0 - 487	String (4), variable	Related application name.
<release>	Optional	0 - 1	String (8), variable	Release name.

RLSMAPPL SERVICE RELEASE

The RLSMAPPL SERVICE RELEASE message lists release data for each release to which an application is joined.

The XML service/scope/message tags and attributes for this message are:

```
<service name="RLSMAPPL">
<scope name="SERVICE">
<message name="RELEASE">
```

These tags appear in both requests and replies.

RLSMAPPL SERVICE RELEASE — Request

The following example shows how you might code a request to list release data for an application. Data structure details for the <request> tag follow the example.

Example XML — RLSMAPPL SERVICE RELEASE Request

```
<?xml version="1.0"?>
<service name="RLSMAPPL">
 <scope name="SERVICE">
  <message name="RELEASE">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
    <applName>ACTP</applName>
    <release>S4711010</release>
    <releaseParms> </releaseParms>
   </request>
  </message>
 </scope>
</service>
```

RLSMAPPL SERVICE RELEASE <request> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<applName>	Required	1	String (4), variable	Release application name.
<release>	Required	0 - 1	String (8), variable	Release name.
<releaseParms>	Optional	0 - 1	String (1), variable	Release Parms (A or blank).

RLSMAPPL SERVICE RELEASE — Reply

The XML reply to a RLSMAPPL SERVICE RELEASE request returns zero to many `<result>` data elements. Each `<result>` contains information for a release to which the application is joined.

The standard `<response>` data element follows any `<result>` tags in the reply and indicates the success or failure of the request. Successful requests have a return code of 00. Unsuccessful requests have a return code of 04 or higher. Because it is the final data element in the XML reply message, the `<response>` tag serves as an end-of-list marker.

The following example shows what the reply message might look like. Data structure details for the `<result>` tag follow the example.

Example XML — RLSMAPPL SERVICE RELEASE Reply

```
<?xml version="1.0"?>
<service name="RLSMAPPL">
 <scope name="SERVICE">
  <message name="RELEASE">
   <result>
    <release>S4712010</release>
    <applName>ACTP</applName>
    <applDesc>ACTP Accounts Payable (Base ZMF)</applDesc>
    <releaseDesc>Financial Accounting Release S4.V712 #010</releaseDesc>
    <releaseStatus>DEV</releaseStatus>
    <releaseFromInstallDate>20121005</releaseFromInstallDate>
    <releaseFromInstallTime>000100</releaseFromInstallTime>
    <releaseToInstallDate>20121231</releaseToInstallDate>
    <releaseToInstallTime>235959</releaseToInstallTime>
    <releaseDateCreated>20120312</releaseDateCreated>
    <releaseTimeCreated>123024</releaseTimeCreated>
    <releaseCreator>KCAMPBE</releaseCreator>
    <releaseSchedulerCmn>Y</releaseSchedulerCmn>
    <releaseSchedulerManual>Y</releaseSchedulerManual>
    <releaseSchedulerOther>Y</releaseSchedulerOther>
    <releaseSchedulerType>MANUAL</releaseSchedulerType>
    <applDateJoined>20111005</applDateJoined>
    <applTimeJoined>173838</applTimeJoined>
    <isApplInstalled>N</isApplInstalled>
    <isApplBackedOut>N</isApplBackedOut>
   </result>
   <result>
    <release>S4711010</release>
    <applName>ACTP</applName>
```

```
 <applDesc>ACTP Accounts Payable (Base ZMF)</applDesc>
 <releaseDesc>Financial Accounting Release S4.V711 #010</releaseDesc>
 <releaseStatus>DEV</releaseStatus>
 <releaseFromInstallDate>20131005</releaseFromInstallDate>
 <releaseFromInstallTime>000100</releaseFromInstallTime>
 <releaseToInstallDate>20131231</releaseToInstallDate>
 <releaseToInstallTime>235959</releaseToInstallTime>
 <releaseDateCreated>20111005</releaseDateCreated>
 <releaseTimeCreated>203509</releaseTimeCreated>
 <releaseCreator>KCAMPBE</releaseCreator>
 <releaseSchedulerCmn>Y</releaseSchedulerCmn>
 <releaseSchedulerManual>Y</releaseSchedulerManual>
 <releaseSchedulerOther>Y</releaseSchedulerOther>
 <releaseSchedulerType>MANUAL</releaseSchedulerType>
 <applDateJoined>20111005</applDateJoined>
 <applTimeJoined>173838</applTimeJoined>
 <isApplInstalled>N</isApplInstalled>
 <isApplBackedOut>N</isApplBackedOut>
</result>
.
.
.
<response>
  <statusMessage>CMR8700I - Appl Release Search service completed
  </statusMessage>
 <statusReturnCode>00</statusReturnCode>
 <statusReasonCode>8700</statusReasonCode>
</response>
</message>
</scope>
</service>
```

RLSMAPPL SERVICE RELEASE <result> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<applDateJoined>	Optional	0 - 1	Date, yyyymmdd	The date that the application was joined to the release.
<applDesc>	Optional	0 - 1	String (72), variable	Application description.
<applName>	Optional	0 - 1	String (4), variable	Application name.
<applTimeJoined>	Optional	0 - 1	Time, hhmmss	The time that the application was joined to the release.
<isApplBackedOut>	Optional	0 - 1	String (1)	**Y** = Application is backed out. **N** = Application is not backed out.
<isApplInstalled>	Optional	0 - 1	String (1)	**Y** = Application is installed. **N** = Application is not installed.
<release>	Optional	0 - 1	String (8), variable	Release name.
<releaseCreator>	Optional	0 - 1	String (8), variable	The user ID of the person who created the release.
<releaseDateCreated>	Optional	0 - 1	Date, yyyymmdd	The date that the release was created.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<releaseDesc>	Optional	0 - 1	String (72), variable	Release description.
<releaseFromInstallDate>	Optional	0 - 1	Date, yyyymmdd	Start date of the install date range for the release.
<releaseFromInstallTime>	Optional	0 - 1	Time, hhmmss	Start time of the install date/time range for the release.
<ReleaseSchedulerCmn>	Optional	0 - 1	String (1)	**Y** = Allow CMN internal release scheduler. **N** = Do not allow CMN internal release scheduler.
<ReleaseSchedulerManual>	Optional	0 - 1	String (1)	**Y** = Allow manual release scheduler. **N** = Do not allow manual release scheduler.
<ReleaseSchedulerOther>	Optional	0 - 1	String (1)	**Y** = Allow other release scheduler. **N** = Do not allow other release scheduler.
<ReleaseSchedulerType>	Optional	0 - 1	String (8), variable	Scheduler type: **CMN** = The installation jobs are submitted by the ChangeMan ZMF started task at the scheduled install date and time. **Manual** = The installation jobs are submitted as soon as the package approvals are complete. **Other** = The installation jobs are inserted into a third party scheduler via a batch job.
<releaseStatus>	Optional	0 - 1	String (3)	Release status: **APR** = Approved. **BAK** = Backed out. **BAS** = Baselined. **BLK** = Blocked. **DEL** = Deleted. **DEV** = In development. **DIS** = Distributed. **INS** = Installed. **REJ** = Rejected.
<releaseTimeCreated>	Optional	0 - 1	Time, hhmmss	The time that the release was created.
<releaseToInstallDate>	Optional	0 - 1	Date, yyyymmdd	End date of the install date range for the release.
<releaseToInstallTime>	Optional	0 - 1	Time, hhmmss	End time of the install date/time range for the release.

RLSMAPPL SYSLIB LIST

The RLSMAPPL SYSLIB LIST message lists SYSLIB data for release applications.

The XML service/scope/message tags and attributes for this message are:

```
<service name="RLSMAPPL">
<scope name="SYSLIB">
<message name="LIST">
```

These tags appear in both requests and replies.

RLSMAPPL SYSLIB LIST — Request

The following example shows how you might code a request to list SYSLIB data for all of the applications in a release. Data structure details for the <request> tag follow the example.

Example XML — RLSMAPPL SYSLIB LIST Request

```
<?xml version="1.0"?>
<service name="RLSMAPPL">
 <scope name="SYSLIB">
  <message name="LIST">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
    <release>S4711010</release>
   </request>
  </message>
 </scope>
</service>
```

RLSMAPPL SYSLIB LIST <request> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<buildProc>	Optional	0 - 1	String (8), variable	SYSLIB procedure name.
<language>	Optional	0 - 1	String (8), variable	SYSLIB language name.
<libType>	Optional	0 - 1	String (3), variable	SYSLIB release library type.
<release>	Required	1	String (8), variable	Release name.
<releaseApplName>	Optional	0 - 1	String (4), variable	Release application name.

RLSMAPPL SYSLIB LIST — Reply

The XML reply to a RLSMAPPL SYSLIB LIST request returns zero to many <result> data elements. Each <result> contains SYSLIB data for a release application.

The standard <response> data element follows any <result> tags in the reply and indicates the success or failure of the request. Successful requests have a return code of

00. Unsuccessful requests have a return code of 04 or higher. Because it is the final data element in the XML reply message, the `<response>` tag serves as an end-of-list marker.

The following example shows what the reply message might look like. Data structure details for the `<result>` tag follow the example.

Example XML — RLSMAPPL SYSLIB LIST Reply

```
<?xml version="1.0"?>
<service name="RLSMAPPL">
 <scope name="SYSLIB">
  <message name="LIST">
   <result>
    <release>S4711010</release>
    <releaseApplName>ACTP</releaseApplName>
    <language>COBOL2</language>
    <buildProc>CMNCOB2</buildProc>
    <libType>LCT</libType>
    <likeType>K</likeType>
    <isHfsLibType>N</isHfsLibType>
    <applLibTypeCount>00002</applLibTypeCount>
    <libTypeList>
     <relatedLibType>LOD</relatedLibType>
     <libTypeOrderNumber>00000</libTypeOrderNumber>
    </libTypeList>
    <libTypeList>
     <relatedLibType>LOS</relatedLibType>
     <libTypeOrderNumber>00000</libTypeOrderNumber>
    </libTypeList>
   </result>
   <result>
    <release>S4711010</release>
    <releaseApplName>ACTP</releaseApplName>
    <language>COBOL2</language>
    <buildProc>CMNCOB2</buildProc>
    <libType>LOD</libType>
    <likeType>L</likeType>
    <isHfsLibType>N</isHfsLibType>
    <applLibTypeCount>00001</applLibTypeCount>
    <libTypeList>
     <relatedLibType>LOS</relatedLibType>
     <libTypeOrderNumber>00010</libTypeOrderNumber>
    </libTypeList>
   </result>
   .
   .
   .
   <response>
    <statusMessage>CMR8700I - Application SYSLIB service completed
     </statusMessage>
    <statusReturnCode>00</statusReturnCode>
    <statusReasonCode>8700</statusReasonCode>
   </response>
  </message>
 </scope>
</service>
```

RLSMAPPL SYSLIB LIST <result> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<applLibTYpeCount>	Optional	0 - 1	String (5), variable	Count of related application library types.
<buildProc>	Optional	0 - 1	String (8), variable	SYSLIB procedure name.
<isHfsLibType>	Optional	0 - 1	String (1)	**Y** = SYSLIB library type is HFS. **N** = SYSLIB library type is not HFS.
<language>	Optional	0 - 1	String (8), variable	SYSLIB language name.
<libType>	Optional	0 - 1	String (3), variable	SYSLIB library type.
<libTypeList>	Optional	0 - 48	Complex	Related library type information. Each occurrence contains the following subtags: <relatedLibType> <libTypeOrderNumber>).
<relatedLibType>	Optional	0 - 1	String (3), variable	Related library type. Subtag of <libTypeList>.
<libTypeOrderNumber>	Optional	0 - 1	Integer	Order number of related library type. Subtag of <libTypeList>.
<likeType>	Optional	0 - 1	String (1)	"Like Type" of SYSLIB library type. Values for all library types: **C** = Copy **K** = LCT (link control cards) **L** = Load **N** = NCAL **O** = Object Additional values for HFS library types: **P** = PDS **S** = Source **X** = List blank = other
<release>	Optional	0 - 1	String (8), variable	Release name.
<releaseApplName>	Optional	0 - 1	String (4), variable	Release application name.

RLSMAPPR AREA LIST

The RLSMAPPR AREA LIST message lists data for the approval entities in a named release and release area.

The XML service/scope/message tags and attributes for this message are:

```
<service name="RLSMAPPR">
<scope name="AREA">
<message name="LIST">
```

These tags appear in both requests and replies.

RLSMAPPR AREA LIST — Request

The following example shows how you might code a request to list the data for all of the approval entities in a release/release area. Data structure details for the <request> tag follow the example.

Example XML — RLSMAPPR AREA LIST Request

```
<?xml version="1.0"?>
<service name="RLSMAPPR">
 <scope name="AREA">
  <message name="LIST">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
    <release>S4711010</release>
    <releaseArea>FINANCE </releaseArea>
   </request>
  </message>
 </scope>
</service>
```

RLSMAPPR AREA LIST <request> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<release>	Required	1	String (8), variable	Release name.
<releaseApprovalEntity>	Optional	0 - 1	String (8), variable	Release approval entity.
<releaseArea>	Required	1	String (8), variable	Release area name.

RLSMAPPR AREA LIST — Reply

The XML reply to a RLSMAPPR AREA LIST request returns zero to many <result> data elements. Each <result> contains information for an approval entity in the release/ release area.

The standard <response> data element follows any <result> tags in the reply and indicates the success or failure of the request. Successful requests have a return code of 00. Unsuccessful requests have a return code of 04 or higher. Because it is the final data element in the XML reply message, the <response> tag serves as an end-of-list marker.

The following example shows what the reply message might look like. Data structure details for the <result> tag follow the example.

Example XML — RLSMAPPR AREA LIST Reply

```
<?xml version="1.0"?>
<service name="RLSMAPPR">
 <scope name="AREA">
  <message name="LIST">
   <result>
    <releaseApprovalEntity>ACTPLEAD</releaseApprovalEntity>
    <release>S4711010</release>
    <releaseArea>ACCTPAY</releaseArea>
    <releaseApproverDesc>Financial Accounting Manager</releaseApproverDesc>
    <releaseApprovalOrder>0010</releaseApprovalOrder>
    <isApproverRelatedToLibType>N</isApproverRelatedToLibType>
    <isApproverRelatedToRemoteSite>N</isApproverRelatedToRemoteSite>
    <isApproverRelatedToAppl>N</isApproverRelatedToAppl>
    <isReleaseInstallApprover>N</isReleaseInstallApprover>
    <isReleaseAreaChkInApprover>N</isReleaseAreaChkInApprover>
    <isReleaseAreaChkOffApprover>Y</isReleaseAreaChkOffApprover>
    <isReleaseRelatedApprover>N</isReleaseRelatedApprover>
    <isReleaseApproverNotified>N</isReleaseApproverNotified>
    <releaseApproverListCount>0001</releaseApproverListCount>
    <notificationListLength>00000045</notificationListLength>
    <notification>
     <notifierType>1</notifierType>
     <userList>KCAMPBE</userList>
    </notification>
   </result>
   <result>
    <releaseApprovalEntity>ACCTPAY</releaseApprovalEntity>
    <release>S4711010</release>
    <releaseArea>ACCTPAY</releaseArea>
    <releaseApproverDesc>Accounts Payable Manager</releaseApproverDesc>
    <releaseApprovalOrder>0020</releaseApprovalOrder>
    <isApproverRelatedToLibType>N</isApproverRelatedToLibType>
    <isApproverRelatedToRemoteSite>N</isApproverRelatedToRemoteSite>
    <isApproverRelatedToAppl>N</isApproverRelatedToAppl>
    <isReleaseInstallApprover>N</isReleaseInstallApprover>
    <isReleaseAreaChkInApprover>N</isReleaseAreaChkInApprover>
    <isReleaseAreaChkOffApprover>Y</isReleaseAreaChkOffApprover>
    <isReleaseRelatedApprover>N</isReleaseRelatedApprover>
    <isReleaseApproverNotified>N</isReleaseApproverNotified>
    <releaseApproverListCount>0001</releaseApproverListCount>
    <notificationListLength>00000045</notificationListLength>
    <notification>
     <notifierType>1</notifierType>
     <userList>KCAMPBE</userList>
    </notification>
   </result>
   .
   .
   .
   <response>
    <statusMessage>CMR8700I - LIST Approver service completed</statusMessage>
    <statusReturnCode>00</statusReturnCode>
    <statusReasonCode>8700</statusReasonCode>
   </response>
  </message>
 </scope>
</service>
```

RLSMAPPR AREA LIST <result> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<isApproverRelatedToAppl>	Optional	0 - 1	String (1)	**Y** = Approver is associated with application. **N** = Approver is not associated with application.
<isApproverRelatedToLibType>	Optional	0 - 1	String (1)	**Y** = Approver is associated with library type. **N** = Approver is not associated with library type.
<isApproverRelatedToRemoteSite>	Optional	0 - 1	String (1)	**Y** = Approver is associated with remote site. **N** = Approver is not associated with remote site.
<isReleaseApprovedOrRejected>	Optional	0 - 1	String (1)	**A** = Release area is approved. **R** = Release area is rejected.
<isReleaseApproverNotified>	Optional	0 - 1	String (1)	**Y** = Approver is notified. **N** = Approver is not notified.
<isReleaseAreaChkInApprover>	Optional	0 - 1	String (1)	**Y** = Release area checkin approver. **N** = Not a release area checkin approver.
<isReleaseAreaChkOffApprover>	Optional	0 - 1	String (1)	**Y** = Release area check-off approver. **N** = Not a release area check-off approver.
<isReleaseInstallApprover>	Optional	0 - 1	String (1)	**Y** = Release install approver. **N** = Not a release install approver.
<isReleaseRelatedApprover>	Optional	0 - 1	String (1)	**Y** = Approver is associated with release. **N** = Approver is not associated with release.
<notification>	Optional	0 - 40	Complex	Notification vehicle and list of users to receive messages. Each occurrence contains the following subtags: <notifierType> <userList>).
<notifierType>	Optional	0 - 1	String (1)	Method used for sending messages. Subtag of <notification>. **1** = MVS/TSO send (MVSSEND) **4** = Email via SMTP (EMAIL) **5** = Email via Sernet and ECP web server (SERNET) **6** = Batch job (BATCH)
<userList>	Optional	0 - 1	String (44), variable	List of user IDs to be notified. Subtag of <notification>.
<notificationListLength>	Optional	0 - 1	String (8), variable	Length of <notification> tag group.
<reasons>	Optional	0 - 10	String (72), variable	Release reject reasons.
<release>	Optional	0 - 1	String (8), variable	Release name.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<releaseApprovalEntity>	Optional	0 - 1	String (8), variable	Release approval entity.
<releaseApprovalOrder>	Optional	0 - 1	String (4), variable	Hierarchical order of approver notifications.
<releaseApproverDesc>	Optional	0 - 1	String (44), variable	Release approver description.
<releaseApproverListCount>	Optional	0 - 1	String (4), variable	Number of approver notifications.
<releaseApproverUserid>	Optional	0 - 1	String (8), variable	User ID of user who approved/rejected release.
<releaseArea>	Optional	0 - 1	String (8), variable	Release area name.
<releaseDateApproved>	Optional	0 - 1	Date, yyyymmdd	Release approved/rejected date.
<releaseTimeApproved>	Optional	0 - 1	Time, hhmmss	Release approved/rejected time.

RLSMAPPR ASCAPPRV LIST

The RLSMAPPR ASCAPPRV LIST message lists the items that are associated with an associated approval entity.

The XML service/scope/message tags and attributes for this message are:

```
<service name="RLSMAPPR">
<scope name="ASCAPPRV">
<message name="LIST">
```

These tags appear in both requests and replies.

RLSMAPPR ASCAPPRV LIST — Request

The following example shows how you might code a request to list the associated items for an associated approval entity. Data structure details for the `<request>` tag follow the example.

Example XML — RLSMAPPR ASCAPPRV LIST Request

```
<?xml version="1.0"?>
<service name="RLSMAPPR">
 <scope name="ASCAPPRV">
  <message name="LIST">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
```

```
    <releaseApprovalEntity>DBA     </releaseApprovalEntity>
   </request>
  </message>
 </scope>
</service>
```

RLSMAPPR ASCAPPRV LIST <request> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<releaseApprovalEntity>	Required	1	String (8), variable	Release approval entity.

RLSMAPPR ASCAPPRV LIST — Reply

The XML reply to a RLSMAPPR ASCAPPRV LIST request returns one `<result>` data element, which contains the items that are associated with the approval entity.

The standard `<response>` data element follows any `<result>` tags in the reply and indicates the success or failure of the request. Successful requests have a return code of 00. Unsuccessful requests have a return code of 04 or higher. Because it is the final data element in the XML reply message, the `<response>` tag serves as an end-of-list marker.

The following example shows what the reply message might look like. Data structure details for the `<result>` tag follow the example.

Example XML — RLSMAPPR ASCAPPRV LIST Reply

```
<?xml version="1.0"?>
<service name="RLSMAPPR">
 <scope name="ASCAPPRV">
  <message name="LIST">
   <result>
    <releaseApprovalEntity>DBA</releaseApprovalEntity>
    <relatedLibTypeCount>00009</relatedLibTypeCount>
    <relatedLibType>DBB</relatedLibType>
    <relatedLibType>DBR</relatedLibType>
    <relatedLibType>PKG</relatedLibType>
    <relatedLibType>SPD</relatedLibType>
    <relatedLibType>SPQ</relatedLibType>
    <relatedLibType>STL</relatedLibType>
    <relatedLibType>STP</relatedLibType>
    <relatedLibType>TRG</relatedLibType>
    <relatedLibType>UDF</relatedLibType>
   </result>
   <response>
    <statusMessage>CMR8700I - Associated Approver service completed
    </statusMessage>
    <statusReturnCode>00</statusReturnCode>
    <statusReasonCode>8700</statusReasonCode>
   </response>
  </message>
 </scope>
</service>
```

RLSMAPPR ASCAPPRV LIST <result> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<relatedApplName>	Optional	0 - 1	String (4), variable	Related application name or mask.
<relatedLibType>	Optional	0 - 195	String (3), variable	Related release library type.
<relatedLibTypeCount>	Optional	0 - 1	String (4), variable	Related library type count.
<relatedRelease>	Optional	0 - 1	String (8), variable	Related release name or mask.
<relatedReleaseArea>	Optional	0 - 1	String (8), variable	Related release area name or mask.
<relatedSiteName>	Optional	0 - 1	String (8), variable	Related site name or mask.
<releaseApprovalEntity>	Optional	0 - 1	String (8), variable	Release approval entity.

RLSMAPPR GLOBAL LIST

The RLSMAPPR GLOBAL LIST message lists the global approval data for all approval entities or for a named entity.

The XML service/scope/message tags and attributes for this message are:

```
<service name="RLSMAPPR">
<scope name="GLOBAL">
<message name="LIST">
```

These tags appear in both requests and replies.

RLSMAPPR GLOBAL LIST — Request

There is only one subtag (`<releaseApprovalEntity>`) for this request and it is optional. Omitting the subtag returns the global approval data for all approval entities. The `<request>` tag itself is required even if you are omitting the subtag and it may be coded in either of the following ways:

Long form:

```
<request>
</request>
```

Equivalent short form:

```
<request/>
```

The following example shows how you might code a request to list the global data for a specific approval entity. Data structure details for the `<request>` tag follow the example.

Example XML — RLSMAPPR GLOBAL LIST Request

```
<?xml version="1.0"?>
<service name="RLSMAPPR">
 <scope name="GLOBAL">
  <message name="LIST">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
    <releaseApprovalEntity>FINACCTG</releaseApprovalEntity>
   </request>
  </message>
 </scope>
</service>
```

RLSMAPPR GLOBAL LIST <request> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<releaseApprovalEntity>	Optional	0 - 1	String (8), variable	Release approval entity.

RLSMAPPR GLOBAL LIST — Reply

The XML reply to a RLSMAPPR GLOBAL LIST request returns zero to many `<result>` data elements. Each `<result>` contains global information for an approval entity.

The standard `<response>` data element follows any `<result>` tags in the reply and indicates the success or failure of the request. Successful requests have a return code of 00. Unsuccessful requests have a return code of 04 or higher. Because it is the final data element in the XML reply message, the `<response>` tag serves as an end-of-list marker.

The following example shows what the reply message might look like. Data structure details for the `<result>` tag follow the example.

Example XML — RLSMAPPR GLOBAL LIST Reply

```
<?xml version="1.0"?>
<service name="RLSMAPPR">
 <scope name="GLOBAL">
  <message name="LIST">
   <result>
    <releaseApprovalEntity>FINACCTG</releaseApprovalEntity>
    <releaseApproverDesc>Financial Accounting Manager</releaseApproverDesc>
    <releaseApprovalOrder>0030</releaseApprovalOrder>
    <isApproverRelatedToLibType>N</isApproverRelatedToLibType>
    <isApproverRelatedToRemoteSite>N</isApproverRelatedToRemoteSite>
    <isApproverRelatedToAppl>N</isApproverRelatedToAppl>
    <isReleaseInstallApprover>Y</isReleaseInstallApprover>
    <isReleaseAreaChkInApprover>N</isReleaseAreaChkInApprover>
    <isReleaseAreaChkOffApprover>Y</isReleaseAreaChkOffApprover>
    <isReleaseRelatedApprover>N</isReleaseRelatedApprover>
    <approverListCount>0001</approverListCount>
```

```
    <notificationListLength>00000045</notificationListLength>
    <notification>
     <notifierType>1</notifierType>
     <userList>KCAMPBE</userList>
    </notification>
   </result>
   <response>
    <statusMessage>CMR8700I - Global Approvers service completed
     </statusMessage>
    <statusReturnCode>00</statusReturnCode>
    <statusReasonCode>8700</statusReasonCode>
   </response>
  </message>
 </scope>
</service>
```

RLSMAPPR GLOBAL LIST <result> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<approverListCount>	Optional	0 - 1	String (4), variable	Number of approver notifications.
<isApproverRelatedToAppl>	Optional	0 - 1	String (1)	**Y** = Approver is associated with an application. **N** = Approver is not associated with an application.
<isApproverRelatedToLibType>	Optional	0 - 1	String (1)	**Y** = Approver is associated with a library type. **N** = Approver is not associated with a library type.
<isApproverRelatedToRemoteSite>	Optional	0 - 1	String (1)	**Y** = Approver is associated with a remote site. **N** = Approver is not associated with a remote site.
<isReleaseAreaChkInApprover>	Optional	0 - 1	String (1)	**Y** = Release area checkin approver. **N** = Not a release area checkin approver.
<isReleaseAreaChkOffApprover>	Optional	0 - 1	String (1)	**Y** = Release area check-off approver. **N** = Not a release area check-off approver.
<isReleaseInstallApprover>	Optional	0 - 1	String (1)	**Y** = Release install approver. **N** = Not a release install approver.
<isReleaseRelatedApprover>	Optional	0 - 1	String (1)	**Y** = Approver is associated with a release. **N** = Approver is not associated with a release.
<notification>	Optional	0 - 40	Complex	Notification vehicle and list of users to receive messages. Each occurrence contains the following subtags: <notifierType> <userList>).

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<notifierType>	Optional	0 - 1	String (1)	Method used for sending messages. Subtag of <notification>. **1** = MVS/TSO send (MVSSEND) **4** = Email via SMTP (EMAIL) **5** = Email via Sernet and ECP web server (SERNET) **6** = Batch job (BATCH)
<userList>	Optional	0 - 1	String (44), variable	List of user IDs to be notified. Subtag of <notification>.
<notificationListLength>	Optional	0 - 1	String (8), variable	Length of <notification> tag group.
<releaseApprovalEntity>	Optional	0 - 1	String (8), variable	Release approval entity.
<releaseApprovalOrder>	Optional	0 - 1	String (4), variable	Hierarchical order of approval notifications.
<releaseApproverDesc>	Optional	0 - 1	String (44), variable	Release approver description.

RLSMAPPR RELEASE LIST

The RLSMAPPR RELEASE LIST message lists the data for install approval entities for a named release.

The XML service/scope/message tags and attributes for this message are:

```
<service name="RLSMAPPR">
<scope name="RELEASE">
<message name="LIST">
```

These tags appear in both requests and replies.

RLSMAPPR RELEASE LIST — Request

The following example shows how you might code a request to list the data for all of the install approvers for a release. Data structure details for the `<request>` tag follow the example.

Example XML — RLSMAPPR RELEASE LIST Request

```
<?xml version="1.0"?>
<service name="RLSMAPPR">
 <scope name="RELEASE">
  <message name="LIST">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
```

```
    <release>S4711010</release>
   </request>
  </message>
 </scope>
</service>
```

RLSMAPPR RELEASE LIST <request> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<release>	Required	1	String (8), variable	Release name.
<releaseApprovalEntity>	Optional	0 - 1	String (8), variable	Release approval entity.

RLSMAPPR RELEASE LIST — Reply

The XML reply to a RLSMAPPR RELEASE LIST request returns zero to many `<result>` data elements. Each `<result>` contains information for an install approval entity.

The standard `<response>` data element follows any `<result>` tags in the reply and indicates the success or failure of the request. Successful requests have a return code of 00. Unsuccessful requests have a return code of 04 or higher. Because it is the final data element in the XML reply message, the `<response>` tag serves as an end-of-list marker.

The following example shows what the reply message might look like. Data structure details for the `<result>` tag are identical to those for the RLSMAPPR AREA LIST message except that there is no `<releaseArea>` tag - see "RLSMAPPR AREA LIST <request> Data Structure" on page 31.

Example XML — RLSMAPPR RELEASE LIST Reply

```
<?xml version="1.0"?>
<service name="RLSMAPPR">
 <scope name="RELEASE">
  <message name="LIST">
   <result>
    <releaseApprovalEntity>FINACCTG</releaseApprovalEntity>
    <release>S4711010</release>
    <releaseApproverDesc>Financial Accounting Manager</releaseApproverDesc>
    <releaseApprovalOrder>0030</releaseApprovalOrder>
    <isApproverRelatedToLibType>N</isApproverRelatedToLibType>
    <isApproverRelatedToRemoteSite>N</isApproverRelatedToRemoteSite>
    <isApproverRelatedToAppl>N</isApproverRelatedToAppl>
    <isReleaseInstallApprover>Y</isReleaseInstallApprover>
    <isReleaseAreaChkInApprover>N</isReleaseAreaChkInApprover>
    <isReleaseAreaChkOffApprover>N</isReleaseAreaChkOffApprover>
    <isReleaseRelatedApprover>N</isReleaseRelatedApprover>
    <isReleaseApproverNotified>N</isReleaseApproverNotified>
    <releaseApproverListCount>0001</releaseApproverListCount>
    <notificationListLength>00000045</notificationListLength>
    <notification>
     <notifierType>1</notifierType>
     <userList>KCAMPBE</userList>
    </notification>
   </result>
```

```
   <result>
    <releaseApprovalEntity>RLSEMNGR</releaseApprovalEntity>
    <release>S4711010</release>
    <releaseApproverDesc>Release Manager</releaseApproverDesc>
    <releaseApprovalOrder>0030</releaseApprovalOrder>
    <isApproverRelatedToLibType>N</isApproverRelatedToLibType>
    <isApproverRelatedToRemoteSite>N</isApproverRelatedToRemoteSite>
    <isApproverRelatedToAppl>N</isApproverRelatedToAppl>
    <isReleaseInstallApprover>Y</isReleaseInstallApprover>
    <isReleaseAreaChkInApprover>N</isReleaseAreaChkInApprover>
    <isReleaseAreaChkOffApprover>N</isReleaseAreaChkOffApprover>
    <isReleaseRelatedApprover>N</isReleaseRelatedApprover>
    <isReleaseApproverNotified>N</isReleaseApproverNotified>
    <releaseApproverListCount>0001</releaseApproverListCount>
    <notificationListLength>00000045</notificationListLength>
    <notification>
     <notifierType>1</notifierType>
     <userList>KCAMPBE</userList>
    </notification>
   </result>
   .
   .
   .
   <response>
    <statusMessage>CMR8700I - LIST Approver service completed</statusMessage>
    <statusReturnCode>00</statusReturnCode>
    <statusReasonCode>8700</statusReasonCode>
   </response>
  </message>
 </scope>
</service>
```

RLSMAREA ALL_CHK SYSLIB

The RLSMAREA ALL_CHK SYSLIB message lists the COPYLIB, LOADLIB, and source concatenation lists for libraries that are allocated.

The XML service/scope/message tags and attributes for this message are:

```
<service name="RLSMAREA">
<scope name="ALL_CHK">
<message name="SYSLIB">
```

These tags appear in both requests and replies.

RLSMAREA ALL_CHK SYSLIB — Request

The following example shows how you might code a request to list the COPYLIB, LOADLIB, and source concatenation lists for allocated libraries in a release application. Data structure details for the `<request>` tag follow the example.

Example XML — RLSMAREA ALL_CHK SYSLIB Request

```
<?xml version="1.0"?>
<service name="RLSMAREA">
 <scope name="ALL_CHK">
  <message name="SYSLIB">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
    <release>S4711010</release>
    <releaseArea>ACCTPAY </releaseArea>
    <releaseApplName>ACTP</releaseApplName>
    <language>COBOL2  </language>
    <buildProc>CMNCOB2 </buildProc>
    <libType>LOD</libType>
    <package>          </package>
   </request>
  </message>
 </scope>
</service>
```

RLSMAREA ALL_CHK SYSLIB <request> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<buildProc>	Required	1	String (8), variable	Release application procedure name.
<language>	Required	1	String (8), variable	Release application language name.
<libType>	Required	1	String (3), variable	Release application library type.
<package>	Optional	0 - 1	String (10), variable	Release application package name.
<release>	Required	1	String (8), variable	Release name.
<releaseApplName>	Required	1	String (4), variable	Release application name.
<releaseArea>	Required	1	String (8), variable	Release area name.

RLSMAREA ALL_CHK SYSLIB — Reply

The XML reply to a RLSMAREA ALL_CHK SYSLIB request returns zero to many `<result>` data elements. Each result contains SYSLIB data for each allocated library type in a release application.

The standard `<response>` data element follows any `<result>` tags in the reply and indicates the success or failure of the request. Successful requests have a return code of 00. Unsuccessful requests have a return code of 04 or higher. Because it is the final data element in the XML reply message, the `<response>` tag serves as an end-of-list marker.

The following example shows what the reply message might look like. Data structure details for the <result> tag follow the example.

Example XML — RLSMAREA ALL_CHK SYSLIB Reply

```
<?xml version="1.0"?>
<service name="RLSMAREA">
 <scope name="ALL_CHK">
  <message name="SYSLIB">
   <result>
    <release>S4711010</release>
    <releaseArea>ACCTPAY</releaseArea>
    <releaseApplName>ACTP</releaseApplName>
    <libType>LOS</libType>
    <likeType>N</likeType>
    <concatType>L</concatType>
    <libraryFromType>B</libraryFromType>
    <library>CMNTP.S4.V711.BASE.ACTP.LOS</library>
    <libraryOrg>PDS</libraryOrg>
   </result>
   <result>
    <release>S4711010</release>
    <releaseArea>ACCTPAY</releaseArea>
    <releaseApplName>COMM</releaseApplName>
    <libType>LOS</libType>
    <likeType>N</likeType>
    <concatType>L</concatType>
    <libraryFromType>B</libraryFromType>
    <library>CMNTP.S4.V711.BASE.COMM.LOS</library>
    <libraryOrg>PDS</libraryOrg>
   </result>
   <result>
    <release>S4711010</release>
    <releaseArea>ACCTPAY</releaseArea>
    <releaseApplName>ACTP</releaseApplName>
    <libType>LOD</libType>
    <likeType>L</likeType>
    <concatType>S</concatType>
    <libraryFromType>B</libraryFromType>
    <library>CMNTP.S4.V711.BASE.ACTP.LOD</library>
    <libraryOrg>PDS</libraryOrg>
   </result>
   <response>
    <statusMessage>SER8209I Logon accepted for user MTULLY; Local CCSID=00037
     </statusMessage>
    <statusReturnCode>00</statusReturnCode>
    <statusReasonCode>8700</statusReasonCode>
   </response>
  </message>
 </scope>
</service>
```

RLSMAREA ALL_CHK SYSLIB <result> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<concatType>	Optional	0 - 1	String (1)	SYSLIB concatenation type. Values: **C** = Copy **L** = Load **S** = Source
<libType>	Optional	0 - 1	String (3), variable	Release application library type.
<library>	Optional	0 - 1	String (1024), variable	Release dataset name.
<libraryFromType>	Optional	0 - 1	String (1)	SYSLIB from type. Values: **B** = Baseline **C** = Current **P** = Prior release **S** = Staging
<libraryOrg>	Optional	0 - 1	String (4), variable	Release dataset organization. Values: **LIB** = Librarian **LIBA** = Librarian archie **PAN** = Panvalet **PDS** = PDS **PDSE** = PDSE
<likeType>	Optional	0 - 1	String (1)	Release application library "like type". Values for all library types: **C** = Copy **K** = LCT (link control cards) **L** = Load **N** = NCAL **O** = Object Additional values for HFS library types: **P** = PDS **S** = Source **X** = List blank = other
<release>	Optional	0 - 1	String (8), variable	Release name.
<releaseApplName>	Optional	0 - 1	String (4), variable	Release application name.
<releaseArea>	Optional	0 - 1	String (8), variable	Release area name.

RLSMAREA ALL_NOC SYSLIB

The RLSMAREA ALL_NOC SYSLIB message lists the COPYLIB, LOADLIB, and source concatenation lists for all libraries, including those that are not yet allocated.

The XML service/scope/message tags and attributes for this message are:

```
<service name="RLSMAREA">
<scope name="ALL_NOC">
<message name="SYSLIB">
```

These tags appear in both requests and replies.

RLSMAREA ALL_NOC SYSLIB — Request

The following example shows how you might code a request to list the COPYLIB, LOADLIB, and source concatenation lists for all libraries in a release application. Data structure details for the <request> tag are identical to those for the RLSMAREA ALL_CHK SYSLIB message - see "RLSMAREA ALL_CHK SYSLIB <request> Data Structure" on page 42.

Example XML — RLSMAREA ALL_NOC SYSLIB Request

```
<?xml version="1.0"?>
<service name="RLSMAREA">
 <scope name="ALL_NOC">
  <message name="SYSLIB">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
    <release>S4711010</release>
    <releaseArea>ACCTPAY </releaseArea>
    <releaseApplName>ACTP</releaseApplName>
    <language>COBOL2  </language>
    <buildProc>CMNCOB2 </buildProc>
    <libType>LOD</libType>
    <package>          </package>
   </request>
  </message>
 </scope>
</service>
```

RLSMAREA ALL_NOC SYSLIB — Reply

The XML reply to a RLSMAREA ALL_NOC SYSLIB request returns zero to many <result> data elements. Each result contains SYSLIB data for each library type in a release application.

The standard <response> data element follows any <result> tags in the reply and indicates the success or failure of the request. Successful requests have a return code of 00. Unsuccessful requests have a return code of 04 or higher. Because it is the final data element in the XML reply message, the <response> tag serves as an end-of-list marker.

The following example shows what the reply message might look like. Data structure details for the <result> tag are identical to those for the RLSMAREA ALL_CHK SYSLIB message - see "RLSMAREA ALL_CHK SYSLIB <result> Data Structure" on page 44.

Example XML — RLSMAREA ALL_NOC SYSLIB Reply

```
<?xml version="1.0"?>
<service name="RLSMAREA">
 <scope name="ALL_NOC">
  <message name="SYSLIB">
   <result>
    <release>S4711010</release>
    <releaseArea>ACCTPAY</releaseArea>
    <releaseApplName>ACTP</releaseApplName>
    <libType>LOS</libType>
    <likeType>N</likeType>
    <concatType>L</concatType>
    <libraryFromType>C</libraryFromType>
    <library>CMNTP.S4711010.ACCTPAY.ACTP.LOS</library>
    <libraryOrg>PDS</libraryOrg>
   </result>
   <result>
    <release>S4711010</release>
    <releaseArea>ACCTPAY</releaseArea>
    <releaseApplName>COMM</releaseApplName>
    <libType>LOS</libType>
    <likeType>N</likeType>
    <concatType>L</concatType>
    <libraryFromType>C</libraryFromType>
    <library>CMNTP.S4711010.ACCTPAY.COMM.LOS</library>
    <libraryOrg>PDS</libraryOrg>
   </result>
   <result>
    <release>S4711010</release>
    <releaseArea>FINANCE</releaseArea>
    <releaseApplName>ACTP</releaseApplName>
    <libType>LOS</libType>
    <likeType>N</likeType>
    <concatType>L</concatType>
    <libraryFromType>C</libraryFromType>
    <library>CMNTP.S4711010.FINANCE.ACTP.LOS</library>
    <libraryOrg>PDS</libraryOrg>
   </result>
   .
   .
   .
   <response>
    <statusMessage>SER8209I Logon accepted for user MTULLY; Local CCSID=00037
     </statusMessage>
    <statusReturnCode>00</statusReturnCode>
    <statusReasonCode>8700</statusReasonCode>
   </response>
  </message>
 </scope>
</service>
```

RLSMAREA CIM LIST

The RLSMAREA CIM LIST message lists release area component in motion (CIM) information from the ERO Db2 CIM table.

The XML service/scope/message tags and attributes for this message are:

```
<service name="RLSMAREA">
<scope name="CIM">
<message name="LIST">
```

These tags appear in both requests and replies.

RLSMAREA CIM LIST — Request

The following example shows how you might code a request to list component in motion information for a release. Data structure details for the `<request>` tag follow the example.

Example XML — RLSMAREA CIM LIST Request

```
<?xml version="1.0"?>
<service name="RLSMAREA">
 <scope name="CIM">
  <message name="LIST">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
    <release>S4712COM</release>
   </request>
  </message>
 </scope>
</service>
```

RLSMAREA CIM LIST <request> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<applName>	Optional	0 - 1	String (4), variable	Release application name.
<buildProc>	Optional	0 - 1	String (8), variable	Component build procedure.
<checkinDescription>	Optional	0 - 1	String (120), variable	Checkin description.
<checkinUser>	Optional	0 - 1	String (8), variable	Component checkin user ID.
<component>	Optional	0 - 1	String (256), variable	Release area component name.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<componentFromChangeDate>	Optional	0 - 1	Date, yyyymmdd	Release checkin "from" date.
<componentGenerateStatus>	Optional	0 - 1	String (6), variable	Component generate status. Values: ACTIVE = Component is active FAILED = Component failed SUBMIT = Component submitted
<componentToChangeDate>	Optional	0 - 1	Date, yyyymmdd	Release checkin "to" date.
<componentType>	Optional	0 - 1	String (3), variable	Component library type.
<language>	Optional	0 - 1	String (8), variable	Component language name.
<package>	Optional	0 - 1	String (10), variable	Package name. *NOTE:* May be masked using asterisk (*) wildcard.
<packageId>	Optional	0 - 1	Integer (6), variable	Package ID.
<release>	Required	1	String (8), variable	Release name.
<releaseArea>	Optional	0 - 1	String (8), variable	Release area name.
<releaseAreaFromCheckinDate>	Optional	0 - 1	Date, yyyymmdd	Release area checkin "from" date.
<releaseAreaFromGenerateDate>	Optional	0 - 1	Date, yyyymmdd	Release area generate "from" date.
<releaseAreaToCheckinDate>	Optional	0 - 1	Date, yyyymmdd	Release area checkin "to" date.
<releaseAreaToGenerateDate>	Optional	0 - 1	Date, yyyymmdd	Release area generate "to" date.

RLSMAREA CIM LIST — Reply

The XML reply to a RLSMAREA CIM LIST request returns zero to many `<result>` data elements. Each result lists a row of component in motion information from the ERO Db2 CIM table.

The standard `<response>` data element follows any `<result>` tags in the reply and indicates the success or failure of the request. Successful requests have a return code of 00. Unsuccessful requests have a return code of 04 or higher. Because it is the final data element in the XML reply message, the `<response>` tag serves as an end-of-list marker.

The following example shows what the reply message might look like. Data structure details for the `<result>` tag follow the example.

Example XML — RLSMAREA CIM LIST Reply

```
<?xml version="1.0"?>
<service name="RLSMAREA">
 <scope name="CIM">
  <message name="LIST">
   <result>
    <release>S4712COM</release>
    <releaseArea>FINANCE</releaseArea>
    <package>ACTP000076</package>
    <applName>ACTP</applName>
    <packageId>000076</packageId>
    <componentType>CPY</componentType>
    <likeType>C</likeType>
    <libraryStorageMeans>PDS</libraryStorageMeans>
    <component>ACPCPYCE</component>
    <checkinUser>KCAMPBE</checkinUser>
    <releaseAreaCheckinDate>20120827</releaseAreaCheckinDate>
    <releaseAreaCheckinTime>140138</releaseAreaCheckinTime>
    <loadModule>N</loadModule>
    <generateRequired>N</generateRequired>
    <componentChangeDate>20120827</componentChangeDate>
    <componentChangeTime>083512</componentChangeTime>
    <setssi>630B28C0</setssi>
    <hashToken>B7A76B0C00000079</hashToken>
    <releaseAreaGenerateDate>19000101</releaseAreaGenerateDate>
    <releaseAreaGenerateTime>000000</releaseAreaGenerateTime>
    <componentVersionNo>01</componentVersionNo>
    <componentModLevel>02</componentModLevel>
    <componentSizeLines>00000000</componentSizeLines>
    <componentSizeBytes>00000000</componentSizeBytes>
    <componentSizeInit>00000001</componentSizeInit>
    <componentCreateDate>20020507</componentCreateDate>
    <componentLmodRENTattr>N</componentLmodRENTattr>
    <componentLmodREUSattr>N</componentLmodREUSattr>
    <componentLmodOVLYattr>N</componentLmodOVLYattr>
    <componentLmodTESTattr>N</componentLmodTESTattr>
    <componentLmodOLODattr>N</componentLmodOLODattr>
    <componentLmodEXECattr>N</componentLmodEXECattr>
    <componentLmodRFRSattr>N</componentLmodRFRSattr>
    <currentHashToken>B7A75D0A00000079</currentHashToken>
    <priorHashToken>B7A75D0A00000079</priorHashToken>
    <currentAssocPkg>BASELINE</currentAssocPkg>
    <priorAssocPkg>BASELINE</priorAssocPkg>
   </result>
   <result>
    <release>S4712COM</release>
    <releaseArea>ACCTPAY</releaseArea>
    <package>ACTP000076</package>
    <applName>ACTP</applName>
    <packageId>000076</packageId>
    <componentType>CPY</componentType>
    <likeType>C</likeType>
    <libraryStorageMeans>PDS</libraryStorageMeans>
    <component>ACPCPYCE</component>
    <checkinUser>KCAMPBE</checkinUser>
    <releaseAreaCheckinDate>20120827</releaseAreaCheckinDate>
    <releaseAreaCheckinTime>135951</releaseAreaCheckinTime>
    <loadModule>N</loadModule>
    <generateRequired>N</generateRequired>
    <componentChangeDate>20120827</componentChangeDate>
```

```
<componentChangeTime>083512</componentChangeTime>
<setssi>630B28C0</setssi>
<hashToken>B7A76B0C00000079</hashToken>
<releaseAreaGenerateDate>19000101</releaseAreaGenerateDate>
<releaseAreaGenerateTime>000000</releaseAreaGenerateTime>
<componentVersionNo>01</componentVersionNo>
<componentModLevel>02</componentModLevel>
<componentSizeLines>00000003</componentSizeLines>
<componentSizeBytes>00000000</componentSizeBytes>
<componentSizeInit>00000001</componentSizeInit>
<componentCreateDate>20020507</componentCreateDate>
<componentLmodRENTattr>N</componentLmodRENTattr>
<componentLmodREUSattr>N</componentLmodREUSattr>
<componentLmodOVLYattr>N</componentLmodOVLYattr>
<componentLmodTESTattr>N</componentLmodTESTattr>
<componentLmodOLODattr>N</componentLmodOLODattr>
<componentLmodEXECattr>N</componentLmodEXECattr>
<componentLmodRFRSattr>N</componentLmodRFRSattr>
<currentHashToken>B7A75D0A00000079</currentHashToken>
<priorHashToken>B7A75D0A00000079</priorHashToken>
<currentAssocPkg>BASELINE</currentAssocPkg>
<priorAssocPkg>BASELINE</priorAssocPkg>
</result>
.
.
.
<response>
<statusMessage>CMR8700I - Release CIM Table service completed
</statusMessage>
<statusReturnCode>00</statusReturnCode>
<statusReasonCode>8700</statusReasonCode>
</response>
</message>
</scope>
</service>
```

RLSMAREA CIM LIST <result> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<applName>	Optional	0 - 1	String (4), variable	Release application name.
<buildProc>	Optional	0 - 1	String (8), variable	Component build procedure.
<checkinDescription>	Optional	0 - 1	String (120), variable	Checkin description.
<checkinUser>	Optional	0 - 1	String (8), variable	Component checkin user ID.
<component>	Optional	0 - 1	String (256), variable	Release area component name.
<componentChangeDate>	Optional	0 - 1	Date, yyyymmdd	Component changed date.
<componentChangeTime>	Optional	0 - 1	Time, hhmmss	Component changed time.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<componentCreateDate>	Optional	0 - 1	Date, yyyymmdd	Non-load component create date.
<componentGenerateStatus>	Optional	0 - 1	String (6), variable	Component generate status. Values: **ACTIVE** = Component is active **FAILED** = Component failed **SUBMIT** = Component submitted
<componentGenerated>	Optional	0 - 1	String (1)	**Y** = Component has been generated. **N** = Component has not been generated.
<componentLmodAliasOf>	Optional	0 - 1	String (8), variable	Load component "alias of" name.
<componentLmodAuth>	Optional	0 - 1	String (1)	**Y** = Load component is authorized. **N** = Load component is not authorized.
<componentLmodEXECattr>	Optional	0 - 1	String (1)	**Y** = Load component is executable. **N** = Load component is not executable.
<componentLmodOLODattr>	Optional	0 - 1	String (1)	**Y** = Load component is only loadable. **N** = Load component may be linked to.
<componentLmodOVLYattr>	Optional	0 - 1	String (1)	**Y** = Load component overlay format. **N** = Not load component overlay format.
<componentLmodRENTattr>	Optional	0 - 1	String (1)	**Y** = Load component is reentrant. **N** = Load component is not reentrant.
<componentLmodREUSattr>	Optional	0 - 1	String (1)	**Y** = Load component is reusable. **N** = Load component is not reusable.
<componentLmodRFRSattr>	Optional	0 - 1	String (1)	**Y** = Load component is refreshable. **N** = Load component is not refreshable.
<componentLmodTESTattr>	Optional	0 - 1	String (1)	**Y** = Load component in test format. **N** = Load component is not in test format.
<componentModLevel>	Optional	0 - 1	Integer (2)	Component modification level.
<componentSizeBytes	Optional	0 - 1	Integer (8)	Component size in bytes.
<componentSizeInit>	Optional	0 - 1	Integer (8)	Non-load component initial size in lines.
<componentSizeLines>	Optional	0 - 1	Integer (8)	Non-load component current size in lines.
<componentType>	Optional	0 - 1	String (3)	Component library type.
<componentVersionNo>	Optional	0 - 1	Integer (2)	Component version number.
<currentAssocPkg>	Optional	0 - 1	String (10)	Current associated package.
<currentHashToken>	Optional	0 - 1	String (16)	Current hash token.
<generateJobName>	Optional	0 - 1	String (8), variable	Component generate job name.
<generateJobNumber>	Optional	0 - 1	String (8), variable	Component generate job number.
<generateRequired>	Optional	0 - 1	String (1)	**Y** = Generate is required. **N** = Generate is not required.
<hashToken>	Optional	0 - 1	String (16)	Release area component hash value.
<language>	Optional	0 - 1	String (8), variable	Component language name.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<libraryStorageMeans>	Optional	0 - 1	String (3)	Release library storage means. Values: **HFS** = HFS **LIB** = Librarian **OTH** = Other **PAN** = Panvalet **PDS** = PDS **SEQ** = Sequential **SRD** = Stacked reverse delta
<likeType>	Optional	0 - 1	String (1)	Component library "like type". Values for all library types: **C** = Copy **K** = LCT (link control cards) **L** = Load **N** = NCAL **O** = Object Additional values for HFS library types: **P** = PDS **S** = Source **X** = List blank = other
<loadModule>	Optional	0 - 1	String (1)	**Y** = Component is a load module. **N** = Component is not a load module.
<package>	Optional	0 - 1	String (10)	Release package name.
<packageId>	Optional	0 - 1	Integer (6)	Release package number.
<priorAssocPkg>	Optional	0 - 1	String (10)	Prior associated package name.
<priorHashToken>	Optional	0 - 1	String (16)	Prior hash token.
<release>	Optiona	0 - 1	String (8), variable	Release name.
<releaseArea>	Optiona	0 - 1	String (8), variable	Release area name.
<releaseAreaCheckinDate>	Optional	0 - 1	Date, yyyymmdd	Release area checkin date.
<releaseAreaCheckinTime>	Optional	0 - 1	Time, hhmmss	Release area checkin time.
<releaseAreaGenerateDate>	Optional	0 - 1	Date, yyyymmdd	Release area generate date.
<releaseAreaGenerateTime>	Optional	0 - 1	Time, hhmmss	Release area generate time.
<setssi>	Optional	0 - 1	String (8)	Setssi for component.
<sourceComponent>	Optional	0 - 1	String (256), variable	Source component name.
<sourceComponentType>	Optional	0 - 1	String (3)	Source component library type.

RLSMAREA CMP_LOCK LIST

The RLSMAREA CMP_LOCK LIST message lists information about release component locks.

The XML service/scope/message tags and attributes for this message are:

```
<service name="RLSMAREA">
<scope name="CMP_LOCK">
<message name="LIST">
```

These tags appear in both requests and replies.

RLSMAREA CMP_LOCK LIST — Request

The following example shows how you might code a request to list component lock information for a release. Data structure details for the `<request>` tag follow the example.

Example XML — RLSMAREA CMP_LOCK LIST Request

```
<?xml version="1.0"?>
<service name="RLSMAREA">
 <scope name="CMP_LOCK">
  <message name="LIST">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
    <release>S4711010</release>
    <lockType>I</lockType>
   </request>
  </message>
 </scope>
</service>
```

RLSMAREA CMP_LOCK LIST <request> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<applName>	Optional	0 - 1	String (4), variable	Release application name.
<componentNameMask>	Optional	0 - 1	String (256), variable	Component name or mask.
<componentTypeMask>	Optional	0 - 1	String (3), variable	Component type or mask.
<lockType>	Required	1	String (1)	Component lock type. Values: **I** = Checkin lock
<release>	Required	1	String (8), variable	Release name.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<releaseArea>	Optional	0 - 1	String (8), variable	Release area name.
<user>	Optional	0 - 1	String (8), variable	User ID of user who has lock on component.

RLSMAREA CMP_LOCK LIST — Reply

The XML reply to a RLSMAREA CMP_LOCK LIST request returns zero to many `<result>` data elements. Each result contains information for a locked component.

The standard `<response>` data element follows any `<result>` tags in the reply and indicates the success or failure of the request. Successful requests have a return code of 00. Unsuccessful requests have a return code of 04 or higher. Because it is the final data element in the XML reply message, the `<response>` tag serves as an end-of-list marker.

The following example shows what the reply message might look like if no component locks are found. Data structure details for the `<result>` tag follow the example.

Example XML — RLSMAREA CMP_LOCK LIST Reply

```
<?xml version="1.0"?>
<service name="RLSMAREA">
 <scope name="CMP_LOCK">
  <message name="LIST">
   <response>
    <statusMessage>CMR6504I - No information found for Component locks
      request.</statusMessage>
    <statusReturnCode>08</statusReturnCode>
    <statusReasonCode>6504</statusReasonCode>
   </response>
  </message>
 </scope>
</service>
```

RLSMAREA CMP_LOCK LIST <result> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<applName>	Optional	0 - 1	String (4), variable	Release application name.
<componentName>	Optional	0 - 1	String (256), variable	Component name.
<componentType>	Optional	0 - 1	String (3), variable	Component type.
<lockDate>	Optional	0 - 1	Date, yyyymmdd	Component lock date.
<lockTime>	Optional	0 - 1	Time, hhmmss	Component lock time.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<lockType>	Optional	0 - 1	String (1)	Component lock type. Values: **I** = Checkin lock
<release>	Optional	0 - 1	String (8), variable	Release name.
<releaseArea>	Optional	0 - 1	String (8), variable	Release area name.
<user>	Optional	0 - 1	String (8), variable	User ID of user who has lock on component.

RLSMAREA CPY SYSLIB

The RLSMAREA CPY SYSLIB message lists the COPYLIB concatenation for a release application.

The XML service/scope/message tags and attributes for this message are:

```
<service name="RLSMAREA">
<scope name="CPY">
<message name="SYSLIB">
```

These tags appear in both requests and replies.

RLSMAREA CPY SYSLIB — Request

The following example shows how you might code a request to list the COPYLIB concatenation for a release application. Data structure details for the `<request>` tag are identical to those for the RLSMAREA ALL_CHK SYSLIB message - see "RLSMAREA ALL_CHK SYSLIB <request> Data Structure" on page 42.

Example XML — RLSMAREA CPY SYSLIB Request

```
<?xml version="1.0"?>
<service name="RLSMAREA">
 <scope name="CPY">
  <message name="SYSLIB">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
    <release>S4711010</release>
    <releaseArea>ACCTPAY </releaseArea>
    <releaseApplName>ACTP</releaseApplName>
    <language>COBOL2  </language>
    <buildProc>CMNCOB2 </buildProc>
    <libType>SRC</libType>
    <package>          </package>
   </request>
  </message>
```

```
 </scope>
</service>
```

RLSMAREA CPY SYSLIB — Reply

The XML reply to a RLSMAREA CPY SYSLIB request returns zero to many <result> data elements. Each result contains COPYLIB information data for a release application.

The standard <response> data element follows any <result> tags in the reply and indicates the success or failure of the request. Successful requests have a return code of 00. Unsuccessful requests have a return code of 04 or higher. Because it is the final data element in the XML reply message, the <response> tag serves as an end-of-list marker.

The following example shows what the reply message might look like. Data structure details for the <result> tag are identical to those for the RLSMAREA ALL_CHK SYSLIB message - see "RLSMAREA ALL_CHK SYSLIB <result> Data Structure" on page 44.

Example XML — RLSMAREA CPY SYSLIB Reply

```
<?xml version="1.0"?>
<service name="RLSMAREA">
 <scope name="CPY">
  <message name="SYSLIB">
   <result>
    <release>S4711010</release>
    <releaseArea>ACCTPAY</releaseArea>
    <releaseApplName>ACTP</releaseApplName>
    <libType>CPY</libType>
    <likeType>C</likeType>
    <concatType>C</concatType>
    <libraryFromType>B</libraryFromType>
    <library>CMNTP.S4.V711.BASE.ACTP.CPY</library>
    <libraryOrg>PDS</libraryOrg>
   </result>
   <result>
    <release>S4711010</release>
    <releaseArea>ACCTPAY</releaseArea>
    <releaseApplName>COMM</releaseApplName>
    <libType>CPY</libType>
    <likeType>C</likeType>
    <concatType>C</concatType>
    <libraryFromType>B</libraryFromType>
    <library>CMNTP.S4.V711.BASE.COMM.CPY</library>
    <libraryOrg>PDS</libraryOrg>
   </result>
   <response>
    <statusMessage>SER8209I Logon accepted for user MTULLY; Local CCSID=00037
     </statusMessage>
    <statusReturnCode>00</statusReturnCode>
    <statusReasonCode>8700</statusReasonCode>
   </response>
  </message>
 </scope>
</service>
```

RLSMAREA DETAIL CMP_RLSE

The RLSMAREA DETAIL CMP_RLSE message lists all components in a release concatenation and shows all locations where each component resides.

The XML service/scope/message tags and attributes for this message are:

```
<service name="RLSMAREA">
<scope name="DETAIL">
<message name="CMP_RLSE">
```

These tags appear in both requests and replies.

RLSMAREA DETAIL CMP_RLSE — Request

The following example shows how you might code a request to list the LOD library type information for all components in a release area. Data structure details for the `<request>` tag follow the example.

Example XML — RLSMAREA DETAIL CMP_RLSE Request

```
<?xml version="1.0"?>
<service name="RLSMAREA">
 <scope name="DETAIL">
  <message name="CMP_RLSE">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
    <release>S4711010</release>
    <releaseArea>ACCTPAY </releaseArea>
    <applName>    </applName>
    <libType>LOD</libType>
    <component>                                                    </component>
    <caseMixed> </caseMixed>
   </request>
  </message>
 </scope>
</service>
```

RLSMAREA DETAIL CMP_RLSE <request> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<applName>	Optional	0 - 1	String (4), variable	Release application name.
<caseMixed>	Optional	0 - 1	String (1)	**Y** = Case is mixed. **N** = Case is not mixed.
<component>	Optional	0 - 1	String (256), variable	Component name. *NOTE:* May be masked using asterisk (*) wildcard.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<libType>	Required	1	String (3), variable	Release library type.
<listType>	Optional	1	String (1)	Component list type.
<release>	Required	1	String (8), variable	Release name.
<releaseArea>	Required	1	String (8), variable	Release area name.

RLSMAREA DETAIL CMP_RLSE — Reply

The XML reply to a RLSMAREA DETAIL CMP_RLSE request returns zero to many `<result>` data elements. Each result contains information for a component/library type.

The standard `<response>` data element follows any `<result>` tags in the reply and indicates the success or failure of the request. Successful requests have a return code of 00. Unsuccessful requests have a return code of 04 or higher. Because it is the final data element in the XML reply message, the `<response>` tag serves as an end-of-list marker.

The following example shows what the reply message might look like. Data structure details for the `<result>` tag follow the example.

Example XML — RLSMAREA DETAIL CMP_RLSE Reply

```
<?xml version="1.0"?>
<service name="RLSMAREA">
 <scope name="DETAIL">
  <message name="CMP_RLSE">
   <result>
    <release>BASELINE</release>
    <applName>ACTP</applName>
    <component>ACPSRCCA</component>
    <libType>LOD</libType>
    <likeType>L</likeType>
    <libName>CMNTP.S4.V711.BASE.ACTP.LOD</libName>
   </result>
   <result>
    <release>BASELINE</release>
    <applName>ACTP</applName>
    <component>ACPSRCCC</component>
    <libType>LOD</libType>
    <likeType>L</likeType>
    <libName>CMNTP.S4.V711.BASE.ACTP.LOD</libName>
   </result>
   <result>
    <release>BASELINE</release>
    <applName>ACTP</applName>
    <component>ACPSRCCE</component>
    <libType>LOD</libType>
    <likeType>L</likeType>
    <libName>CMNTP.S4.V711.BASE.ACTP.LOD</libName>
   </result>
   .
   .
```

```
.
<response>
 <statusMessage>CMR9570I - Component list service completed
  </statusMessage>
 <statusReturnCode>00</statusReturnCode>
 <statusReasonCode>9570</statusReasonCode>
</response>
</message>
</scope>
</service>
```

RLSMAREA DETAIL CMP_RLSE <result> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<applName>	Optional	0 - 1	String (4), variable	Release application name.
<component>	Optional	0 - 1	String (256), variable	Component name.
<libName>	Optional	0 - 1	String (44), variable	Library name where component resides.
<libType>	Optional	0 - 1	String (3)	Component library type.
<likeType>	Optional	0 - 1	String (1)	Release application library "like type". Values for all library types: **C** = Copy **K** = LCT (link control cards) **L** = Load **N** = NCAL **O** = Object Additional values for HFS library types: **P** = PDS **S** = Source **X** = List blank = other
<release>	Optional	0 - 1	String (8), variable	Release name.
<releaseArea>	Optional	0 - 1	String (8), variable	Release area name.
<releaseAreaToCheckinDate>	Optional	0 - 1	Date, yyyymmdd	Release area component checkin date.
<releaseAreaToCheckinTime>	Optional	0 - 1	Time, hhmmss	Release area component checkin time.
<sortNumber>	Required	1 - 1	Integer, undefined	Sort number for sorting list.

RLSMAREA DETAIL INTEGRTY

The RLSMAREA DETAIL INTEGRTY message checks the integrity of the component-in-motion (CIM) table against physical members in the release area libraries. All versions of all components in the release concatenation are checked.

The XML service/scope/message tags and attributes for this message are:

```
<service name="RLSMAREA">
<scope name="DETAIL">
<message name="INTEGRTY">
```

These tags appear in both requests and replies.

RLSMAREA DETAIL INTEGRTY — Request

The following example shows how you might code a request to check the integrity of a named release. Data structure details for the `<request>` tag follow the example.

Example XML — RLSMAREA DETAIL INTEGRTY Request

```
<?xml version="1.0"?>
<service name="RLSMAREA">
 <scope name="DETAIL">
  <message name="INTEGRTY">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
    <applName>ACTR    </applName>
    <release>S4712COM</release>
   </request>
  </message>
 </scope>
</service>
```

RLSMAREA DETAIL INTEGRTY <request> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<applName>	Required	1	String (4), variable	Release application name.
<release>	Required	1	String (8), variable	Release name.
<releaseArea>	Optional	0 - 1	String (8), variable	Release area name.

RLSMAREA DETAIL INTEGRTY — Reply

The XML reply to a RLSMAREA DETAIL INTEGRTY request returns zero to many <result> data elements. Each result contains information for a mismatched component.

The standard <response> data element follows any <result> tags in the reply and indicates the success or failure of the request. Successful requests have a return code of 00. Unsuccessful requests have a return code of 04 or higher. Because it is the final data element in the XML reply message, the <response> tag serves as an end-of-list marker.

The following example shows what the reply message might look like. Data structure details for the <result> tag follow the example.

Example XML — RLSMAREA DETAIL INTEGRTY Reply

```
<?xml version="1.0"?>
<service name="RLSMAREA">
 <scope name="DETAIL">
  <message name="INTEGRTY">
   <result>
    <release>S4712COM</release>
    <releaseArea>FINANCE</releaseArea>
    <applName>ACTR</applName>
    <libType>CPY</libType>
    <component>ACRCPYCE</component>
    <isCIMMissing>N</isCIMMissing>
    <isMemberMissing>Y</isMemberMissing>
   </result>
   <result>
    <release>S4712COM</release>
    <releaseArea>ACCTPAY</releaseArea>
    <applName>ACTR</applName>
    <libType>CPY</libType>
    <component>ACRCPYCE</component>
    <isCIMMissing>N</isCIMMissing>
    <isMemberMissing>Y</isMemberMissing>
   </result>
   <result>
    <release>S4712COM</release>
    <releaseArea>FINANCE</releaseArea>
    <applName>ACTR</applName>
    <libType>LCS</libType>
    <component>ACRSCSCE</component>
    <isCIMMissing>N</isCIMMissing>
    <isMemberMissing>Y</isMemberMissing>
   </result>
   .
   .
   .
   <response>
    <statusMessage>CMR9569I - CIM records and library/file contents do not
     match</statusMessage>
    <statusReturnCode>00</statusReturnCode>
    <statusReasonCode>9569</statusReasonCode>
   </response>
  </message>
 </scope>
</service>
```

RLSMAREA DETAIL INTEGRTY <result> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<applName>	Optional	0 - 1	String (4), variable	Release application name.
<component>	Optional	0 - 1	String (256), variable	Mismatched component name.
<isCIMMissing>	Optional	0 - 1	String (1)	**Y** = component is missing from CIM table. **N** = component exists in CIM table.
<isMemberMissing>	Optional	0 - 1	String (1)	**Y** = component is missing from release area library. **N** = component exists in release area library.
<libType>	Optional	0 - 1	String (3)	Component library type.
<release>	Optional	0 - 1	String (8), variable	Release name.
<releaseArea>	Optional	0 - 1	String (8), variable	Release area name.

RLSMAREA DETAIL TEST

The RLSMAREA DETAIL TEST message tests the contents of a release area against all of the packages that may place a component in that area.

The XML service/scope/message tags and attributes for this message are:

```
<service name="RLSMAREA">
<scope name="DETAIL">
<message name="TEST">
```

These tags appear in both requests and replies.

RLSMAREA DETAIL TEST — Request

The following example shows how you might code a request to test the contents of a release area. Data structure details for the `<request>` tag follow the example.

Example XML — RLSMAREA DETAIL TEST Request

```
<?xml version="1.0"?>
<service name="RLSMAREA">
 <scope name="DETAIL">
  <message name="TEST">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
```

```
    <release>S4711010</release>
    <releaseArea>ACCTPAY </releaseArea>
   </request>
  </message>
 </scope>
</service>
```

RLSMAREA DETAIL TEST <request> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<includeExcludePackage>	Optional	1	String (1)	Include or Exclude package (I/X).
<Package>	Optional	1	String (10)	Package name.
<release>	Required	1	String (8), variable	Release name.
<releaseArea>	Required	1	String (8), variable	Release area name.

RLSMAREA DETAIL TEST — Reply

The XML reply to a RLSMAREA DETAIL TEST request returns zero to many `<result>` data elements. Each result contains information for a failing component or package.

The standard `<response>` data element follows any `<result>` tags in the reply and indicates the success or failure of the request. Successful requests have a return code of 00. Unsuccessful requests have a return code of 04 or higher. Because it is the final data element in the XML reply message, the `<response>` tag serves as an end-of-list marker.

The following example shows what the reply message might look like. Data structure details for the `<result>` tag follow the example.

Example XML — RLSMAREA DETAIL TEST Reply

```
<?xml version="1.0"?>
<service name="RLSMAREA">
 <scope name="DETAIL">
  <message name="TEST">
   <result>
    <release>S4711010</release>
    <releaseArea>ACCTPAY</releaseArea>
    <package>ACTP000094</package>
    <packageStatus>DEV</packageStatus>
    <reasonCode>E</reasonCode>
    <reasonForFailure>Empty Package</reasonForFailure>
   </result>
   <result>
    <release>S4711010</release>
    <releaseArea>ACCTPAY</releaseArea>
    <package>ACTP000095</package>
    <packageStatus>DEV</packageStatus>
    <reasonCode>E</reasonCode>
    <reasonForFailure>Empty Package</reasonForFailure>
   </result>
   <response>
```

```
    <statusMessage>CMR1506I - Release S4711010/ACCTPAY and package
     components do not match.</statusMessage>
    <statusReturnCode>00</statusReturnCode>
    <statusReasonCode>1506</statusReasonCode>
   </response>
  </message>
 </scope>
</service>
```

RLSMAREA DETAIL TEST <result> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<componentName>	Optional	0 - 1	String (256), variable	Failing component name.
<componentType>	Optional	0 - 1	String (3)	Failing component type.
<componentUserid>	Optional	0 - 1	String (8), variable	Failing component user ID.
<originatingPackage>	Optional	0 - 1	String (10)	Originating package name.
<originatingUserid>	Optional	0 - 1	String (8), variable	Originating component user ID.
<package>	Optional	0 - 1	String (10)	Failing package name.
<packageStatus>	Optional	0 - 1	String (3)	Package status.
<reasonCode>	Optional	0 - 1	String (1)	Failure reason code.
<reasonForFailure>	Optional	0 - 1	String (20)	Reason for failure.
<recordType>	Optional	0 - 1	String (1)	Component record type.
<release>	Optional	0 - 1	String (8), variable	Release name.
<releaseArea>	Optional	0 - 1	String (8), variable	Release area name.
<sourceComponentName>	Optional	0 - 1	String (256), variable	Failing source component name.
<sourceComponentType>	Optional	0 - 1	String (3)	Failing source component type.

RLSMAREA HST LIST

The RLSMAREA HST LIST message lists the history residing in the ERO component history table. A comprehensive list may be obtained for all components in a release, or a selective list may be obtained by requesting specific fields.

The XML service/scope/message tags and attributes for this message are:

```
<service name="RLSMAREA">
<scope name="HST">
<message name="LIST">
```

These tags appear in both requests and replies.

RLSMAREA HST LIST — Request

The following example shows how you might code a request to list all of the component history for a release. Data structure details for the <request> tag follow the example.

Example XML — RLSMAREA HST LIST Request

```
<?xml version="1.0"?>
<service name="RLSMAREA">
 <scope name="HST">
  <message name="LIST">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
    <release>S4712COM</release>
   </request>
  </message>
 </scope>
</service>
```

RLSMAREA HST LIST <request> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<applName>	Optional	0 - 1	String (4), variable	Release application name.
<buildProc>	Optional	0 - 1	String (8), variable	Component build procedure.
<checkinDescription>	Optional	0 - 1	String (120), variable	Checkin description.
<component>	Optional	0 - 1	String (256), variable	Release area component name.
<componentFromChangeDate>	Optional	0 - 1	Date, yyyymmdd	Component change "from" date.
<componentFromChangeTime>	Optional	0 - 1	Time, hhmmss	Component change "from" time.
<componentHistoryStatus>	Optional	0 - 1	String (3)	Status of action on component. Values: **BAS** = Component was baselined. **CKI** = Component was checked in. **CKO** = Component was checked out. **DEL** = Component was deleted. **DEM** = Component was demoted. **MDL** = Component was memo-deleted. **PRM** = Component was promoted. **RTV** = Component was retrieved. **STG** = Component was staged.
<componentToChangeDate>	Optional	0 - 1	Date, yyyymmdd	Component change "to" date.
<componentToChangeTime>	Optional	0 - 1	Time, hhmmss	Component change "to" time.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<componentType>	Optional	0 - 1	String (3)	Component library type.
<language>	Optional	0 - 1	String (8), variable	Component source code language.
<lastHistory>	Optional	0 - 1	String (1)	**Y** = List last history record only. **N** = List all history records.
<package>	Optional	0 - 1	String (10), variable	Package name. *NOTE:* May be masked using asterisk (*) wildcard.
<packageId>	Optional	0 - 1	Integer (6)	Package number.
<release>	Required	1	String (8), variable	Release name.
<releaseArea>	Optional	0 - 1	String (8), variable	Release area name.
<releaseAreaFromCheckinDate>	Optional	0 - 1	Date, yyyymmdd	Release area checkin "from" date.
<releaseAreaFromRetrieveDate>	Optional	0 - 1	Date, yyyymmdd	Release area retrieve "from" date.
<releaseAreaToCheckinDate>	Optional	0 - 1	Date, yyyymmdd	Release area checkin "to" date.
<releaseAreaToRetrieveDate>	Optional	0 - 1	Date, yyyymmdd	Release area retrieve "to" date.
<updater>	Optional	0 - 1	String (8), variable	Component checkin user ID.

RLSMAREA HST LIST — Reply

The XML reply to a RLSMAREA HST LIST request returns zero to many `<result>` data elements. Each result lists a record of history data for a component.

The standard `<response>` data element follows any `<result>` tags in the reply and indicates the success or failure of the request. Successful requests have a return code of 00. Unsuccessful requests have a return code of 04 or higher. Because it is the final data element in the XML reply message, the `<response>` tag serves as an end-of-list marker.

The following example shows what the reply message might look like. Data structure details for the `<result>` tag follow the example.

Example XML — RLSMAREA HST LIST Reply

```
<?xml version="1.0"?>
<service name="RLSMAREA">
 <scope name="HST">
  <message name="LIST">
   <result>
    <release>S4712COM</release>
    <releaseArea>ACCTPAY</releaseArea>
    <package>ACTP000076</package>
    <applName>ACTP</applName>
    <packageId>000076</packageId>
```

```
<componentHistoryType>RLS</componentHistoryType>
<componentType>CPY</componentType>
<component>ACPCPYCE</component>
<updater>KCAMPBE</updater>
<componentChangeDate>20120827</componentChangeDate>
<componentChangeTime>135951</componentChangeTime>
<componentPriorChangeDate>20120827</componentPriorChangeDate>
<componentPriorChangeTime>135835</componentPriorChangeTime>
<setssi>630B28C0</setssi>
<version>01</version>
<modLevel>02</modLevel>
<componentHistoryStatus>CKI</componentHistoryStatus>
<componentPriorHistoryStatus>RTV</componentPriorHistoryStatus>
<useDb2PreCompileOption>N</useDb2PreCompileOption>
<forceAssignedBuildProc>N</forceAssignedBuildProc>
<size>00000003</size>
<checkinReleaseArea>PACKAGE</checkinReleaseArea>
<checkinUser>KCAMPBE</checkinUser>
<releaseAreaCheckinDate>20120827</releaseAreaCheckinDate>
<releaseAreaCheckinTime>135951</releaseAreaCheckinTime>
<retrieveReleaseArea>ACCTPAY</retrieveReleaseArea>
<retrieveUser>KCAMPBE</retrieveUser>
<releaseAreaRetrieveDate>20120827</releaseAreaRetrieveDate>
<releaseAreaRetrieveTime>135835</releaseAreaRetrieveTime>
<promotionDate>20121231</promotionDate>
<promotionTime>000000</promotionTime>
<componentCheckoutSetssi>630B28C0</componentCheckoutSetssi>
<componentCheckoutDate>20120827</componentCheckoutDate>
<componentCheckoutTime>135951</componentCheckoutTime>
<userOption7205>                                                CMNTP   HRAPL
 &#xFF00;&#xFF00;&#xFF00;&#xFF00;/cmntp</userOption7205>
</result>
.
.
.
<response>
 <statusMessage>CMR8700I - Release HST Table service completed
  </statusMessage>
 <statusReturnCode>00</statusReturnCode>
 <statusReasonCode>8700</statusReasonCode>
</response>
</message>
</scope>
</service>
```

RLSMAREA HST LIST <result> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<applName>	Optional	0 - 1	String (4), variable	Release application name.
<buildProc>	Optional	0 - 1	String (8), variable	Component build procedure.
<builder>	Optional	0 - 1	String (8), variable	Component build user ID.
<checkinDescription>	Optional	0 - 1	String (120), variable	Component checkin description.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<checkinReleaseArea>	Optional	0 - 1	String (8), variable	Component checkin release area name.
<checkinUser>	Optional	0 - 1	String (8), variable	Component checkin user ID.
<checkoutAssocPkg>	Optional	0 - 1	String (10)	Component checkout associated package name.
<checkoutHashToken>	Optional	0 - 1	String (16)	Component checkout hash token.
<checkoutRelease>	Optional	0 - 1	String (8), variable	Component checkout release name.
<checkoutReleaseArea>	Optional	0 - 1	String (8), variable	Component checkout release area name.
<checkoutUser>	Optional	0 - 1	String (8), variable	Component checkout user ID.
<compileOptions>	Optional	0 - 1	String (34), variable	Compile options.
<component>	Optional	0 - 1	String (256), variable	Component name.
<componentBuildNumber>	Optional	0 - 1	String (10)	Component build number.
<componentChangeDate>	Optional	0 - 1	Date, yyyymmdd	Component changed date.
<componentChangeTime>	Optional	0 - 1	Time, hhmmss	Component changed time.
<componentCheckoutDate>	Optional	0 - 1	Date, yyyymmdd	Component checkout date.
<componentCheckoutSetssi>	Optional	0 - 1	String (8)	Component checkout SETSSI.
<componentCheckoutTime>	Optional	0 - 1	Time, hhmmss	Component checkout time.
<componentHistoryStatus>	Optional	0 - 1	String (3)	Status of action on component. Values: **BAS** = Component was baselined. **CKI** = Component was checked in. **CKO** = Component was checked out. **DEL** = Component was deleted. **DEM** = Component was demoted. **MDL** = Component was memo-deleted. **PRM** = Component was promoted. **RTV** = Component was retrieved. **STG** = Component was staged.
<componentHistoryType>	Optional	0 - 1	String (3)	Type of component history record. Values: **GEN** = Component general record. **RLS** = Component release record.
<componentLastBaselineDate>	Optional	0 - 1	Date, yyyymmdd	Component last baseline date.
<componentLastBaselineTime>	Optional	0 - 1	Time, hhmmss	Component last baseline time.
<componentLastBuildDate>	Optional	0 - 1	Date, yyyymmdd	Component last build date.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<componentLastBuildTime>	Optional	0 - 1	Time, hhmmss	Component last build time.
<componentLastCheckoutDate>	Optional	0 - 1	Date, yyyymmdd	Component last checkout date.
<componentLastCheckoutTime>	Optional	0 - 1	Time, hhmmss	Component last checkout time.
<componentPriorChangeDate>	Optional	0 - 1	Date, yyyymmdd	Component prior change date.
<componentPriorChangeTime>	Optional	0 - 1	Time, hhmmss	Component prior change time.
<componentPriorHistoryStatus>	Optional	0 - 1	String (3)	Status of prior action on component. Values: **BAS** = Component was baselined. **CKI** = Component was checked in. **CKO** = Component was checked out. **DEL** = Component was deleted. **DEM** = Component was demoted. **MDL** = Component was memo-deleted. **PRM** = Component was promoted. **RTV** = Component was retrieved. **STG** = Component was staged.
<componentType>	Optional	0 - 1	String (3)	Component library type.
<forceAssignedBuildProc>	Optional	0 - 1	String (1)	Compile "force" option. Values: **1** = Allow users to compile this component with alternate procedures prior to freezing the package. The last compile prior to freezing the package must be done with the designated procedure. **2** = Force users to use the designated procedure for all compiles.
<language>	Optional	0 - 1	String (8), variable	Name of source code language for component.
<linkOptions>	Optional	0 - 1	String (34), variable	Link options.
<loadModuleSize>	Optional	0 - 1	String (8)	Load module size.
<modLevel>	Optional	0 - 1	String (2)	Component modification level.
<package>	Optional	0 - 1	String (10)	Release package name.
<packageId>	Optional	0 - 1	Integer (6)	Release package number.
<promoter>	Optional	0 - 1	String (8), variable	Promoter TSO ID/job name.
<promotionDate>	Optional	0 - 1	Date, yyyymmdd	Component promotion date.
<promotionLevel>	Optional	0 - 1	String (2)	Promotion level number.
<promotionName>	Optional	0 - 1	String (8), variable	Promotion nick name.
<promotionSite>	Optional	0 - 1	String (8), variable	Promotion site name.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<promotionTime>	Optional	0 - 1	Time, hhmmss	Component promotion time.
<release>	Optiona	0 - 1	String (8), variable	Release name.
<releaseArea>	Optiona	0 - 1	String (8), variable	Release area name.
<releaseAreaCheckinDate>	Optional	0 - 1	Date, yyyymmdd	Release area checkin date.
<releaseAreaCheckinTime>	Optional	0 - 1	Time, hhmmss	Release area checkin time.
<releaseAreaRetrieveDate>	Optional	0 - 1	Date, yyyymmdd	Release area retrieve date.
<releaseAreaRetrieveTime>	Optional	0 - 1	Time, hhmmss	Release area retrieve time.
<requestorDept>	Optiona	0 - 1	String (4), variable	Requestor department.
<retrievePackage>	Optiona	0 - 1	String (10)	Release retrieve package name.
<retrieveReleaseArea>	Optiona	0 - 1	String (8), variable	Component retrieve release area name.
<retrieveUser>	Optiona	0 - 1	String (8), variable	Component retrieve user ID.
<setssi>	Optional	0 - 1	String (8)	Component SETSSI date.
<size>	Optional	0 - 1	String (8)	Size of component in lines of code.
<targetLoadLibType>	Optional	0 - 1	String (3)	Target load library type.
<updater>	Optional	0 - 1	String (8), variable	User ID of user who last updated component.
<useDb2PreCompileOption>	Optional	0 - 1	String (1)	**Y** = Use Db2 pre-processing step. **N** = Do not use Db2 pre-processing step.
<userOption01> . . . <userOption20>	Optional	0 - 1	String (1)	Set of up to 20 one-byte, custom, administrator-defined variables. Values: **Y** = Yes **N** = No
<userOption0101> . . . <userOption0105>	Optional	0 - 1 each	String (1)	Administrator-defined build options assigned to component. Each tag corresponds to User Option 0101 to 0105 on the ISPF user options panel for component build.
<userOption0201> . . . <userOption0203>	Optional	0 - 1 each	String (2), variable	Administrator-defined build options assigned to component. Each tag corresponds to User Option 0201 to 0203 on the ISPF user options panel for component build.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<userOption0301> . . . <userOption0303>	Optional	0 - 1 each	String (3), variable	Administrator-defined build options assigned to component. Each tag corresponds to User Option 0301 to 0303 on the ISPF user options panel for component build.
<userOption0401> . . . <userOption0403>	Optional	0 - 1 each	String (4), variable	Administrator-defined build options assigned to component. Each tag corresponds to User Option 0401 to 0403 on the ISPF user options panel for component build.
<userOption0801> . . . <userOption0805>	Optional	0 - 1 each	String (8), variable	Administrator-defined build options assigned to component. Each tag corresponds to User Option 0801 to 0805 on the ISPF user options panel for component build.
<userOption1001> . . . <userOption1002>	Optional	0 - 1 each	String (10), variable	Administrator-defined build options assigned to component. Each tag corresponds to User Option 1001 to 1002 on the ISPF user options panel for component build.
<userOption1601> . . . <userOption1602>	Optional	0 - 1 each	String (16), variable	Administrator-defined build options assigned to component. Each tag corresponds to User Option 1601 to 1603 on the ISPF user options panel for component build.
<userOption3401> . . . <userOption3402>	Optional	0 - 1 each	String (34), variable	Administrator-defined build options assigned to component. Each tag corresponds to User Option 3401 to 3402 on the ISPF user options panel for component build.
<userOption4401> . . . <userOption4402>	Optional	0 - 1 each	String (44), variable	Administrator-defined build options assigned to component. Each tag corresponds to User Option 4401 to 4402 on the ISPF user options panel for component build.
<userOption6401> . . . <userOption6405>	Optional	0 - 1 each	String (64), variable	Administrator-defined build options assigned to component. Each tag corresponds to User Option 6401 to 6405 on the ISPF user options panel for component build.
<userOption7201> . . . <userOption7205>	Optional	0 - 1 each	String (72), variable	Administrator-defined build options assigned to component. Each tag corresponds to User Option 7201 to 7205 on the ISPF user options panel for component build.
<version>	Optional	0 - 1	String (2)	Component version number.

RLSMAREA IMP LIST

The RLSMAREA IMP LIST message lists the impact data stored in the ERO Db2 impact table.

The XML service/scope/message tags and attributes for this message are:

```
<service name="RLSMAREA">
<scope name="IMP">
<message name="LIST">
```

These tags appear in both requests and replies.

RLSMAREA IMP LIST — Request

The only required subtag for this request is <sourceRelease>, which will list all of the impact data for a release. The remaining subtags allow you to request specific component relationships.

The 'source' tags refer to the top level of the relationship and the 'target' tags refer to the bottom level. For example, in a source-to-copybook relationship, the 'source' tags pertain to where the source component was generated; the 'target' fields pertain to where the copybook was pulled from. Similarly, for a composite load-to-subroutine relationship, the 'source' is for the composite and the 'target' is for the subroutine. The <toWhat> tag value is the name of the subcomponent (bottom level component, i.e. copybook or subroutine).

The following example shows how you might code a request to list all of the impact data for a release. Data structure details for the <request> tag follow the example.

Example XML — RLSMAREA IMP LIST Request

```
<?xml version="1.0"?>
<service name="RLSMAREA">
 <scope name="IMP">
  <message name="LIST">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
    <sourceRelease>S4712COM</sourceRelease>
   </request>
  </message>
 </scope>
</service>
```

RLSMAREA IMP LIST <request> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<relation>	Optional	0 - 1	String (3)	Impact relationship. Values: **CPY** = Subordinate copybooks. **LOD** = Subordinate statically linked subroutines.
<sourceApplName>	Optional	0 - 1	String (4), variable	Source release application name.
<sourceComponent>	Optional	0 - 1	String (256), variable	Source release area component name.
<sourceLibraryType>	Optional	0 - 1	String (3)	Source release component library type.
<sourceRelease>	Required	1	String (8), variable	Source release name.
<sourceReleaseArea>	Optional	0 - 1	String (8), variable	Source release area name.
<targetApplName>	Optional	0 - 1	String (4), variable	Target release application name.
<targetLibraryType>	Optional	0 - 1	String (3)	Target release component library type.
<targetRelease>	Optional	0 - 1	String (8), variable	Target release name.
<targetReleaseArea>	Optional	0 - 1	String (8), variable	Target release area name.
<toWhat>	Optional	0 - 1	String (256), variable	Target subcomponent of relationship.

RLSMAREA IMP LIST — Reply

The XML reply to a RLSMAREA IMP LIST request returns zero to many `<result>` data elements. Each result lists impact data for a component relationship.

The standard `<response>` data element follows any `<result>` tags in the reply and indicates the success or failure of the request. Successful requests have a return code of 00. Unsuccessful requests have a return code of 04 or higher. Because it is the final data element in the XML reply message, the `<response>` tag serves as an end-of-list marker.

The following example shows what the reply message might look like. Data structure details for the `<result>` tag follow the example.

Example XML — RLSMAREA IMP LIST Reply

```
<?xml version="1.0"?>
<service name="RLSMAREA">
 <scope name="IMP">
  <message name="LIST">
   <result>
    <sourceRelease>S4712COM</sourceRelease>
    <sourceReleaseArea>ACCTPAY</sourceReleaseArea>
    <sourceApplName>COMM</sourceApplName>
    <sourceLibraryType>SRS</sourceLibraryType>
```

```
 <sourceLibraryBun>0000000032</sourceLibraryBun>
 <sourceComponent>COMSRS10</sourceComponent>
 <relation>CPY</relation>
 <targetApplName>COMM</targetApplName>
 <targetLibraryType>CPY</targetLibraryType>
 <targetLibraryBun>0000000028</targetLibraryBun>
 <toWhat>COMCPY10</toWhat>
 <hashToken>8C2DCBB800000080</hashToken>
</result>
<result>
 <sourceRelease>S4712COM</sourceRelease>
 <sourceReleaseArea>ACCTPAY</sourceReleaseArea>
 <sourceApplName>ACTR</sourceApplName>
 <sourceLibraryType>SCS</sourceLibraryType>
 <sourceLibraryBun>0000000024</sourceLibraryBun>
 <sourceComponent>ACRSCSCE</sourceComponent>
 <relation>CPY</relation>
 <targetRelease>S4712COM</targetRelease>
 <targetReleaseArea>ACCTPAY</targetReleaseArea>
 <targetApplName>ACTR</targetApplName>
 <targetLibraryType>CPY</targetLibraryType>
 <targetLibraryBun>0000000017</targetLibraryBun>
 <toWhat>ACRCPYCE</toWhat>
 <hashToken>8FA76F0000000079</hashToken>
</result>
.
.
.
<response>
 <statusMessage>CMR8700I - Release IMP Table service completed
  </statusMessage>
 <statusReturnCode>00</statusReturnCode>
 <statusReasonCode>8700</statusReasonCode>
</response>
</message>
</scope>
</service>
```

RLSMAREA IMP LIST <result> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<hashToken>	Optional	0 - 1	String (16)	Component hash token for CPY relation.
<package>	Optional	0 - 1	String (10)	Package name for LOD relation.
<relation>	Optional	0 - 1	String (3)	Impact relationship. Values: **CPY** = Subordinate copybooks. **LOD** = Subordinate statically linked subroutines.
<setssi>	Optional	0 - 1	String (8)	Component SETSSI for LOD relation.
<sourceApplName>	Optional	0 - 1	String (4), variable	Source release application name.
<sourceComponent>	Optional	0 - 1	String (256), variable	Source release area component name.
<sourceLibraryBun>	Optional	0 - 1	String (10)	Source library type BUN number.
<sourceLibraryType>	Optional	0 - 1	String (3)	Source release component library type.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<sourceRelease>	Required	0 - 1	String (8), variable	Source release name.
<sourceReleaseArea>	Optional	0 - 1	String (8), variable	Source release area name.
<targetApplName>	Optional	0 - 1	String (4), variable	Target release application name.
<targetLibraryBun>	Optional	0 - 1	String (10)	Target library type BUN number.
<targetLibraryType>	Optional	0 - 1	String (3)	Target release component library type.
<targetRelease>	Required	0 - 1	String (8), variable	Target release name.
<targetReleaseArea>	Optional	0 - 1	String (8), variable	Target release area name.
<toWhat>	Optional	0 - 1	String (256), variable	Target subcomponent of relationship.

RLSMAREA LOAD SYSLIB

The RLSMAREA LOAD SYSLIB message lists the LOADLIB concatenation for a release application.

The XML service/scope/message tags and attributes for this message are:

```
<service name="RLSMAREA">
<scope name="LOAD">
<message name="SYSLIB">
```

These tags appear in both requests and replies.

RLSMAREA LOAD SYSLIB — Request

The following example shows how you might code a request to list the LOADLIB concatenation for a release application. Data structure details for the `<request>` tag are identical to those for the RLSMAREA ALL_CHK SYSLIB message - see "RLSMAREA ALL_CHK SYSLIB <request> Data Structure" on page 42.

Example XML — RLSMAREA LOAD SYSLIB Request

```
<?xml version="1.0"?>
<service name="RLSMAREA">
 <scope name="LOAD">
  <message name="SYSLIB">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
    <release>S4711010</release>
```

```
    <releaseArea>ACCTPAY </releaseArea>
    <releaseApplName>ACTP</releaseApplName>
    <language>COBOL2  </language>
    <buildProc>CMNCOB2 </buildProc>
    <libType>LOD</libType>
    <package>          </package>
   </request>
  </message>
 </scope>
</service>
```

RLSMAREA LOAD SYSLIB — Reply

The XML reply to a RLSMAREA LOAD SYSLIB request returns zero to many `<result>` data elements. Each result contains LOADLIB information data for a release application.

The standard `<response>` data element follows any `<result>` tags in the reply and indicates the success or failure of the request. Successful requests have a return code of 00. Unsuccessful requests have a return code of 04 or higher. Because it is the final data element in the XML reply message, the `<response>` tag serves as an end-of-list marker.

The following example shows what the reply message might look like. Data structure details for the `<result>` tag are identical to those for the RLSMAREA ALL_CHK SYSLIB message - see "RLSMAREA ALL_CHK SYSLIB <result> Data Structure" on page 44.

Example XML — RLSMAREA LOAD SYSLIB Reply

```
<?xml version="1.0"?>
<service name="RLSMAREA">
 <scope name="LOAD">
  <message name="SYSLIB">
   <result>
    <release>S4711010</release>
    <releaseArea>ACCTPAY</releaseArea>
    <releaseApplName>ACTP</releaseApplName>
    <libType>LOS</libType>
    <likeType>N</likeType>
    <concatType>L</concatType>
    <libraryFromType>B</libraryFromType>
    <library>CMNTP.S4.V711.BASE.ACTP.LOS</library>
    <libraryOrg>PDS</libraryOrg>
   </result>
   <result>
    <release>S4711010</release>
    <releaseArea>ACCTPAY</releaseArea>
    <releaseApplName>COMM</releaseApplName>
    <libType>LOS</libType>
    <likeType>N</likeType>
    <concatType>L</concatType>
    <libraryFromType>B</libraryFromType>
    <library>CMNTP.S4.V711.BASE.COMM.LOS</library>
    <libraryOrg>PDS</libraryOrg>
   </result>
   <response>
    <statusMessage>SER8209I Logon accepted for user MTULLY; Local CCSID=00037
     </statusMessage>
    <statusReturnCode>00</statusReturnCode>
    <statusReasonCode>8700</statusReasonCode>
```

```
   </response>
  </message>
 </scope>
</service>
```

RLSMAREA SCAN CMP_RLSE

The RLSMAREA SCAN CMP_RLSE message scans the latest version of components in a release concatenation to find those with contents matching a search string. Release area libraries are searched for the current and prior releases, as well as baseline libraries.

The XML service/scope/message tags and attributes for this message are:

```
<service name="RLSMAREA">
<scope name="SCAN">
<message name="CMP_RLSE">
```

These tags appear in both requests and replies.

RLSMAREA SCAN CMP_RLSE — Request

The following example shows how you might code a request to scan the latest version of components of library type "SRC" that contain the string "SAMCPY1B". Data structure details for the `<request>` tag follow the example.

Example XML — RLSMAREA SCAN CMP_RLSE Request

```
<?xml version="1.0"?>
<service name="RLSMAREA">
 <scope name="SCAN">
  <message name="CMP_RLSE">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
    <release>S4711010</release>
    <releaseArea>ACCTPAY </releaseArea>
    <libType>SRC</libType>
    <scan1>SAMCPY1B                                    </scan1>
   </request>
  </message>
 </scope>
</service>
```

RLSMAREA SCAN CMP_RLSE <request> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<and>	Optional	0 - 1	String (1)	**Y** = Concatenate <scan1> and <scan2> search strings. **N** = Do not concatenate search strings.
<applName>	Optional	0 - 1	String (4), variable	Release application name.
<caseMixed>	Optional	0 - 1	String (1)	**Y** = Search strings, <scan1> and <scan2>, are mixed case. **N** = Search strings are not mixed case.
<caseSensitive>	Optional	0 - 1	String (1)	**Y** = Search strings, <scan1> and <scan2>, are case sensitive. **N** = Search strings are not case sensitive.
<component>	Optional	0 - 1	String (256), variable	Component name to search for.
<libType>	Required	1	String (3)	Component library type.
<listType>	Optional	1	String (1)	Component list type.
<release>	Required	1	String (8), variable	Release name.
<releaseArea>	Required	1	String (8), variable	Release area name.
<scan1>	Optional	0 - 1	String (40), variable	Scan string 1.
<scan2>	Optional	0 - 1	String (40), variable	Scan string 2.

RLSMAREA SCAN CMP_RLSE — Reply

The XML reply to a RLSMAREA SCAN CMP_RLSE request returns zero to many `<result>` data elements. Each result lists information for a component that matched the search string.

The standard `<response>` data element follows any `<result>` tags in the reply and indicates the success or failure of the request. Successful requests have a return code of 00. Unsuccessful requests have a return code of 04 or higher. Because it is the final data element in the XML reply message, the `<response>` tag serves as an end-of-list marker.

The following example shows what the reply message might look like. Data structure details for the `<result>` tag follow the example.

Example XML — RLSMAREA SCAN CMP_RLSE Reply

```
<?xml version="1.0"?>
<service name="RLSMAREA">
 <scope name="SCAN">
  <message name="CMP_RLSE">
   <result>
    <release>BASELINE</release>
    <applName>ACTP</applName>
```

```
    <component>SAMSRC1A</component>
    <libType>SRC</libType>
    <package>ACTP</package>
    <matchLine>           COPY SAMCPY1B.</matchLine>
    <libName>CMNTP.S4.V711.BASE.ACTP.SRC</libName>
   </result>
   <result>
    <release>BASELINE</release>
    <applName>ACTP</applName>
    <component>SAMSRC1A</component>
    <libType>SRC</libType>
    <package>ACTP</package>
    <matchLine>           CALL &apos;SAMSRS1B&apos; USING SAMCPY1B.
     </matchLine>
    <libName>CMNTP.S4.V711.BASE.ACTP.SRC</libName>
   </result>
   <response>
    <statusMessage>CMR9572I - Component scan service completed
     </statusMessage>
    <statusReturnCode>00</statusReturnCode>
    <statusReasonCode>9572</statusReasonCode>
   </response>
  </message>
 </scope>
</service>
```

RLSMAREA SCAN CMP_RLSE <result> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<applName>	Optional	0 - 1	String (4), variable	Release application name.
<component>	Optional	0 - 1	String (256), variable	Component name.
<libName>	Required	0 - 1	String (44), variable	Name of library where component resides.
<libType>	Required	0 - 1	String (3)	Component library type.
<lineNumber>	Required	0 - 1	Integer, Undefined	Line number the text is on.
<matchLine>	Optional	0 - 1	String (256), variable	Matching line in component.
<release>	Required	1 - 1	String (8), variable	Release name.
<releaseArea>	Required	1 - 1	String (8), variable	Release area name.
<sortNumber>	Required	1 - 1	Integer, undefined	Sort number for sorting list.

RLSMAREA SCANALL CMP_RLSE

The RLSMAREA SCANALL CMP_RLSE message scans all versions of components in a release concatenation to find those with contents matching a search string. Release area libraries are searched for the current and prior releases, as well as baseline libraries.

The XML service/scope/message tags and attributes for this message are:

```
<service name="RLSMAREA">
<scope name="SCANALL">
<message name="CMP_RLSE">
```

These tags appear in both requests and replies.

RLSMAREA SCANALL CMP_RLSE — Request

The following example shows how you might code a request to scan all versions of components of library type "SRC" that contain the string "SAMCPY1B". Data structure details for the <request> tag are identical to those for the RLSMAREA SCAN CMP_RLSE message - see "RLSMAREA SCAN CMP_RLSE <request> Data Structure" on page 78.

Example XML — RLSMAREA SCANALL CMP_RLSE Request

```
<?xml version="1.0"?>
<service name="RLSMAREA">
 <scope name="SCANALL">
  <message name="CMP_RLSE">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
    <release>S4711010</release>
    <releaseArea>ACCTPAY </releaseArea>
    <libType>SRC</libType>
    <scan1>SAMCPY1B                                    </scan1>
   </request>
  </message>
 </scope>
</service>
```

RLSMAREA SCANALL CMP_RLSE — Reply

The XML reply to a RLSMAREA SCANALL CMP_RLSE request returns zero to many <result> data elements. Each result lists information for a component that matched the search string.

The standard <response> data element follows any <result> tags in the reply and indicates the success or failure of the request. Successful requests have a return code of 00. Unsuccessful requests have a return code of 04 or higher. Because it is the final data element in the XML reply message, the <response> tag serves as an end-of-list marker.

The following example shows what the reply message might look like. Data structure details for the <result> tag are identical to those for the RLSMAREA SCAN CMP_RLSE message - see "RLSMAREA SCAN CMP_RLSE <result> Data Structure" on page 79.

Example XML — RLSMAREA SCANALL CMP_RLSE Reply

```
<?xml version="1.0"?>
<service name="RLSMAREA">
 <scope name="SCANALL">
  <message name="CMP_RLSE">
   <result>
    <release>BASELINE</release>
    <applName>ACTP</applName>
    <component>SAMSRC1A</component>
    <libType>SRC</libType>
    <matchLine>           COPY SAMCPY1B.</matchLine>
    <libName>CMNTP.S4.V711.BASE.ACTP.SRC</libName>
   </result>
   <result>
    <release>BASELINE</release>
    <applName>ACTP</applName>
    <component>SAMSRC1A</component>
    <libType>SRC</libType>
    <matchLine>           CALL &amp;apos;SAMSRS1B&amp;apos; USING SAMCPY1B.
     </matchLine>
    <libName>CMNTP.S4.V711.BASE.ACTP.SRC</libName>
   </result>
   <response>
    <statusMessage>CMR9572I - Component scan service completed
     </statusMessage>
    <statusReturnCode>00</statusReturnCode>
    <statusReasonCode>9572</statusReasonCode>
   </response>
  </message>
 </scope>
</service>
```

RLSMAREA SERVICE LIST

The RLSMAREA SERVICE LIST message lists release area definitions.

The XML service/scope/message tags and attributes for this message are:

```
<service name="RLSMAREA">
<scope name="SERVICE">
<message name="LIST">
```

These tags appear in both requests and replies.

RLSMAREA SERVICE LIST — Request

The following example shows how you might code a request to list the release area definitions for all areas in a named release. Data structure details for the <request> tag follow the example.

Example XML — RLSMAREA SERVICE LIST Request

```
<?xml version="1.0"?>
<service name="RLSMAREA">
 <scope name="SERVICE">
  <message name="LIST">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
    <release>S4711010</release>
   </request>
  </message>
 </scope>
</service>
```

RLSMAREA SERVICE LIST <request> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<release>	Required	1	String (8), variable	Release name.
<releaseArea>	Optional	0 - 1	String (8), variable	Release area name.
<releaseParms>	Optional	0 - 1	String (1), variable	Release parms S or blank.

RLSMAREA SERVICE LIST — Reply

The XML reply to a RLSMAREA SERVICE LIST request returns zero to many `<result>` data elements. Each `<result>` contains information for a release area.

The standard `<response>` data element follows any `<result>` tags in the reply and indicates the success or failure of the request. Successful requests have a return code of 00. Unsuccessful requests have a return code of 04 or higher. Because it is the final data element in the XML reply message, the `<response>` tag serves as an end-of-list marker.

The following example shows what the reply message might look like. Data structure details for the `<result>` tag follow the example.

Example XML — RLSMAREA SERVICE LIST Reply

```
<?xml version="1.0"?>
<service name="RLSMAREA">
 <scope name="SERVICE">
  <message name="LIST">
   <result>
    <release>S4711010</release>
    <releaseArea>ACCTPAY</releaseArea>
    <releaseAreaDesc>Starting area for Accounts Payable components
     </releaseAreaDesc>
    <releaseAreaType>0</releaseAreaType>
    <releaseAreaStepNumber>0001</releaseAreaStepNumber>
```

```
<releaseNextReleaseArea>FINANCE</releaseNextReleaseArea>
<isReleaseAreaBlocked>N</isReleaseAreaBlocked>
<isReleaseAreaApprovedForCheckin>Y</isReleaseAreaApprovedForCheckin>
<isReleaseAreaApprovedForCheckoff>N</isReleaseAreaApprovedForCheckoff>
<isReleaseAreaRejectedForCheckin>N</isReleaseAreaRejectedForCheckin>
<isReleaseAreaRejectedForCheckoff>N</isReleaseAreaRejectedForCheckoff>
<releaseAreaAuditRule>0</releaseAreaAuditRule>
<releaseAreaBlockingRule>0</releaseAreaBlockingRule>
<releaseAreaCheckinRule>0</releaseAreaCheckinRule>
<releaseAreaRetrieveRule>0</releaseAreaRetrieveRule>
<releaseAreaApproveRule>0</releaseAreaApproveRule>
<isReleaseAreaAuditPending>N</isReleaseAreaAuditPending>
<isReleaseAreaCheckinPending>N</isReleaseAreaCheckinPending>
<isReleaseAreaApprovalPending>N</isReleaseAreaApprovalPending>
<isReleaseAreaBlockPending>N</isReleaseAreaBlockPending>
<addRelatedApprovers>Y</addRelatedApprovers>
<allowComponentCheckout>Y</allowComponentCheckout>
<excludeAreaFromConcatenation>N</excludeAreaFromConcatenation>
<overrideOverlaidComponent>N</overrideOverlaidComponent>
<bypassAreaTestDevPkg>N</bypassAreaTestDevPkg>
<bypassAreaTestFrzPkg>N</bypassAreaTestFrzPkg>
<bypassAreaTestAprPkg>N</bypassAreaTestAprPkg>
<bypassAreaTestEmptyPkg>N</bypassAreaTestEmptyPkg>
<bypassAreaTestPkgIntg>N</bypassAreaTestPkgIntg>
</result>
.
.
.
<response>
<statusMessage>CMR8700I - Area information service completed
</statusMessage>
<statusReturnCode>00</statusReturnCode>
<statusReasonCode>8700</statusReasonCode>
</response>
</message>
</scope>
</service>
```

RLSMAREA SERVICE LIST <result> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<addRelatedApprovers>	Optional	0 - 1	String (1)	**Y** = Add related release area approvers. **N** = Do not add related release area approvers.
<allowComponentCheckout>	Optional	0 - 1	String (1)	**Y** = Allow component checkout from release area. **N** = Do not allow component checkout from release area.
<bypassAreaTestAprPkg>	Optional	0 - 1	String (1)	**Y** = Bypass test area packages in APR status. **N** = Do not bypass test area packages in APR status.
<bypassAreaTestDevPkg>	Optional	0 - 1	String (1)	**Y** = Bypass test area packages in DEV status. **N** = Do not bypass test area packages in DEV status.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<bypassAreaTestEmptyPkg>	Optional	0 - 1	String (1)	**Y** = Bypass test area packages that are empty. **N** = Do not bypass test area packages that are empty.
<bypassAreaTestFrzPkg>	Optional	0 - 1	String (1)	**Y** = Bypass test area packages in FRZ status. **N** = Do not bypass test area packages in FRZ status.
<bypassAreaTestPkgIntg>	Optional	0 - 1	String (1)	**Y** = Bypass test area package integrity. **N** = Do not bypass test area package integrity.
<excludeAreaFromConcatenation>	Optional	0 - 1	String (1)	**Y** = Exclude release area from SYSLIB concatenation. **N** = Do not exclude release area from SYSLIB concatenation.
<isReleaseAreaApprovalPending>	Optional	0 - 1	String (1)	**Y** = Release area approval is pending. **N** = Release area approval is not pending.
<isReleaseAreaApprovedForCheckin>	Optional	0 - 1	String (1)	**Y** = Release area is approved for checkin. **N** = Release area is not approved for checkin.
<isReleaseAreaApprovedForCheckoff>	Optional	0 - 1	String (1)	**Y** = Release area is approved for checkoff. **N** = Release area is not approved for checkoff.
<isReleaseAreaAuditPending>	Optional	0 - 1	String (1)	**Y** = Release area audit is pending. **N** = Release area audit is not pending.
<isReleaseAreaBlockPending>	Optional	0 - 1	String (1)	**Y** = Release area block is pending. **N** = Release area block is not pending.
<isReleaseAreaBlocked>	Optional	0 - 1	String (1)	**Y** = Release area is blocked. **N** = Release area is not blocked.
<isReleaseAreaCheckinPending>	Optional	0 - 1	String (1)	**Y** = Release area checkin is pending. **N** = Release area checkin is not pending.
<isReleaseAreaRejectedForCheckin>	Optional	0 - 1	String (1)	**Y** = Release area is rejected for checkin. **N** = Release area is not rejected for checkin.
<isReleaseAreaRejectedForCheckoff>	Optional	0 - 1	String (1)	**Y** = Release area is rejected for checkoff. **N** = Release area is not rejected for checkoff.
<overrideOverlaidComponent>	Optional	0 - 1	String (1)	**Y** = Override overlaid components. **N** = Do not override overlaid components.
<release>	Optional	0 - 1	String (8), variable	Release name.
<releaseArea>	Optional	0 - 1	String (8), variable	Release area name.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<releaseAreaApproveRule>	Optional	0 - 1	Integer (1)	Release area approval rule. Values: **0** = No approvals required. **1** = Approvals required before checkin. **2** = Approvals required before advancing to the next area (check-off). **3** = Rule 1 and rule 2.
<releaseAreaAuditReturnCode>	Optional	0 - 1	String (2)	Release area audit return code.
<releaseAreaAuditRule>	Optional	0 - 1	Integer (1)	Release area audit rule. Values: **0** = Audit optional. **1** = Audit required, RC<20 (audit failure). **2** = Audit required, RC<=12 (out-of-sync errors within audited area). **3** = Audit required, RC<=8 (out-of-sync errors with respect to next areas/final areas in prior releases/baseline). **4** = Audit required, RC<=4 (no out-of-sync errors but some duplicates exist). **5** = Audit required, RC=0 (no out-of-sync errors and no warnings).
<releaseAreaBlockingRule>	Optional	0 - 1	Integer (1)	Release area blocking rule. Values: **0** = Wide open. **1** = Audit required. **2** = User must pass entity check (and rule 0). **3** = Rule 1 and rule 2.
<releaseAreaBlockingRuleEntity>	Optional	0 - 1	String (8), variable	Release area blocking rule entity.
<releaseAreaCheckinRule>	Optional	0 - 1	Integer (1)	Release area checkin rule. Values: **0** = Wide open. **1** = Area or package audit required before moving to next area. **2** = Area must be blocked (or package frozen) before moving to next area. **3** = User must pass entity check (and rule 0). **4** = Rule 1 and rule 2. **5** = Rule 1 and rule 3. **6** = Rule 2 and rule 3. **7** = Rule 1, rule 2, and rule 3.
<releaseAreaCheckinRuleEntity>	Optional	0 - 1	String (8), variable	Release area checkin rule entity.
<releaseAreaDesc>	Optional	0 - 1	String (72), variable	Release area description.
<releaseAreaRetrieveRule>	Optional	0 - 1	Integer (1)	Release area retrieve rule. Values: **0** = Retrieve wide open for any area, blocked or unblocked. **1** = Retrieve only from unblocked areas. **2** = Must pass entity check (and rule 0). **3** = Rule 1 and rule 2.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<releaseAreaRetrieveRuleEntity>	Optional	0 - 1	String (8), variable	Release area retrieve rule entity.
<releaseAreaStepNumber>	Optional	0 - 1	String (4)	Release area step number.
<releaseAreaType>	Optional	0 - 1	String (1)	Release area type. Values: **0** = Subsystem **1** = System
<releaseBuildNumber>	Optional	0 - 1	String (10), variable	Release area build number.
<releaseNextReleaseArea>	Optional	0 - 1	String (8), variable	Next release area name in this release.
<releasePriorReleaseArea>	Optional	0 - 1	String (8), variable	Prior release area name in this release.

RLSMAREA SERVICE TEST

The RLSMAREA SERVICE TEST message tests the contents of a release area against all of the packages that may place a component in that area. The reply message does not return any data, only a response with a message describing the status of packages and components in the release area.

The XML service/scope/message tags and attributes for this message are:

```
<service name="RLSMAREA">
<scope name="SERVICE">
<message name="TEST">
```

These tags appear in both requests and replies.

RLSMAREA SERVICE TEST — Request

The following example shows how you might code a request to test the contents of a release area. Data structure details for the `<request>` tag follow the example.

NOTE Job cards are only required if you are using the auto-cleanup subtags:
<cleanupComponentFromDiffPkg>
<cleanupEmptyPackage>
<cleanupNotCheckedInComponent>
For example, job cards would be required if an auto-cleanup operation needed to demote a component prior to deletion.

Example XML — RLSMAREA SERVICE TEST Request

```
<?xml version="1.0"?>
<service name="RLSMAREA">
 <scope name="SERVICE">
  <message name="TEST">
   <header>
    <subsys>4</subsys>
```

```
    <test> </test>
    <product>CMN</product>
   </header>
   <request>
    <release>S4711010</release>
    <releaseArea>GENLEDGR</releaseArea>
   </request>
  </message>
 </scope>
</service>
```

RLSMAREA SERVICE TEST <request> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<includeExcludePackage>	Optional	1	String (1), variable	Include or exclude package (I/X).
<package>	Optional	1	String (10), variable	Package name.
<release>	Required	1	String (8), variable	Release name.
<releaseArea>	Required	1	String (8), variable	Release area name.

RLSMAREA SERVICE TEST — Reply

The XML reply to a RLSMAREA SERVICE TEST request does not return any `<result>` data elements. The `<response>` data element indicates the success or failure of the request. Successful requests have a return code of 00. Unsuccessful requests have a return code of 04 or higher.

The following examples show what a reply message might look for a successful and unsuccessful request.

Example XML — RLSMAREA SERVICE TEST Replies

Successful Request

```
<?xml version="1.0"?>
<service name="RLSMAREA">
 <scope name="SERVICE">
  <message name="TEST">
   <response>
    <statusMessage>CMR1507I - Release S4711010/GENLEDGR and package
     components match.</statusMessage>
    <statusReturnCode>00</statusReturnCode>
    <statusReasonCode>1507</statusReasonCode>
   </response>
  </message>
 </scope>
</service>
```

Unsuccessful Request

```
<?xml version="1.0"?>
```

```
<service name="RLSMAREA">
 <scope name="SERVICE">
  <message name="TEST">
   <response>
    <statusMessage>CMR1506I - Release S4711010/ACCTPAY and package
     components do not match.</statusMessage>
    <statusReturnCode>08</statusReturnCode>
    <statusReasonCode>1506</statusReasonCode>
   </response>
  </message>
 </scope>
</service>
```

RLSMAREA SOURCE SYSLIB

The RLSMAREA SOURCE SYSLIB message lists the source SYSLIB information for a library type in a release application.

The XML service/scope/message tags and attributes for this message are:

```
<service name="RLSMAREA">
<scope name="SOURCE">
<message name="SYSLIB">
```

These tags appear in both requests and replies.

RLSMAREA SOURCE SYSLIB — Request

The following example shows how you might code a request to list the source SYSLIB information for a library type in a release application. Data structure details for the `<request>` tag follow the example.

Example XML — RLSMAREA SOURCE SYSLIB Request

```
<?xml version="1.0"?>
<service name="RLSMAREA">
 <scope name="SOURCE">
  <message name="SYSLIB">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
    <release>S4711010</release>
    <releaseArea>ACCTPAY </releaseArea>
    <releaseApplName>ACTP</releaseApplName>
    <libType>LOD</libType>
    <package>          </package>
   </request>
  </message>
 </scope>
</service>
```

RLSMAREA SOURCE SYSLIB <request> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<libType>	Required	1	String (3), variable	Release application library type.
<package>	Optional	0 - 1	String (10), variable	Release application package name.
<release>	Required	1	String (8), variable	Release name.
<releaseApplName>	Required	1	String (4), variable	Release application name.
<releaseArea>	Required	1	String (8), variable	Release area name.
<releaseParms>	Optional	1	String (1), variable	Release parms C or blank.

RLSMAREA SOURCE SYSLIB — Reply

The XML reply to a RLSMAREA SOURCE SYSLIB request returns one `<result>` data element, which lists the SYSLIB data for the requested library type for a release application.

The standard `<response>` data element follows any `<result>` tags in the reply and indicates the success or failure of the request. Successful requests have a return code of 00. Unsuccessful requests have a return code of 04 or higher.

The following example shows what the reply message might look like. Data structure details for the `<result>` tag are identical to those for the RLSMAREA ALL_CHK SYSLIB message - see "RLSMAREA ALL_CHK SYSLIB <result> Data Structure" on page 44.

Example XML — RLSMAREA SOURCE SYSLIB Reply

```
<?xml version="1.0"?>
<service name="RLSMAREA">
 <scope name="SOURCE">
  <message name="SYSLIB">
   <result>
    <release>S4711010</release>
    <releaseArea>ACCTPAY</releaseArea>
    <releaseApplName>ACTP</releaseApplName>
    <libType>LOD</libType>
    <likeType>L</likeType>
    <concatType>S</concatType>
    <libraryFromType>B</libraryFromType>
    <library>CMNTP.S4.V711.BASE.ACTP.LOD</library>
    <libraryOrg>PDS</libraryOrg>
   </result>
   <response>
    <statusMessage>SER8209I Logon accepted for user MTULLY; Local CCSID=00037
     </statusMessage>
    <statusReturnCode>00</statusReturnCode>
    <statusReasonCode>8700</statusReasonCode>
   </response>
```

```
  </message>
 </scope>
</service>
```

RLSMAREA START LIST

The RLSMAREA START LIST message lists the release area definitions for a named release and starting area.

The XML service/scope/message tags and attributes for this message are:

```
<service name="RLSMAREA">
<scope name="START">
<message name="LIST">
```

These tags appear in both requests and replies.

RLSMAREA START LIST — Request

The following example shows how you might code a request to list the definitions for a starting release area in a release. Data structure details for the <request> tag follow the example.

Example XML — RLSMAREA START LIST Request

```
<?xml version="1.0"?>
<service name="RLSMAREA">
 <scope name="START">
  <message name="LIST">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
    <release>S4711010</release>
    <releaseArea>ACCTPAY</releaseArea>
   </request>
  </message>
 </scope>
</service>
```

RLSMAREA START LIST <request> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<release>	Required	1	String (8), variable	Release name.
<releaseArea>	Required	1	String (8), variable	Release starting area name.

RLSMAREA START LIST — Reply

The XML reply to a RLSMAREA START LIST request returns one `<result>` data element, which contains information for a starting area of a release.

The standard `<response>` data element follows any `<result>` tags in the reply and indicates the success or failure of the request. Successful requests have a return code of 00. Unsuccessful requests have a return code of 04 or higher.

The following example shows what the reply message might look like. Data structure details for the `<result>` tag are identical to those for the RLSMAREA SERVICE LIST message - see "RLSMAREA SERVICE LIST <result> Data Structure" on page 83.

Example XML — RLSMAREA START LIST Reply

```
<?xml version="1.0"?>
<service name="RLSMAREA">
 <scope name="START">
  <message name="LIST">
   <result>
    <release>S4711010</release>
    <releaseArea>ACCTPAY</releaseArea>
    <releaseAreaDesc>Starting area for Accounts Payable components
     </releaseAreaDesc>
    <releaseAreaType>0</releaseAreaType>
    <releaseAreaStepNumber>0001</releaseAreaStepNumber>
    <releaseNextReleaseArea>FINANCE</releaseNextReleaseArea>
    <isReleaseAreaBlocked>N</isReleaseAreaBlocked>
    <isReleaseAreaApprovedForCheckin>Y</isReleaseAreaApprovedForCheckin>
    <isReleaseAreaApprovedForCheckoff>N</isReleaseAreaApprovedForCheckoff>
    <isReleaseAreaRejectedForCheckin>N</isReleaseAreaRejectedForCheckin>
    <isReleaseAreaRejectedForCheckoff>N</isReleaseAreaRejectedForCheckoff>
    <releaseAreaAuditRule>0</releaseAreaAuditRule>
    <releaseAreaBlockingRule>0</releaseAreaBlockingRule>
    <releaseAreaCheckinRule>0</releaseAreaCheckinRule>
    <releaseAreaRetrieveRule>0</releaseAreaRetrieveRule>
    <releaseAreaApproveRule>0</releaseAreaApproveRule>
    <isReleaseAreaAuditPending>N</isReleaseAreaAuditPending>
    <isReleaseAreaCheckinPending>N</isReleaseAreaCheckinPending>
    <isReleaseAreaApprovalPending>N</isReleaseAreaApprovalPending>
    <isReleaseAreaBlockPending>N</isReleaseAreaBlockPending>
    <addRelatedApprovers>Y</addRelatedApprovers>
    <allowComponentCheckout>Y</allowComponentCheckout>
    <excludeAreaFromConcatenation>N</excludeAreaFromConcatenation>
    <overrideOverlaidComponent>N</overrideOverlaidComponent>
    <bypassAreaTestDevPkg>N</bypassAreaTestDevPkg>
    <bypassAreaTestFrzPkg>N</bypassAreaTestFrzPkg>
    <bypassAreaTestAprPkg>N</bypassAreaTestAprPkg>
    <bypassAreaTestEmptyPkg>N</bypassAreaTestEmptyPkg>
    <bypassAreaTestPkgIntg>N</bypassAreaTestPkgIntg>
   </result>
   </reponse>
    <statusMessage>CMR8700I - Area Information service completed
     </statusMessage>
    <statusReturnCode>00</statusReturnCode>
    <statusReasonCode>8700</statusReasonCode>
   </response>
  </message>
 </scope>
```

```
</service>
```

RLSMAREA SUMMARY CMP_RLSE

The RLSMAREA SUMMARY CMP_RLSE message lists information for the latest version of each component in a release concatenation.

The XML service/scope/message tags and attributes for this message are:

```
<service name="RLSMAREA">
<scope name="SUMMARY">
<message name="CMP_RLSE">
```

These tags appear in both requests and replies.

RLSMAREA SUMMARY CMP_RLSE — Request

The following example shows how you might code a request to list information for the latest version of SRC-type components in a release concatenation. Data structure details for the <request> tag follow the example.

Example XML — RLSMAREA SUMMARY CMP_RLSE Request

```
<?xml version="1.0"?>
<service name="RLSMAREA">
 <scope name="SUMMARY">
  <message name="CMP_RLSE">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
    <release>S4711010</release>
    <releaseArea>ACCTPAY </releaseArea>
    <applName>    </applName>
    <libType>SRC</libType>
    <component>                                                    </component>
    <caseMixed> </caseMixed>
   </request>
  </message>
 </scope>
</service>
```

RLSMAREA SUMMARY CMP_RLSE <request> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<applName>	Optional	0 - 1	String (4), variable	Release application name.
<caseMixed>	Optional	0 - 1	String (1)	**Y** = Case is mixed. **N** = Case is not mixed.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<component>	Optional	0 - 1	String (256), variable	Component name. *NOTE:* May be masked using asterisk (*) wildcard.
<libType>	Required	0 - 1	String (3), variable	Release library type.
<listType>	Optional	0 - 1	String (1), variable	Component list type.
<release>	Required	1	String (8), variable	Release name.
<releaseArea>	Required	1	String (8), variable	Release area name.

RLSMAREA SUMMARY CMP_RLSE — Reply

The XML reply to a RLSMAREA SUMMARY CMP_RLSE request returns zero to many `<result>` data elements. Each result contains information for a component/library type.

The standard `<response>` data element follows any `<result>` tags in the reply and indicates the success or failure of the request. Successful requests have a return code of 00. Unsuccessful requests have a return code of 04 or higher. Because it is the final data element in the XML reply message, the `<response>` tag serves as an end-of-list marker.

The following example shows what the reply message might look like. Data structure details for the `<result>` tag follow the example.

Example XML — RLSMAREA SUMMARY CMP_RLSE Reply

```
<?xml version="1.0"?>
<service name="RLSMAREA">
 <scope name="SUMMARY">
  <message name="CMP_RLSE">
   <result>
    <release>BASELINE</release>
    <applName>ACTP</applName>
    <component>ACPSRCCA</component>
    <libType>SRC</libType>
    <package>ACTP000050</package>
    <releaseAreaToCheckinDate>20120527</releaseAreaToCheckinDate>
    <releaseAreaToCheckinTime>042200</releaseAreaToCheckinTime>
    <likeType>S</likeType>
    <libName>CMNTP.S4.V711.BASE.ACTP.SRC</libName>
   </result>
   <result>
    <release>BASELINE</release>
    <applName>ACTP</applName>
    <component>ACPSRCCC</component>
    <libType>SRC</libType>
    <package>ACTP000050</package>
    <releaseAreaToCheckinDate>20120528</releaseAreaToCheckinDate>
    <releaseAreaToCheckinTime>055000</releaseAreaToCheckinTime>
    <likeType>S</likeType>
    <libName>CMNTP.S4.V711.BASE.ACTP.SRC</libName>
   </result>
```

```
  <result>
   <release>BASELINE</release>
   <applName>ACTP</applName>
   <component>ACPSRCCE</component>
   <libType>SRC</libType>
   <package>ACTP000076</package>
   <releaseAreaToCheckinDate>20120827</releaseAreaToCheckinDate>
   <releaseAreaToCheckinTime>135500</releaseAreaToCheckinTime>
   <likeType>S</likeType>
   <libName>CMNTP.S4.V711.BASE.ACTP.SRC</libName>
  </result>
  .
  .
  .
  <response>
   <statusMessage>CMR9570I - Component list service completed
    </statusMessage>
   <statusReturnCode>00</statusReturnCode>
   <statusReasonCode>9570</statusReasonCode>
  </response>
 </message>
</scope>
</service>
```

RLSMAREA SUMMARY CMP_RLSE <result> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<applName>	Optional	0 - 1	String (4), variable	Release application name.
<component>	Optional	0 - 1	String (256), variable	Component name.
<libName>	Required	0 - 1	String (44), variable	Library name where component resides.
<libType>	Required	0 - 1	String (3)	Component library type.
<likeType>	Optional	0 - 1	String (1)	Release application library "like type". Values for all library types: **C** = Copy **L** = Load **P** = PDS **S** = Source **N** = NCAL **O** = Object **K** = LCT (link control cards) **X** = List **J** = JCL/Proc
<package>	Optional	0 - 1	String (10)	Release package name.
<release>	Required	0 - 1	String (8), variable	Release name.
<releaseArea>	Required	0 - 1	String (8), variable	Release area name.
<releaseAreaToCheckinDate>	Optional	0 - 1	Date, yyyymmdd	Release area checkin to date.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<releaseAreaToCheckinTime>	Optional	0 - 1	Time, hhmmss	Release area component checkin time.
<sortNumber>	Required	0 - 1	Integer	Sort number for sorting list.

RLSMAREA SUMMARY INTEGRTY

The RLSMAREA SUMMARY INTEGRTY message checks the integrity of the component-in-motion (CIM) table against physical members in the release area libraries. This function is similar to the RLSMAREA DETAIL INTEGRTY service; the only difference is that detail information is not returned for each mismatch, only for the first one. Once a mismatch is found, the process stops and an error message is returned.

The XML service/scope/message tags and attributes for this message are:

```
<service name="RLSMAREA">
<scope name="SUMMARY">
<message name="INTEGRTY">
```

These tags appear in both requests and replies.

RLSMAREA SUMMARY INTEGRTY — Request

The following example shows how you might code a request to check the integrity of a named release. Data structure details for the <request> tag follow the example.

Example XML — RLSMAREA SUMMARY INTEGRTY Request

```
<?xml version="1.0"?>
<service name="RLSMAREA">
 <scope name="SUMMARY">
  <message name="INTEGRTY">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
    <release>S4711010</release>
    <releaseArea>        </releaseArea>
    <applName>ACTP    </applName>
   </request>
  </message>
 </scope>
</service>
```

RLSMAREA SUMMARY INTEGRTY <request> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<applName>	Required	1	String (4), variable	Release application name.
<release>	Required	1	String (8), variable	Release name.
<releaseArea>	Optional	0 - 1	String (8), variable	Release area name.

RLSMAREA SUMMARY INTEGRTY — Reply

The XML reply to a RLSMAREA SUMMARY INTEGRTY request returns zero to one `<result>` data elements. If there are no mismatched components, only a `<response>` data element is returned with a message indicating that the service completed successfully. If a mismatched component is found, a `<result>` data element is returned for the first mismatch only, followed by a `<response>` data element.

The following examples show what the reply message might look if a mismatch is found. Data structure details for the `<result>` tag follow the example.

Example XML — RLSMAREA SUMMARY INTEGRTY Reply

```
<?xml version="1.0"?>
<service name="RLSMAREA">
 <scope name="SUMMARY">
  <message name="INTEGRTY">
   <result>
    <releaseArea>FINANCE</releaseArea>
    <applName>ACTP</applName>
    <libType>CPY</libType>
    <component>ACPCPYCE</component>
    <isCIMMissing>N</isCIMMissing>
    <isMemberMissing>Y</isMemberMissing>
   </result>
   <response>
    <statusMessage>CMR9569I - CIM records and library/file contents do not
     match</statusMessage>
    <statusReturnCode>00</statusReturnCode>
    <statusReasonCode>9569</statusReasonCode>
   </response>
  </message>
 </scope>
</service>
```

RLSMAREA SUMMARY INTEGRTY <result> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<applName>	Optional	0 - 1	String (4), variable	Release application name.
<component>	Optional	0 - 1	String (256), variable	Mismatched component name.
<isCIMMissing>	Optional	0 - 1	String (1)	**Y** = component is missing from CIM table. **N** = component exists in CIM table.
<isMemberMissing>	Optional	0 - 1	String (1)	**Y** = component is missing from release area library. **N** = component exists in release area library.
<libType>	Optional	0 - 1	String (3)	Component library type.
<release>	Optional	0 - 1	String (8), variable	Release name.
<releaseArea>	Optional	0 - 1	String (8), variable	Release area name.

RLSMAREA SYSLIB LIST

The RLSMAREA SYSLIB LIST message lists the data that defines a SYSLIB for an application.

The XML service/scope/message tags and attributes for this message are:

```
<service name="RLSMAREA">
<scope name="SYSLIB">
<message name="LIST">
```

These tags appear in both requests and replies.

RLSMAREA SYSLIB LIST — Request

The following example shows how you might code a request to list SYSLIB data for a release. Data structure details for the `<request>` tag follow the example.

Example XML — RLSMAREA SYSLIB LIST Request

```
<?xml version="1.0"?>
<service name="RLSMAREA">
 <scope name="SYSLIB">
  <message name="LIST">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
```

```
    <release>S4711010</release>
    <releaseArea>ACCTPAY </releaseArea>
    <releaseApplName>    </releaseApplName>
   </request>
  </message>
 </scope>
</service>
```

RLSMAREA SYSLIB LIST <request> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<release>	Required	1	String (8), variable	Release name.
<releaseApplName>	Optional	0 - 1	String (4), variable	Release application name.
<releaseArea>	Required	1	String (8), variable	Release area name.

RLSMAREA SYSLIB LIST — Reply

The XML reply to a RLSMAREA SYSLIB LIST request returns zero to many `<result>` data elements. Each result contains SYSLIB information for a release area.

The standard `<response>` data element follows any `<result>` tags in the reply and indicates the success or failure of the request. Successful requests have a return code of 00. Unsuccessful requests have a return code of 04 or higher. Because it is the final data element in the XML reply message, the `<response>` tag serves as an end-of-list marker.

The following example shows what the reply message might look like. The requested release area, ACCTPAY, is a starting area. The reply message lists the ACCTPAY area as well as the system area, FINANCE.

Data structure details for the `<result>` tag follow the example.

Example XML — RLSMAREA SYSLIB LIST Reply

```
<?xml version="1.0"?>
<service name="RLSMAREA">
 <scope name="SYSLIB">
  <message name="LIST">
   <result>
    <release>S4711010</release>
    <releaseInstallDate>20131005</releaseInstallDate>
    <releaseInstallTime>000100</releaseInstallTime>
    <releaseHighLevelName>CMNTP</releaseHighLevelName>
    <releaseDsnPattern>HRAPL</releaseDsnPattern>
    <releaseHighLevelPath>/cmntp</releaseHighLevelPath>
    <releaseArea>ACCTPAY</releaseArea>
    <releaseAreaType>0</releaseAreaType>
    <releaseAreaStepNumber>00001</releaseAreaStepNumber>
    <releaseAreaFromType>C</releaseAreaFromType>
    <releaseNextReleaseArea>FINANCE</releaseNextReleaseArea>
   </result>
   <result>
```

```
    <release>S4711010</release>
    <releaseInstallDate>20131005</releaseInstallDate>
    <releaseInstallTime>000100</releaseInstallTime>
    <releaseHighLevelName>CMNTP</releaseHighLevelName>
    <releaseDsnPattern>HRAPL</releaseDsnPattern>
    <releaseHighLevelPath>/cmntp</releaseHighLevelPath>
    <releaseArea>FINANCE</releaseArea>
    <releaseAreaType>1</releaseAreaType>
    <releaseAreaStepNumber>00003</releaseAreaStepNumber>
    <releaseAreaFromType>C</releaseAreaFromType>
    <releasePriorReleaseArea>ACCTPAY</releasePriorReleaseArea>
   </result>
   <response>
    <statusMessage>CMR8700I - Application SYSLIB service completed
     </statusMessage>
    <statusReturnCode>00</statusReturnCode>
    <statusReasonCode>8700</statusReasonCode>
   </response>
  </message>
 </scope>
</service>
```

RLSMAREA SYSLIB LIST <result> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<release>	Optional	0 - 1	String (8), variable	Release name.
<releaseApplName>	Optional	0 - 1	String (4), variable	Release application name.
<releaseArea>	Optional	0 - 1	String (8), variable	Release area name.
<releaseAreaFromType>	Optional	0 - 1	String (1)	Release area "from" type. Values: **C** = Current release area "from" type. **P** = Prior release area "from" type.
<releaseAreaStepNumber>	Optional	0 - 1	String (5)	Release area step number.
<releaseAreaType>	Optional	0 - 1	String (1)	Release area type. Values: **0** = Subsystem **1** = System
<releaseDsnPattern>	Optional	0 - 1	String (5), variable	Release DSN pattern.
<releaseHighLevelName>	Optional	0 - 1	String (8), variable	Release high-level qualifier.
<releaseHighLevelPath>	Optional	0 - 1	String (40), variable	Release high-level path name.
<releaseInstallDate>	Optional	0 - 1	Date, yyyymmdd	Release install date.
<releaseInstallTime>	Optional	0 - 1	Time, hhmmss	Release install time.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<releaseNextReleaseArea>	Optional	0 - 1	String (8), variable	Next release area name in this release.
<releasePriorReleaseArea>	Optional	0 - 1	String (8), variable	Prior release area name in this release.

RLSMAREA VER_REGR LIST

The RLSMAREA VER_REGR LIST message performs a version regression check on components. If a version regression situation exists between the current release and a prior release, information for the current and prior versions is listed.

The XML service/scope/message tags and attributes for this message are:

```
<service name="RLSMAREA">
<scope name="VER_REGR">
<message name="LIST">
```

These tags appear in both requests and replies.

RLSMAREA VER_REGR LIST — Request

The following example shows how you might code a request to perform a version regression check for a single component. Data structure details for the `<request>` tag follow the example.

Example XML — RLSMAREA VER_REGR LIST Request

```
<?xml version="1.0"?>
<service name="RLSMAREA">
 <scope name="VER_REGR">
  <message name="LIST">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
    <package>ACTP000478</package>
    <libType>SRC</libType>
    <component>AA002                                                    </component>
   </request>
  </message>
 </scope>
</service>
```

RLSMAREA VER_REGR LIST <request> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<component>	Required	1	String (256), variable	Component name. ***NOTE:*** May be masked using asterisk (*) wildcard. However, processing a single component uses a lot of resources, so it is recommended that you request only a single component or a small number of components.
<libType>	Required	1	String (3), variable	Library type.
<package>	Required	1	String (10), variable	The package from which the area component was checked in.

RLSMAREA VER_REGR LIST — Reply

The XML reply to a RLSMAREA VER_REGR LIST request returns zero to many `<result>` data elements.

If a version regression situation does not exist, an informational message is displayed in the `<response>` data element, but no `<result>` data element is returned.

If a version regression situation does exist, a `<result>` data element is returned for each component, listing information for the current and prior releases.

The standard `<response>` data element follows any `<result>` tags in the reply and indicates the success or failure of the request. Successful requests have a return code of 00. Unsuccessful requests have a return code of 04 or higher. Because it is the final data element in the XML reply message, the `<response>` tag serves as an end-of-list marker.

The following examples show what the reply message might look like. Data structure details for the `<result>` tag follow the examples.

Example XML — RLSMAREA VER_REGR LIST Reply

Example of No Version Regression

The following example shows a reply where a version regression situation does not exist:

```
<?xml version="1.0"?>
<service name="RLSMAREA">
 <scope name="VER_REGR">
  <message name="LIST">
   <response>
    <statusMessage>CMR6504I - No information found for Prior rls regress
      request.</statusMessage>
    <statusReturnCode>08</statusReturnCode>
    <statusReasonCode>6504</statusReasonCode>
   </response>
  </message>
 </scope>
</service>
```

Example of a Version Regression

The following example shows a reply where a version regression situation does exist:

```
<?xml version="1.0"?>
<service name="RLSMAREA">
 <scope name="VER_REGR">
  <message name="LIST">
   <result>
    <package>ACTP000478</package>
    <applName>ACTP</applName>
    <packageId>000478</packageId>
    <libType>SRC</libType>
    <component>AA002</component>
    <currentRelease>SDJUL112</currentRelease>
    <currentArea>UNIT</currentArea>
    <priorRelease>SDJUL111</priorRelease>
    <priorArea>UNIT</priorArea>
    <currentUser>WSER58</currentUser>
    <priorUser>WSER58</priorUser>
    <currentChangeDate>20110812</currentChangeDate>
    <currentChangeTime>033517</currentChangeTime>
    <priorChangeDate>20110822</priorChangeDate>
    <priorChangeTime>072426</priorChangeTime>
    <currentCheckInDate>20110812</currentCheckInDate>
    <currentCheckInTime>033610</currentCheckInTime>
    <priorCheckInDate>20110822</priorCheckInDate>
    <priorCheckInTime>072540</priorCheckInTime>
    <currentVersion>01</currentVersion>
    <currentModLevel>00</currentModLevel>
    <priorVersion>01</priorVersion>
    <priorModLevel>00</priorModLevel>
    <currentHashToken>D4C03B4A0000023E</currentHashToken>
    <priorHashToken>A10E947500000290</priorHashToken>
    <currentAssocPkg>ACTP000472</currentAssocPkg>
    <priorAssocPkg>ACTP000472</priorAssocPkg>
   </result>
   <response>
    <statusMessage>CMR8700I - Prior rls regress service completed
     </statusMessage>
    <statusReturnCode>00</statusReturnCode>
    <statusReasonCode>8700</statusReasonCode>
   </response>
  </message>
 </scope>
</service>
```

Most of the fields returned in the above example are informational, but the following fields are important to note:

`<currentHashToken>D4C03B4A0000023E</currentHashToken>`

This was the hash token of the current area component at the time it was originally checked out. If nothing has changed in the meantime then this field should be the same as the next one (`priorHashToken`).

`<priorHashToken>A10E947500000290</priorHashToken>`

This is the hash token of the component which is now in the "first found" prior release. The fact that it is different means that the prior release has been changed since the current release component was checked out.

`<currentAssocPkg>ACTP000472</currentAssocPkg>`

This is the name of the package which supplied the prior release component at the time the current release component was checked out. Again, for there to be no prior release version regression, this must also match the next field (`priorAssocPkg`).

`<priorAssocPkg>ACTP000472</priorAssocPkg>`

This is the name of the package which supplied the prior release component as it exists now. The fact that these are the same in this case but the hash tokens are different means that the prior release version has been updated in package ACTP000472 and then checked back in.

If you are satisfied that the current area component is up to date with all recently applied changes to earlier releases, you can reset the meta data, which will resolve an ERR0417 in the ERO release area audit. For information on how to reset the meta data using the ChangeMan ZMF ISPF interface, refer to:

ChangeMan ZMF ERO Getting Started Guide
"Auditing Release Areas" chapter
"Repair ERR0417 Prior Release Version Regression" section

RLSMAREA VER_REGR LIST <result> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<applName>	Optional	0 - 1	String (4), variable	Release application name.
<component>	Optional	0 - 1	String (256), variable	Release area component name.
<currentArea>	Optiona	0 - 1	String (8), variable	Current release area name.
<currentAssocPkg>	Optional	0 - 1	String (10)	Current associated package name.
<currentChangeDate>	Optional	0 - 1	Date, yyyymmdd	Current change date.
<currentChangeTime>	Optional	0 - 1	Time, hhmmss	Current change time.
<currentCheckinDate>	Optional	0 - 1	Date, yyyymmdd	Current checkin date.
<currentCheckinTime>	Optional	0 - 1	Time, hhmmss	Current checkin time.
<currentHashToken>	Optional	0 - 1	String (16)	Current hash token.
<currentModLevel>	Optional	0 - 1	String (2)	Current modification level.
<currentRelease>	Optiona	0 - 1	String (8), variable	Current release name.
<currentUser>	Optiona	0 - 1	String (8), variable	Current user ID.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<currentVersion>	Optiona	0 - 1	String (2)	Current version.
<libType>	Optiona	0 - 1	String (3)	Library type.
<package>	Optional	0 - 1	String (10)	The package from which the area component was checked in.
<packageId>	Optional	0 - 1	Integer (6)	Package number.
<priorArea>	Optional	0 - 1	String (8), variable	Prior release area name.
<priorAssocPkg>	Optional	0 - 1	String (10)	Prior associated package name.
<priorChangeDate>	Optional	0 - 1	Date, yyyymmdd	Prior change date.
<priorChangeTime>	Optional	0 - 1	Time, hhmmss	Prior change time.
<priorCheckinDate>	Optional	0 - 1	Date, yyyymmdd	Prior checkin date.
<priorCheckinTime>	Optional	0 - 1	Time, hhmmss	Prior checkin time.
<priorHashToken>	Optional	0 - 1	String (16)	Prior hash token.
<priorModLevel>	Optional	0 - 1	String (2)	Prior modification level.
<priorRelease>	Optional	0 - 1	String (8), variable	Prior release name.
<priorUser>	Optional	0 - 1	String (8), variable	Prior user ID.
<priorVersion>	Optional	0 - 1	String (2)	Prior version.

RLSMLTYP BUN LIST

The RLSMLTYP BUN LIST message lists information from the release BUN library-type table.

The XML service/scope/message tags and attributes for this message are:

```
<service name="RLSMLTYP">
<scope name="BUN">
<message name="LIST">
```

These tags appear in both requests and replies.

RLSMLTYP BUN LIST — Request

The following example shows how you might code a request to list information from the BUN library-type table for a named application. Data structure details for the `<request>` tag follow the example.

Example XML — RLSMLTYP BUN LIST Request

```
<?xml version="1.0"?>
<service name="RLSMLTYP">
 <scope name="BUN">
  <message name="LIST">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
    <applName>ACTP</applName>
   </request>
  </message>
 </scope>
</service>
```

RLSMLTYP BUN LIST <request> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<applName>	Optional	0 - 1	String (4), variable	Application name in BUN table.
<baseLib>	Optional	0 - 1	String (44), variable	Baseline library dataset name.
<bunNumber>	Optional	0 - 1	Integer (10)	Baseline unique number.
<libType>	Optional	0 - 1	String (3)	Library type in BUN table.
<libTypeLike>	Optional	0 - 1	String (3)	Library "like" type (for example CPY).
<likeType>	Optional	0 - 1	String (1)	Component type attribute which determines how a component is processed. **C** = Copy **L** = Load **P** = PDS **S** = Source **N** = NCAL **O** = Object **K** = LCT (link control cards **X** = List **J** = JCL/Proc

RLSMLTYP BUN LIST — Reply

The XML reply to a RLSMLTYP BUN LIST request returns zero to many `<result>` data elements. Each result lists information about a library type.

The standard `<response>` data element follows any `<result>` tags in the reply and indicates the success or failure of the request. Successful requests have a return code of 00. Unsuccessful requests have a return code of 04 or higher. Because it is the final data element in the XML reply message, the `<response>` tag serves as an end-of-list marker.

The following examples show what the reply message might look like. Data structure details for the `<result>` tag follow the example.

Example XML — RLSMLTYP BUN LIST Reply

```
<?xml version="1.0"?>
<service name="RLSMLTYP">
 <scope name="BUN">
  <message name="LIST">
   <result>
    <applName>ACTP</applName>
    <libType>CPY</libType>
    <likeType>C</likeType>
    <libTypeLike>CPY</libTypeLike>
    <bunNumber>0000000001</bunNumber>
    <baseLib>CMNTP.S4.V711.BASE.ACTP.CPY</baseLib>
   </result>
   <result>
    <applName>ACTP</applName>
    <libType>CTC</libType>
    <likeType>P</likeType>
    <libTypeLike>PDS</libTypeLike>
    <bunNumber>0000000002</bunNumber>
    <baseLib>CMNTP.S4.V711.BASE.CTC</baseLib>
   </result>
   <result>
    <applName>ACTP</applName>
    <libType>DOC</libType>
    <likeType>P</likeType>
    <libTypeLike>PDS</libTypeLike>
    <bunNumber>0000000003</bunNumber>
    <baseLib>CMNTP.S4.V711.BASE.ACTP.DOC</baseLib>
   </result>
   .
   .
   .
   <response>
    <statusMessage>CMR8700I - Release BUN Table service completed
     </statusMessage>
    <statusReturnCode>00</statusReturnCode>
    <statusReasonCode>8700</statusReasonCode>
   </response>
  </message>
 </scope>
</service>
```

RLSMLTYP BUN LIST <result> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<applName>	Optional	0 - 1	String (4), variable	Application name in BUN table.
<baseLib>	Optional	0 - 1	String (44), variable	Baseline library dataset name.
<bunNumber>	Optional	0 - 1	Integer (10)	Baseline unique number.
<libType>	Optional	0 - 1	String (3)	Library type in BUN table.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<libTypeLike>	Optional	0 - 1	String (3)	Library "like" type (for example CPY).
<likeType>	Optional	0 - 1	String (1)	Component type attribute which determines how a component is processed. **C** = Copy **L** = Load **P** = PDS **S** = Source **N** = NCAL **O** = Object **K** = LCT (link control cards **X** = List **J** = JCL/Proc

RLSMLTYP SERVICE LIST

The RLSMLTYP SERVICE LIST message lists library security and format information.

The XML service/scope/message tags and attributes for this message are:

```
<service name="RLSMLTYP">
<scope name="SERVICE">
<message name="LIST">
```

These tags appear in both requests and replies.

RLSMLTYP SERVICE LIST — Request

The following example shows how you might code a request to list library security and format information for a named release and application. Data structure details for the `<request>` tag follow the example.

Example XML — RLSMLTYP SERVICE LIST Request

```
<?xml version="1.0"?>
<service name="RLSMLTYP">
 <scope name="SERVICE">
  <message name="LIST">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
    <release>S4711010</release>
    <applName>ACTP</applName>
   </request>
  </message>
 </scope>
</service>
```

RLSMLTYP SERVICE LIST <request> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<applName>	Optional	0 - 1	String (4), variable	Release application name.
<libType>	Optional	0 - 1	String (3)	Library type.
<likeType>	Optional	0 - 1	String (1)	Library "like" type. Component type attribute which determines how a component is processed. **C** = Copy **L** = Load **P** = PDS **S** = Source **N** = NCAL **O** = Object **K** = LCT (link control cards **X** = List **J** = JCL/Proc
<release>	Required	1	String (8), variable	Release name.

RLSMLTYP SERVICE LIST — Reply

The XML reply to a RLSMLTYP SERVICE LIST request returns zero to many `<result>` data elements. Each result lists library security and format information.

The standard `<response>` data element follows any `<result>` tags in the reply and indicates the success or failure of the request. Successful requests have a return code of 00. Unsuccessful requests have a return code of 04 or higher. Because it is the final data element in the XML reply message, the `<response>` tag serves as an end-of-list marker.

The following examples show what the reply message might look like. Data structure details for the `<result>` tag follow the example.

Example XML — RLSMLTYP SERVICE LIST Reply

```
<?xml version="1.0"?>
<service name="RLSMLTYP">
 <scope name="SERVICE">
  <message name="LIST">
   <result>
    <release>S4711010</release>
    <applName>ACTP</applName>
    <libType>CPY</libType>
    <libTypeDesc>Copybooks</libTypeDesc>
    <likeType>C</likeType>
    <unitName>SYSDA</unitName>
    <spaceType>TRK</spaceType>
    <primarySpace>00000001</primarySpace>
    <secondarySpace>00000010</secondarySpace>
    <dirBlocks>00010</dirBlocks>
    <recordFormat>FB</recordFormat>
    <recordLength>00080</recordLength>
    <blockSize>00000</blockSize>
```

```
    <isPdseLibType>N</isPdseLibType>
    <isPdsLibType>Y</isPdsLibType>
    <isDatasetSystemManaged>N</isDatasetSystemManaged>
    <isPdseProgramObject>N</isPdseProgramObject>
    <areaLibsCreated>N</areaLibsCreated>
    <ssvOptionAllowed>N</ssvOptionAllowed>
    <ssvOption>N</ssvOption>
    <isHfsLibType>N</isHfsLibType>
    <chkOutComponentGenDesc>N</chkOutComponentGenDesc>
    <chkOutActivityFile>N</chkOutActivityFile>
    <isImsLibType>N</isImsLibType>
    <isApsLibType>N</isApsLibType>
    <includeUtilityInfo>N</includeUtilityInfo>
    <submitProcess>N</submitProcess>
    <isDb2LibType>N</isDb2LibType>
    <db2Type>S</db2Type>
   </result>
   .
   .
   .
   <response>
    <statusMessage>CMR8700I - Release Lib. Type service completed
     </statusMessage>
    <statusReturnCode>00</statusReturnCode>
    <statusReasonCode>8700</statusReasonCode>
   </response>
  </message>
 </scope>
</service>
```

RLSMLTYP SERVICE LIST <result> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<applName>	Optional	0 - 1	String (4), variable	Application name in BUN table.
<apsDevLib>	Optional	0 - 1	String (44), variable	APS development library.
<apsEntity>	Optional	0 - 1	String (8), variable	APS security entity.
<areaLibsCreated>	Optional	0 - 1	String (1)	**Y** = Area libraries are allocated. **N** = Area libraries are not allocated.
<blockSize>	Optional	0 - 1	Integer (5).	Block size.
<chkOutActivityFile>	Optional	0 - 1	String (1)	**Y** = Checkout component activity file. **N** = Do not checkout component activity file.
<chkOutComponentGenDesc>	Optional	0 - 1	String (1)	**Y** = Checkout component general description. **N** = Do not checkout component general description.
<db2SqlTerminationChar>	Optional	0 - 1	String (1)	Db2 SQL sentence termination character.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<db2Type>	Optional	0 - 1	String (1)	Db2 entity type. Values: **B** = DBB bind contro **D** = Db2 general SQL/DDL **P** = Package bind control **Q** = Db2 SQL-based stored procedure source **R** = DBRM database request module **S** = Db2 stored procedure load/rexx **T** = Db2 trigger SQL
<dirBlocks>	Optional	0 - 1	Integer (5).	Directory blocks.
<eAttr>	Optional	0 - 1	String (1)	Extended attribute option. Values: **N** = Dataset cannot have extended attributes or reside in EAS. **O** = Dataset can have extended attributes and reside in EAS. **blank** = Default based on data type.
<imsType>	Optional	0 - 1	String (1)	IMS entity type. Values: **B** = DBD target **D** = DBD source **F** = FMT target **M** = MFS source **P** = PSB source **R** = REF target **S** = PSB target
<includeUtilityInfo>	Optional	0 - 1	String (1)	**Y** = Include utility component information. **N** = Do not include utility component information.
<isApsLibType>	Optional	0 - 1	String (1)	**Y** = Library type is APS. **N** = Library type is not APS.
<isDatasetSystemManaged>	Optional	0 - 1	String (1)	**Y** = Dataset is system-managed. **N** = Dataset is not system-managed.
<isDb2LibType>	Optional	0 - 1	String (1)	**Y** = Library type is Db2. **N** = Library type is not Db2.
<isHfsLibType>	Optional	0 - 1	String (1)	**Y** = Library type is HFS. **N** = Library type is not HFS.
<isImsLibType>	Optional	0 - 1	String (1)	**Y** = Library type is IMS. **N** = Library type is not IMS.
<isPdsLibType>	Optional	0 - 1	String (1)	**Y** = Library type is PDS. **N** = Library type is not PDS.
<isPdseLibType>	Optional	0 - 1	String (1)	**Y** = Library type is PDSE. **N** = Library type is not PDSE.
<isPdseProgramObject>	Optional	0 - 1	String (1)	**Y** = PDSE program object. **N** = Not a PDSE program object.
<libType>	Optional	0 - 1	String (3)	Library type in BUN table.
<libTypeDesc>	Optional	0 - 1	String (44) variable	Library type description.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<likeType>	Optional	0 - 1	String (1)	Component type attribute which determines how a component is processed. **C** = Copy **L** = Load **P** = PDS **S** = Source **N** = NCAL **O** = Object **K** = LCT (link control cards **X** = List **J** = JCL/Proc
<primarySpace>	Optional	0 - 1	Integer (8)	Primary space.
<recordFormat>	Optional	0 - 1	String (3), variable	Record format. Values: F = Fixed FA = Fixed ASA FM = Fixed Machine FB = Fixed Block FBA = Fixed Block ASA FBM = Fixed Block Machine FBS = Fixed Block Standard FS = Fixed Standard FSA = Fixed Standard ASA FSM = Fixed Standard Machine V = Variable VA = Variable ASA VM = Variable Machine VB = Variable Block VBA = Variable Block ASA VBM = Variable Block Machine VS = Variable Spanned VSA = Variable Spanned ASA VSM =Variable Spanned Machine U = Undefined UA = Undefined ASA UM = Undefined Machine UB = Undefined Block UBA = Undefined Block ASA UBM = Undefined Block Machine US = Undefined Spanned USA = Undefined Spanned ASA USM = Undefined Spanned Machine
<recordLength>	Optional	0 - 1	Integer (5)	Library type description.
<release>	Optional	0 - 1	String (8), variable	Release name.
<secondarySpace>	Optional	0 - 1	Integer (8)	Secondary space.
<spaceType>	Optional	0 - 1	String (3)	Unit type for space allocation. Values: **BLK** = Blocks **CYL** = Cylinders **TRK** = Tracks
<ssvOption>	Optional	0 - 1	String (1)	**Y** = SSV is enforced. **N** = SSV is not enforced.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<ssvOptionAllowed>	Optional	0 - 1	String (1)	**Y** = SSV is allowed. **N** = SSV is not allowed.
<submitProcess>	Optional	0 - 1	String (1)	This tag is not currently used.
<targetActivityFile>	Optional	0 - 1	String (3)	Target activity notification file.
<targetLoadLibFile>	Optional	0 - 1	String (3)	Target load library type.
<unitName>	Optional	0 - 1	String (8), variable	Unit name.
<volume>	Optional	0 - 1	String (6)	Volume serial number.

RLSMRLSE CIM LIST

The RLSMRLSE CIM LIST message lists release area component in motion (CIM) information from the ERO Db2 CIM table.

The XML service/scope/message tags and attributes for this message are:

```
<service name="RLSMRLSE">
<scope name="CIM">
<message name="LIST">
```

These tags appear in both requests and replies.

RLSMRLSE CIM LIST — Request

The following example shows how you might code a request to list component in motion information for a release. Data structure details for the `<request>` tag are identical to those for the RLSMAREA CIM LIST message except that the `<release>` subtag is optional - see "RLSMAREA CIM LIST <request> Data Structure" on page 47.

Example XML — RLSMRLSE CIM LIST Request

```
<?xml version="1.0"?>
<service name="RLSMRLSE">
 <scope name="CIM">
  <message name="LIST">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
    <release>S4712COM</release>
   </request>
  </message>
 </scope>
</service>
```

RLSMRLSE CIM LIST — Reply

The XML reply to a RLSMRLSE CIM LIST request returns zero to many `<result>` data elements. Each result lists a row of component in motion information from the ERO Db2 CIM table.

The standard `<response>` data element follows any `<result>` tags in the reply and indicates the success or failure of the request. Successful requests have a return code of 00. Unsuccessful requests have a return code of 04 or higher. Because it is the final data element in the XML reply message, the `<response>` tag serves as an end-of-list marker.

The following example shows what the reply message might look like. Data structure details for the `<result>` tag are identical to those for the RLSMAREA CIM LIST message - see "RLSMAREA CIM LIST <result> Data Structure" on page 50.

Example XML — RLSMRLSE CIM LIST Reply

```
<?xml version="1.0"?>
<service name="RLSMRLSE">
 <scope name="CIM">
  <message name="LIST">
   <result>
    <release>S4712COM</release>
    <releaseArea>FINANCE</releaseArea>
    <package>ACTP000076</package>
    <applName>ACTP</applName>
    <packageId>000076</packageId>
    <componentType>CPY</componentType>
    <likeType>C</likeType>
    <libraryStorageMeans>PDS</libraryStorageMeans>
    <component>ACPCPYCE</component>
    <checkinUser>KCAMPBE</checkinUser>
    <releaseAreaCheckinDate>20120827</releaseAreaCheckinDate>
    <releaseAreaCheckinTime>140138</releaseAreaCheckinTime>
    <loadModule>N</loadModule>
    <generateRequired>N</generateRequired>
    <componentChangeDate>20120827</componentChangeDate>
    <componentChangeTime>083512</componentChangeTime>
    <setssi>630B28C0</setssi>
    <hashToken>B7A76B0C00000079</hashToken>
    <releaseAreaGenerateDate>19000101</releaseAreaGenerateDate>
    <releaseAreaGenerateTime>000000</releaseAreaGenerateTime>
    <componentVersionNo>01</componentVersionNo>
    <componentModLevel>02</componentModLevel>
    <componentSizeLines>00000000</componentSizeLines>
    <componentSizeBytes>00000000</componentSizeBytes>
    <componentSizeInit>00000001</componentSizeInit>
    <componentCreateDate>20020507</componentCreateDate>
    <componentLmodRENTattr>N</componentLmodRENTattr>
    <componentLmodREUSattr>N</componentLmodREUSattr>
    <componentLmodOVLYattr>N</componentLmodOVLYattr>
    <componentLmodTESTattr>N</componentLmodTESTattr>
    <componentLmodOLODattr>N</componentLmodOLODattr>
    <componentLmodEXECattr>N</componentLmodEXECattr>
    <componentLmodRFRSattr>N</componentLmodRFRSattr>
    <currentHashToken>B7A75D0A00000079</currentHashToken>
    <priorHashToken>B7A75D0A00000079</priorHashToken>
    <currentAssocPkg>BASELINE</currentAssocPkg>
    <priorAssocPkg>BASELINE</priorAssocPkg>
```

```
   </result>
   .
   .
   .
   <response>
    <statusMessage>CMR8700I - Release CIM Table service completed
     </statusMessage>
    <statusReturnCode>00</statusReturnCode>
    <statusReasonCode>8700</statusReasonCode>
   </response>
  </message>
 </scope>
</service>
```

RLSMRLSE DETAIL TEST

The RLSMRLSE DETAIL TEST message tests the contents of a release against all of the packages that may place a component in that release. Only the final area of a release is tested.

The XML service/scope/message tags and attributes for this message are:

```
<service name="RLSMRLSE">
<scope name="DETAIL">
<message name="TEST">
```

These tags appear in both requests and replies.

RLSMRLSE DETAIL TEST — Request

The following example shows how you might code a request to test the contents of a release. Data structure details for the `<request>` tag follow the example.

Example XML — RLSMRLSE DETAIL TEST Request

```
<?xml version="1.0"?>
<service name="RLSMRLSE">
 <scope name="DETAIL">
  <message name="TEST">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
    <release>S4711010</release>
   </request>
  </message>
 </scope>
</service>
```

RLSMRLSE DETAIL TEST <request> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<cleanupComponentFromDiffPkg>	Optional	0 - 1	String (1)	**Y** = If different versions of a component exist in different packages, the version not checked into the release will be deleted from the package. **N** = Do not delete components with different versions.
<cleanupEmptyPackage>	Optional	0 - 1	String (1)	**Y** = Automatically cleanup empty packages. **N** = Do not cleanup empty packages.
<cleanupNotCheckedInComponent>	Optional	0 - 1	String (1)	**Y** = Automatically cleanup components that were not checked into release. **N** = Do not cleanup components that were not checked into release.
<jobCard01>	Required	1	String (72), variable	Job card 1
<jobCard02>	Required	1	String (72), variable	Job card 2
<jobCard03>	Required	1	String (72), variable	Job card 3
<jobCard04>	Required	1	String (72), variable	Job card 4
<release>	Required	1	String (8), variable	Release name.

RLSMRLSE DETAIL TEST — Reply

The XML reply to a RLSMRLSE DETAIL TEST request returns zero to many `<result>` data elements. Each result contains information for a failing component or package.

The standard `<response>` data element follows any `<result>` tags in the reply and indicates the success or failure of the request. Successful requests have a return code of 00. Unsuccessful requests have a return code of 04 or higher. Because it is the final data element in the XML reply message, the `<response>` tag serves as an end-of-list marker.

The following example shows what the reply message might look like. Data structure details for the `<result>` tag follow the example.

Example XML — RLSMRLSE DETAIL TEST Reply

```
<?xml version="1.0"?>
<service name="RLSMRLSE">
 <scope name="DETAIL">
  <message name="TEST">
   <result>
    <release>S4711010</release>
```

```
    <releaseArea>FINANCE</releaseArea>
    <package>ACTP000094</package>
    <packageStatus>DEV</packageStatus>
    <reasonCode>E</reasonCode>
    <reasonForFailure>Empty Package</reasonForFailure>
   </result>
   <result>
    <release>S4711010</release>
    <releaseArea>FINANCE</releaseArea>
    <package>ACTP000095</package>
    <packageStatus>DEV</packageStatus>
    <reasonCode>E</reasonCode>
    <reasonForFailure>Empty Package</reasonForFailure>
   </result>
   <response>
    <statusMessage>CMR1506I - Release S4711010/FINANCE and package
     components do not match.</statusMessage>
    <statusReturnCode>00</statusReturnCode>
    <statusReasonCode>1506</statusReasonCode>
   </response>
  </message>
 </scope>
</service>
```

RLSMRLSE DETAIL TEST <result> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<componentName>	Optional	0 - 1	String (256), variable	Failing component name.
<componentType>	Optional	0 - 1	String (3)	Failing component type.
<componentUserid>	Optional	0 - 1	String (8), variable	Failing component user ID.
<originatingPackage>	Optional	0 - 1	String (10)	Originating package name.
<originatingUserid>	Optional	0 - 1	String (8), variable	Originating component user ID.
<package>	Optional	0 - 1	String (10)	Failing package name.
<packageStatus>	Optional	0 - 1	String (3)	Package status.
<reasonCode>	Optional	0 - 1	String (1)	Failure reason code.
<reasonForFailure>	Optional	0 - 1	String (20)	Reason for failure.
<recordType>	Optional	0 - 1	String (1)	Component record type.
<release>	Optional	0 - 1	String (8), variable	Release name.
<releaseArea>	Optional	0 - 1	String (8), variable	Release area name.
<sourceComponentName>	Optional	0 - 1	String (256), variable	Failing source component name.
<sourceComponentType>	Optional	0 - 1	String (3)	Failing source component type.

RLSMRLSE HST LIST

The RLSMRLSE HST LIST message lists the history residing in the ERO component history table for components in a release. A comprehensive list may be obtained for all components in a release, or a selective list may be obtained by requesting specific fields.

The XML service/scope/message tags and attributes for this message are:

```
<service name="RLSMRLSE">
<scope name="HST">
<message name="LIST">
```

These tags appear in both requests and replies.

RLSMRLSE HST LIST — Request

The following example shows how you might code a request to list all of the component history for a release. Data structure details for the `<request>` tag are identical to those for the RLSMAREA HST LIST message except that the `<release>` subtag is optional - see "Example XML — RLSMAREA HST LIST Request" on page 65.

Example XML — RLSMRLSE HST LIST Request

```
<?xml version="1.0"?>
<service name="RLSMRLSE">
 <scope name="HST">
  <message name="LIST">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
    <release>S4712COM</release>
   </request>
  </message>
 </scope>
</service>
```

RLSMRLSE HST LIST — Reply

The XML reply to a RLSMRLSE HST LIST request returns zero to many `<result>` data elements. Each result lists a record of history data for a component.

The standard `<response>` data element follows any `<result>` tags in the reply and indicates the success or failure of the request. Successful requests have a return code of 00. Unsuccessful requests have a return code of 04 or higher. Because it is the final data element in the XML reply message, the `<response>` tag serves as an end-of-list marker.

The following example shows what the reply message might look like. Data structure details for the `<result>` tag are identical to those for the RLSMAREA HST LIST message - see "RLSMAREA HST LIST <result> Data Structure" on page 67.

Example XML — RLSMRLSE HST LIST Reply

```
<?xml version="1.0"?>
<service name="RLSMRLSE">
 <scope name="HST">
  <message name="LIST">
   <result>
    <release>S4712COM</release>
    <releaseArea>ACCTPAY</releaseArea>
    <package>ACTP000076</package>
    <applName>ACTP</applName>
    <packageId>000076</packageId>
    <componentHistoryType>RLS</componentHistoryType>
    <componentType>CPY</componentType>
    <component>ACPCPYCE</component>
    <updater>KCAMPBE</updater>
    <componentChangeDate>20120827</componentChangeDate>
    <componentChangeTime>135951</componentChangeTime>
    <componentPriorChangeDate>20120827</componentPriorChangeDate>
    <componentPriorChangeTime>135835</componentPriorChangeTime>
    <setssi>630B28C0</setssi>
    <version>01</version>
    <modLevel>02</modLevel>
    <componentHistoryStatus>CKI</componentHistoryStatus>
    <componentPriorHistoryStatus>RTV</componentPriorHistoryStatus>
    <useDb2PreCompileOption>N</useDb2PreCompileOption>
    <forceAssignedBuildProc>N</forceAssignedBuildProc>
    <size>00000003</size>
    <checkinReleaseArea>PACKAGE</checkinReleaseArea>
    <checkinUser>KCAMPBE</checkinUser>
    <releaseAreaCheckinDate>20120827</releaseAreaCheckinDate>
    <releaseAreaCheckinTime>135951</releaseAreaCheckinTime>
    <retrieveReleaseArea>ACCTPAY</retrieveReleaseArea>
    <retrieveUser>KCAMPBE</retrieveUser>
    <releaseAreaRetrieveDate>20120827</releaseAreaRetrieveDate>
    <releaseAreaRetrieveTime>135835</releaseAreaRetrieveTime>
    <promotionDate>20121231</promotionDate>
    <promotionTime>000000</promotionTime>
    <componentCheckoutSetssi>630B28C0</componentCheckoutSetssi>
    <componentCheckoutDate>20120827</componentCheckoutDate>
    <componentCheckoutTime>135951</componentCheckoutTime>
   </result>
   <response>
    <statusMessage>CMR8700I - Release HST Table service completed
     </statusMessage>
    <statusReturnCode>00</statusReturnCode>
    <statusReasonCode>8700</statusReasonCode>
   </response>
  </message>
 </scope>
</service>
```

RLSMRLSE IMP LIST

The RLSMRLSE IMP LIST message lists the impact data stored in the ERO Db2 impact table.

The XML service/scope/message tags and attributes for this message are:

```
<service name="RLSMRLSE">
<scope name="IMP">
<message name="LIST">
```

These tags appear in both requests and replies.

RLSMRLSE IMP LIST — Request

There are no required subtags for this request; however, the subtags allow you to request specific releases and component relationships.

The 'source' tags refer to the top level of the relationship and the 'target' tags refer to the bottom level. For example, in a source-to-copybook relationship, the 'source' tags pertain to where the source component was generated; the 'target' fields pertain to where the copybook was pulled from. Similarly, for a composite load-to-subroutine relationship, the 'source' is for the composite and the 'target' is for the subroutine. The `<toWhat>` tag value is the name of the subcomponent (bottom level component, i.e. copybook or subroutine).

The following example shows how you might code a request to list the impact data for all releases. Data structure details for the `<request>` tag are identical to those for the RLSMAREA IMP LIST message except that the `<sourceRelease>` subtag is optional - see "RLSMAREA IMP LIST <request> Data Structure" on page 73.

Example XML — RLSMRLSE IMP LIST Request

```
<?xml version="1.0"?>
<service name="RLSMRLSE">
 <scope name="IMP">
  <message name="LIST">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
   </request>
  </message>
 </scope>
</service>
```

RLSMRLSE IMP LIST — Reply

The XML reply to a RLSMRLSE IMP LIST request returns zero to many `<result>` data elements. Each result lists impact data for a component relationship.

The standard `<response>` data element follows any `<result>` tags in the reply and indicates the success or failure of the request. Successful requests have a return code of 00. Unsuccessful requests have a return code of 04 or higher. Because it is the final data element in the XML reply message, the `<response>` tag serves as an end-of-list marker.

The following example shows what the reply message might look like. Data structure details for the `<result>` tag are identical to those for the RLSMAREA IMP LIST message - see "RLSMAREA IMP LIST <result> Data Structure" on page 74.

Example XML — RLSMRLSE IMP LIST Reply

```
<?xml version="1.0"?>
<service name="RLSMRLSE">
 <scope name="IMP">
  <message name="LIST">
   <result>
    <sourceRelease>S4712COM</sourceRelease>
    <sourceReleaseArea>ACCTPAY</sourceReleaseArea>
    <sourceApplName>COMM</sourceApplName>
    <sourceLibraryType>SRS</sourceLibraryType>
    <sourceLibraryBun>0000000032</sourceLibraryBun>
    <sourceComponent>COMSRS10</sourceComponent>
    <relation>CPY</relation>
    <targetApplName>COMM</targetApplName>
    <targetLibraryType>CPY</targetLibraryType>
    <targetLibraryBun>0000000028</targetLibraryBun>
    <toWhat>COMCPY10</toWhat>
    <hashToken>8C2DCBB800000080</hashToken>
   </result>
   <result>
    <sourceRelease>S4712COM</sourceRelease>
    <sourceReleaseArea>ACCTPAY</sourceReleaseArea>
    <sourceApplName>ACTR</sourceApplName>
    <sourceLibraryType>SCS</sourceLibraryType>
    <sourceLibraryBun>0000000024</sourceLibraryBun>
    <sourceComponent>ACRSCSCE</sourceComponent>
    <relation>CPY</relation>
    <targetRelease>S4712COM</targetRelease>
    <targetReleaseArea>ACCTPAY</targetReleaseArea>
    <targetApplName>ACTR</targetApplName>
    <targetLibraryType>CPY</targetLibraryType>
    <targetLibraryBun>0000000017</targetLibraryBun>
    <toWhat>ACRCPYCE</toWhat>
    <hashToken>8FA76F0000000079</hashToken>
   </result>
   .
   .
   .
   <response>
    <statusMessage>CMR8700I - Release IMP Table service completed
     </statusMessage>
    <statusReturnCode>00</statusReturnCode>
    <statusReasonCode>8700</statusReasonCode>
   </response>
  </message>
 </scope>
</service>
```

RLSMRLSE LIBRARY LIST

The RLSMRLSE LIBRARY LIST message lists allocated release area libraries.

The XML service/scope/message tags and attributes for this message are:

```
<service name="RLSMRLSE">
<scope name="LIBRARY">
```

```
<message name="LIST">
```

These tags appear in both requests and replies.

RLSMRLSE LIBRARY LIST — Request

The following example shows how you might code a request to list the allocated release area libraries for a release. Data structure details for the <request> tag follow the example.

Example XML — RLSMRLSE LIBRARY LIST Request

```
<?xml version="1.0"?>
<service name="RLSMRLSE">
 <scope name="LIBRARY">
  <message name="LIST">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
    <release>S4712COM</release>
   </request>
  </message>
 </scope>
</service>
```

RLSMRLSE LIBRARY LIST <request> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<applName>	Optional	0 - 1	String (4), variable	Release application name. *NOTE:* May be masked using asterisk (*) wildcard.
<libType>	Optional	0 - 1	String (3)	Release library type. *NOTE:* May be masked using asterisk (*) wildcard.
<release>	Required	1	String (8), variable	Release name.
<releaseArea>	Optional	0 - 1	String (8), variable	Release area name. *NOTE:* May be masked using asterisk (*) wildcard.

RLSMRLSE LIBRARY LIST — Reply

The XML reply to a RLSMRLSE LIBRARY LIST request returns zero to many <result> data elements. Each result lists information for an allocated release area library.

The standard <response> data element follows any <result> tags in the reply and indicates the success or failure of the request. Successful requests have a return code of

00. Unsuccessful requests have a return code of 04 or higher. Because it is the final data element in the XML reply message, the <response> tag serves as an end-of-list marker.

The following example shows what the reply message might look like. Data structure details for the <result> tag follow the example.

Example XML — RLSMRLSE LIBRARY LIST Reply

```
<?xml version="1.0"?>
<service name="RLSMRLSE">
 <scope name="LIBRARY">
  <message name="LIST">
   <result>
    <release>S4712COM</release>
    <releaseArea>ACCTPAY</releaseArea>
    <applName>ACTP</applName>
    <libType>CPY</libType>
    <likeType>CPD</likeType>
    <libraryOrg>PDS</libraryOrg>
    <areaLibrary>CMNTP.S4712COM.ACCTPAY.ACTP.CPY</areaLibrary>
   </result>
   <result>
    <release>S4712COM</release>
    <releaseArea>FINANCE</releaseArea>
    <applName>ACTP</applName>
    <libType>CPY</libType>
    <likeType>CPD</likeType>
    <libraryOrg>PDS</libraryOrg>
    <areaLibrary>CMNTP.S4712COM.FINANCE.ACTP.CPY</areaLibrary>
   </result>
   <result>
    <release>S4712COM</release>
    <releaseArea>GENLEDGR</releaseArea>
    <applName>ACTP</applName>
    <libType>CPY</libType>
    <likeType>CPD</likeType>
    <libraryOrg>PDS</libraryOrg>
    <areaLibrary>CMNTP.S4712COM.GENLEDGR.ACTP.CPY</areaLibrary>
   </result>
   .
   .
   .
   <response>
    <statusMessage>CMR8700I - Release Library service completed
     </statusMessage>
    <statusReturnCode>00</statusReturnCode>
    <statusReasonCode>8700</statusReasonCode>
   </response>
  </message>
 </scope>
</service>
```

RLSMRLSE LIBRARY LIST <result> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<applName>	Optional	0 - 1	String (4), variable	Release application name.
<areaLibrary>	Optional	0 - 1	String (1024), variable	Release area library dataset name.
<libType>	Optional	0 - 1	String (3)	Release library type.
<libraryOrg>	Optional	0 - 1	String (8), variable	Release library dataset organization. Values: **LIB** = Librarian **OTH** = Other **PAN** = Panvalet **PDS** = Partitioned dataset **SEQ** = Sequential **VSM** = VSAM **MIG** = Migrate **PDS**E = Partitioned dataset extended
<likeType>	Optional	0 - 1	String (3)	Release library "like" type. Values: C - Copy L - Load blank - Other P - PDS S - Source N - NCAL O - Obj K - LCT X - List J - JCL/Proc
<release>	Optional	0 - 1	String (8), variable	Release name.
<releaseArea>	Optional	0 - 1	String (8), variable	Release area name.

RLSMRLSE PRIOR LIST

The RLSMRLSE PRIOR LIST message lists prior release information.

The XML service/scope/message tags and attributes for this message are:

```
<service name="RLSMRLSE">
<scope name="PRIOR">
<message name="LIST">
```

These tags appear in both requests and replies.

RLSMRLSE PRIOR LIST — Request

The following example shows how you might code a request to list the prior release information for a release. Data structure details for the <request> tag follow the example.

Example XML — RLSMRLSE PRIOR LIST Request

```
<?xml version="1.0"?>
<service name="RLSMRLSE">
 <scope name="PRIOR">
  <message name="LIST">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
    <release>S4712COM</release>
   </request>
  </message>
 </scope>
</service>
```

RLSMRLSE PRIOR LIST <request> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<priorRelease>	Optional	0 - 1	String (8), variable	Prior release name.
<priorReleaseApplName>	Optional	0 - 1	String (4), variable	Prior release application name.
<release>	Required	1	String (8), variable	Release name.

RLSMRLSE PRIOR LIST — Reply

The XML reply to a RLSMRLSE PRIOR LIST request returns zero to many <result> data elements. Each result lists information for a prior release.

The standard <response> data element follows any <result> tags in the reply and indicates the success or failure of the request. Successful requests have a return code of 00. Unsuccessful requests have a return code of 04 or higher. Because it is the final data element in the XML reply message, the <response> tag serves as an end-of-list marker.

The following example shows what the reply message might look like. Data structure details for the <result> tag follow the example.

Example XML — RLSMRLSE PRIOR LIST Reply

```
<?xml version="1.0"?>
<service name="RLSMRLSE">
 <scope name="PRIOR">
```

```
<message name="LIST">
 <result>
  <release>S4712COM</release>
  <priorRelease>S4712010</priorRelease>
  <allPriorReleaseAppls>Y</allPriorReleaseAppls>
  <allPriorReleaseLibTypes>Y</allPriorReleaseLibTypes>
  <includeSystemPriorAreas>N</includeSystemPriorAreas>
  <includeSinglePathPriorAreas>N</includeSinglePathPriorAreas>
  <includeAllPriorAreas>N</includeAllPriorAreas>
  <priorReleaseSyslibAscending>Y</priorReleaseSyslibAscending>
  <priorReleaseInstallDate>20121005</priorReleaseInstallDate>
  <priorReleaseLastArea>FINANCE</priorReleaseLastArea>
  <priorReleaseHighLevelNode>CMNTP</priorReleaseHighLevelNode>
  <priorReleaseLibPattern>HRAPL</priorReleaseLibPattern>
  <priorReleaseHighLevelPath>/cmntp</priorReleaseHighLevelPath>
  <allPriorReleaseLibTypes>Y</allPriorReleaseLibTypes>
  <libTypeCount>00000</libTypeCount>
 </result>
 <response>
  <statusMessage>CMR8700I - Prior Release service completed
   </statusMessage>
  <statusReturnCode>00</statusReturnCode>
  <statusReasonCode>8700</statusReasonCode>
 </response>
</message>
</scope>
</service>
```

RLSMRLSE PRIOR LIST <result> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<allPriorReleaseAppls>	Optional	0 - 1	String (1)	**Y** = Include all prior release applications. **N** = Include only the specified application.
<allPriorReleaseLibTypes>	Optional	0 - 1	String (1)	**Y** = Include all library types. **N** = Include only the defined library types.
<includeAllPriorAreas>	Optional	0 - 1	String (1)	**Y** = Include all release areas of the prior release. **N** = Include only the final area of the prior release.
<includeSinglePathPriorAreas>	Optional	0 - 1	String (1)	**Y** = Include all single path areas of the prior release. **N** = Include only the final single path area of the prior release.
<includeSystemPriorAreas>	Optional	0 - 1	String (1)	**Y** = Include all system areas of the prior release. **N** = Include only the final system area of the prior release.
<libTypeCount>	Optional	0 - 1	String (5)	Prior release related library type count.
<priorRelease>	Optional	0 - 1	String (8), variable	Prior release name.
<priorReleaseApplName>	Optional	0 - 1	String (4), variable	Prior release application name.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<priorReleaseHighLevelNode>	Optional	0 - 1	String (8), variable	Prior release high-level qualifier.
<priorReleaseHighLevelPath>	Optional	0 - 1	String (73), variable	Prior release high-level path name.
<priorReleaseInstallDate>	Optional	0 - 1	Date, yyyymmdd	Prior release install date.
<priorReleaseLastArea>	Optional	0 - 1	String (8), variable	Prior release final area name.
<priorReleaseLibPattern>	Optional	0 - 1	String (5), variable	Prior release dataset name pattern.
<priorReleaseLibType>	Optional	0 - 1	String (3)	Prior release related library type.
<priorReleaseSyslibAscending>	Optional	0 - 1	String (1)	**Y** = SYSLIB build order ascending. **N** = SYSLIB build order descending.
<release>	Optional	0 - 1	String (8), variable	Release name.

RLSMRLSE REASONS LIST

The RLSMRLSE REASONS LIST message lists backout and revert reasons for a release.

The XML service/scope/message tags and attributes for this message are:

```
<service name="RLSMRLSE">
<scope name="REASONS">
<message name="LIST">
```

These tags appear in both requests and replies.

RLSMRLSE REASONS LIST — Request

The following example shows how you might code a request to list backout and revert reasons for a release. Data structure details for the `<request>` tag follow the example.

Example XML — RLSMRLSE REASONS LIST Request

```
<?xml version="1.0"?>
<service name="RLSMRLSE">
 <scope name="REASONS">
  <message name="LIST">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
    <release>S4712COM</release>
   </request>
  </message>
```

```
 </scope>
</service>
```

RLSMRLSE REASONS LIST <request> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<fromDate>	Optional	0 - 1	Date, yyyymmdd	Start date in desired range of reason update dates.
<reasonType>	Optional	0 - 1	String (1)	Reason type. Values: **B** = Backout **R** = Revert
<release>	Required	1	String (8), variable	Release name.
<siteName>	Optional	0 - 1	String (8), variable	Site name.
<toDate>	Optional	0 - 1	Date, yyyymmdd	End date in desired range of reason update dates.
<updater>	Optional	0 - 1	String (8), variable	TSO user ID of last user to back out or revert the release.

RLSMRLSE REASONS LIST — Reply

The XML reply to a RLSMRLSE REASONS LIST request returns zero to many `<result>` data elements. Each result lists backout/revert reasons for the release.

The standard `<response>` data element follows any `<result>` tags in the reply and indicates the success or failure of the request. Successful requests have a return code of 00. Unsuccessful requests have a return code of 04 or higher. Because it is the final data element in the XML reply message, the `<response>` tag serves as an end-of-list marker.

The following example shows what the reply message might look like. Data structure details for the `<result>` tag follow the example.

Example XML — RLSMRLSE REASONS LIST Reply

```
<?xml version="1.0"?>
<service name="RLSMRLSE">
 <scope name="REASONS">
  <message name="LIST">
   <result>
    <release>S4712COM</release>
    <reasonType>B</reasonType>
    <siteName>SERT4</siteName>
    <updater>KCAMPBE</updater>
    <updateDate>20120827</updateDate>
    <updateTime>111831</updateTime>
    <reasons>BACKOUT RELEASE TO CHANGE COMCPY00 TO COMCPY10</reasons>
    <reason01>BACKOUT RELEASE TO CHANGE COMCPY00 TO COMCPY10</reason01>
   </result>
   <result>
    <release>S4712COM</release>
    <reasonType>R</reasonType>
```

```
    <siteName>SERT4</siteName>
    <updater>KCAMPBE</updater>
    <updateDate>20120827</updateDate>
    <updateTime>121011</updateTime>
    <reasons>REVERT TO CHANGE COMCPY00 TO COMCPY10</reasons>
    <reason01>REVERT TO CHANGE COMCPY00 TO COMCPY10</reason01>
   </result>
   .
   .
   .
   <response>
    <statusMessage>CMR8700I - LIST Reasons service completed
     </statusMessage>
    <statusReturnCode>00</statusReturnCode>
    <statusReasonCode>8700</statusReasonCode>
   </response>
  </message>
 </scope>
</service>
```

RLSMRLSE REASONS LIST <result> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<reason01>	Optional	0 - 1	String (72), variable	Reason line 1.
<reason02>	Optional	0 - 1	String (72), variable	Reason line 2.
<reason03>	Optional	0 - 1	String (72), variable	Reason line 3.
<reason04>	Optional	0 - 1	String (72), variable	Reason line 4.
<reason05>	Optional	0 - 1	String (72), variable	Reason line 5.
<reason06>	Optional	0 - 1	String (72), variable	Reason line 6.
<reason07>	Optional	0 - 1	String (72), variable	Reason line 7.
<reason08>	Optional	0 - 1	String (72), variable	Reason line 8.
<reason09>	Optional	0 - 1	String (72), variable	Reason line 9.
<reasonType>	Optional	0 - 1	String (1)	Reason type. Values: **B** = Backout **R** = Revert
<reasons>	Optional	0 - 9	String (72), variable	Reason lines 1 - 9. *NOTE:* The <reasons> tag is deprecated and contains the same information as <reason01> -- <reason09>.
<release>	Optional	0 - 1	String (8), variable	Release name.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<siteName>	Optional	0 - 1	String (8), variable	Site name.
<updateDate>	Optional	0 - 1	Date, yyyymmdd	Reason update date.
<updateTime>	Optional	0 - 1	Time, hhmmss	Reason update time.
<updater>	Optional	0 - 1	String (8), variable	TSO user ID of user who last updated the reason.

RLSMRLSE RLS_LINK LIST

The RLSMRLSE RLS_LINK LIST message lists release management data across a TCP/IP link.

The XML service/scope/message tags and attributes for this message are:

```
<service name="RLSMRLSE">
<scope name="RLS_LINK">
<message name="LIST">
```

These tags appear in both requests and replies.

RLSMRLSE RLS_LINK LIST — Request

The following example shows how you might code a request to list release management data across a TCP/IP link. Data structure details for the `<request>` tag follow the example.

Example XML — RLSMRLSE RLS_LINK LIST Request

```
<?xml version="1.0"?>
<service name="RLSMRLSE">
 <scope name="RLS_LINK">
  <message name="LIST">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
    <release>S4712COM</release>
   </request>
  </message>
 </scope>
</service>
```

RLSMRLSE RLS_LINK LIST <request> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<linkControlItem>	Optional	0 - 1	String (255), variable	Link control item.
<linkRequestor	Optional	0 - 1	String (20), variable	Link requestor.
<release>	Required	1	String (8), variable	Release name.
<sourceLinkIpAddress>	Optional	0 - 1	String (255), variable	Source link IP address.
<sourceLinkPortid>	Optional	0 - 1	String (8), variable	Source link port ID.

RLSMRLSE RLS_LINK LIST — Reply

The XML reply to a RLSMRLSE RLS_LINK LIST request returns one `<result>` data element, which lists information for a linked release.

The standard `<response>` data element follows any `<result>` tags in the reply and indicates the success or failure of the request. Successful requests have a return code of 00. Unsuccessful requests have a return code of 04 or higher. Because it is the final data element in the XML reply message, the `<response>` tag serves as an end-of-list marker.

The following example shows what the reply message might look like. Data structure details for the `<result>` tag follow the example.

Example XML — RLSMRLSE RLS_LINK LIST Reply

```
<?xml version="1.0"?>
<service name="RLSMRLSE">
 <scope name="RLS_LINK">
  <message name="LIST">
   <result>
    <release>S4712COM</release>
    <releaseDesc>Financial Accounting Release S4.V712 #COM</releaseDesc>
    <releaseStatus>BAS</releaseStatus>
    <installFromDate>20120824</installFromDate>
    <installToDate>20121231</installToDate>
   </result>
   <response>
    <statusMessage>CMR8764I - Release S4712COM is not Linked.
     </statusMessage>
    <statusReturnCode>00</statusReturnCode>
    <statusReasonCode>8764</statusReasonCode>
   </response>
  </message>
 </scope>
</service>
```

RLSMRLSE RLS_LINK LIST <result> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<approvePermitRequired>	Optional	0 - 1	String (1)	**Y** = Approve permit is required. **N** = Approve permit is not required.
<backoutPermitRequired>	Optional	0 - 1	String (1)	**Y** = Backout permit is required. **N** = Backout permit is not required.
<blockPermitRequired>	Optional	0 - 1	String (1)	**Y** = Block permit is required. **N** = Block permit is not required.
<deletePermitRequired>	Optional	0 - 1	String (1)	**Y** = Delete permit is required. **N** = Delete permit is not required.
<demotePermitRequired>	Optional	0 - 1	String (1)	**Y** = Demote permit is required. **N** = Demote permit is not required.
<installFromDate>	Optional	0 - 1	Date, yyyymmdd	Install start date.
<installToDate>	Optional	0 - 1	Date, yyyymmdd	Install end date.
<linkControlEventListenerCode Page>	Optional	0 - 1	String (4), variable	Link control event listener code page.
<linkControlItem>	Optional	0 - 1	String (255), variable	Link control item.
<linkDate>	Optional	0 - 1	Date, yyyymmdd	Link date.
<linkRequestor	Optional	0 - 1	String (20), variable	Link requestor.
<linkTime>	Optional	0 - 1	Time, hhmmss	Link time.
<promotePermitRequired>	Optional	0 - 1	String (1)	**Y** = Promote permit is required. **N** = Promote permit is not required.
<rejectPermitRequired>	Optional	0 - 1	String (1)	**Y** = Reject permit is required. **N** = Reject permit is not required.
<release>	Optional	0 - 1	String (8), variable	Release name.
<releaseDesc>	Optional	0 - 1	String (72), variable	Release description.
<releaseStatus>	Optional	0 - 1	String (3)	Release status. Values: **APR** = Approved **BAK** = Backed out **BAS** = Baselined **BLK** = Blocked **DEL** = Deleted **DEV** = In development **DIS** = Distributed **INS** = Installed **REJ** = Rejected
<revertPermitRequired>	Optional	0 - 1	String (1)	**Y** = Revert permit is required. **N** = Revert permit is not required.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<sourceLinkIpAddress>	Required	1	String (255), variable	Source link IP address.
<sourceLinkPortid>	Required	1	String (8), variable	Source link port ID.
<unblockPermitRequired>	Optional	0 - 1	String (1)	**Y** = Unblock permit is required. **N** = Unblock permit is not required.
<undeletePermitRequired>	Optional	0 - 1	String (1)	**Y** = Undelete permit is required. **N** = Undelete permit is not required.

RLSMRLSE SERVICE LIST

The RLSMRLSE SERVICE LIST message lists scheduler dates, times, and status for a named release.

The XML service/scope/message tags and attributes for this message are:

```
<service name="RLSMRLSE">
<scope name="SERVICE">
<message name="LIST">
```

These tags appear in both requests and replies.

RLSMRLSE SERVICE LIST — Request

The following example shows how you might code a request to list the scheduler dates, times, and status for a named release. Data structure details for the `<request>` tag follow the example.

Example XML — RLSMRLSE SERVICE LIST Request

```
<?xml version="1.0"?>
<service name="RLSMRLSE">
 <scope name="SERVICE">
  <message name="LIST">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
    <releaseParms> </releaseParms>
    <release>S4712010</release>
   </request>
  </message>
 </scope>
</service>
```

RLSMRLSE SERVICE LIST <request> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<release>	Required	1	String (8), variable	Release name.
<releaseParms>	Required	1	String (1)	Release parameters. Values: **A** = ADMIN list. List all releases regardless of the release status. **blank** = List only those releases that have fully configured release areas.

RLSMRLSE SERVICE LIST — Reply

The XML reply to a RLSMRLSE SERVICE LIST request returns one `<result>` data element, which lists status flags and other information for the release.

The standard `<response>` data element follows any `<result>` tags in the reply and indicates the success or failure of the request. Successful requests have a return code of 00. Unsuccessful requests have a return code of 04 or higher. Because it is the final data element in the XML reply message, the `<response>` tag serves as an end-of-list marker.

The following example shows what the reply message might look like. Data structure details for the `<result>` tag follow the example.

Example XML — RLSMRLSE SERVICE LIST Reply

```
<?xml version="1.0"?>
<service name="RLSMRLSE">
 <scope name="SERVICE">
  <message name="LIST">
   <result>
    <release>S4712010</release>
    <releaseDesc>Financial Accounting Release S4.V712 #010</releaseDesc>
    <releaseStatus>DEV</releaseStatus>
    <releaseRequestorName>NATHAN CHOY</releaseRequestorName>
    <releaseRequestorPhone>808-555-1212</releaseRequestorPhone>
    <releaseRequestorDept>FINANCE</releaseRequestorDept>
    <releaseWorkChangeRequest>F5567792</releaseWorkChangeRequest>
    <releaseSchedulerCmn>Y</releaseSchedulerCmn>
    <releaseSchedulerManual>Y</releaseSchedulerManual>
    <releaseSchedulerOther>Y</releaseSchedulerOther>
    <releaseSchedulerType>MANUAL</releaseSchedulerType>
    <addReleaseInterfacingApprovers>Y</addReleaseInterfacingApprovers>
    <isReleaseAreasConfigured>Y</isReleaseAreasConfigured>
    <releaseSyslibOrder>A</releaseSyslibOrder>
    <auditEnforceIHASetting>N</auditEnforceIHASetting>
    <auditIgnoreHigherAreas>N</auditIgnoreHigherAreas>
    <isReleaseLinked>N</isReleaseLinked>
    <isAutoFixDevPackages>N</isAutoFixDevPackages>
    <isAutoFixFrzPackages>N</isAutoFixFrzPackages>
    <isReleaseInstallBuildJclPending>N</isReleaseInstallBuildJclPending>
    <isReleaseInstallPending>N</isReleaseInstallPending>
    <isReleaseRevertPending>N</isReleaseRevertPending>
    <isReleaseBackoutPending>N</isReleaseBackoutPending>
    <isReleaseApprovalPending>N</isReleaseApprovalPending>
```

```
 <isReleaseBlockPending>N</isReleaseBlockPending>
 <releaseProblemActionCode>1</releaseProblemActionCode>
 <releaseAuditMinRule>0</releaseAuditMinRule>
 <releaseCheckinMinRule>0</releaseCheckinMinRule>
 <releaseBlockingMinRule>0</releaseBlockingMinRule>
 <releaseRetrieveMinRule>0</releaseRetrieveMinRule>
 <releaseApproveMinRule>0</releaseApproveMinRule>
 <releaseFromInstallDate>20131005</releaseFromInstallDate>
 <releaseFromInstallTime>000100</releaseFromInstallTime>
 <releaseDateCreated>20120312</releaseDateCreated>
 <releaseTimeCreated>123024</releaseTimeCreated>
 <releaseCreateUserid>KCAMPBE</releaseCreateUserid>
 <releaseToInstallDate>20131231</releaseToInstallDate>
 <releaseToInstallTime>235959</releaseToInstallTime>
<releaseImplInst>All packages attached to this release will be installed
 on Saturday</releaseImplInst>
 <releaseImplInst>DEC 17, 2012. If there are problems with any part of
  this install</releaseImplInst>
 <releaseImplInst>call the Release Manager.</releaseImplInst>
 <releaseHighLevelName>CMNTP</releaseHighLevelName>
 <releaseDsnPattern>HRAPL</releaseDsnPattern>
 <releaseHighLevelPath>/cmntp</releaseHighLevelPath>
 <totalReleasePackages>00003</totalReleasePackages>
</result>
<response>
 <statusMessage>CMR8700I - Release Management service completed
  </statusMessage>
 <statusReturnCode>00</statusReturnCode>
 <statusReasonCode>8700</statusReasonCode>
</response>
</message>
</scope>
</service>
```

RLSMRLSE SERVICE LIST <result> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<addReleaseInterfacingApprovers>	Optional	0 - 1	String (1)	**Y** = Add related approvers. **N** = Do not add related approvers.
<auditEnforceIHASetting>	Optional	0 - 1	String (1)	**Y** = Enforce IHA setting. **N** = Do not enforce IHA settings.
<auditIgnoreHigherAreas>	Optional	0 - 1	String (1)	**Y** = Ignore higher areas. **N** = Do not ignore higher areas. **C** = If the current release has only a single path through it, process as if "Y" had been specified. If the current release has multiple paths through it, process as if "N" had been specified.
<isAutoFixAprPackages>	Optional	0 - 1	String (1)	**Y** = Automatically cleanup APR packages. **N** = Do not automatically cleanup APR packages.
<isAutoFixDevPackages>	Optional	0 - 1	String (1)	**Y** = Automatically cleanup DEV packages. **N** = Do not automatically cleanup DEV packages.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<isAutoFixFrzPackages>	Optional	0 - 1	String (1)	**Y** = Automatically cleanup FRZ packages. **N** = Do not automatically cleanup FRZ packages.
<isReleaseApprovalPending>	Optional	0 - 1	String (1)	**Y** = Release approval is pending. **N** = Release approval is not pending.
<isReleaseAreasConfigured>	Optional	0 - 1	String (1)	**Y** = Release areas are configured. **N** = Release areas are not configured.
<isReleaseBackoutPending>	Optional	0 - 1	String (1)	**Y** = Release backout is pending. **N** = Release backout is not pending.
<isReleaseBlockPending>	Optional	0 - 1	String (1)	**Y** = Release block is pending. **N** = Release block is not pending.
<isReleaseInstallBuildJclPending	Optional	0 - 1	String (1)	**Y** = Release install JCL builds are pending. **N** = Release install JCL builds are not pending.
<isReleaseInstallPending>	Optional	0 - 1	String (1)	**Y** = Release install is pending. **N** = Release install is not pending.
<isReleaseLinked>	Optional	0 - 1	String (1)	**Y** = Release is linked. **N** = Release is not linked.
<isReleaseRevertPending>	Optional	0 - 1	String (1)	**Y** = Release revert is pending. **N** = Release revert is not pending.
<release>	Optional	0 - 1	String (8), variable	Release name.
<releaseApproveMinRule>	Optional	0 - 1	Integer (1)	Release minimum approval rule. Values: **0** = No approvals required. **1** = Approvals required before checkin. **2** = Approvals required before advancing to the next area (check-off). **3** = Rule 1 and rule 2.
<releaseAuditMinRule>	Optional	0 - 1	Integer (1)	Release area minimum audit rule. Values: **0** = Audit optional. **1** = Audit required. RC < 20 (audit failure). **2** = Audit required. RC <= 12 (out-of-sync errors within audited area). **3** = Audit required. RC <= 8 (out-of-sync errors with respect to next areas/final areas in prior releases/baseline). **4** = Audit required. RC <= 4 (no out-of-sync errors but some duplicates exist). **5** = Audit required. RC = 0 (no out-of-sync errors and no warnings).
<releaseAuditReturnCode>	Optional	0 - 1	String (2)	Release audit return code.
<releaseBackoutUserid>	Optional	0 - 1	String (8), variable	User ID of user who backed out release.
<releaseBlockUserid>	Optional	0 - 1	String (8), variable	User ID of user who blocked release.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<releaseBlockingMinRule>	Optional	0 - 1	Integer (1)	Release minimum blocking rule. Values: **0** = Wide open. **1** = Audit required. **2** = User must pass entity check (and rule 0). **3** = Rule 1 and rule 2.
<releaseBuildNumber>	Optional	0 - 1	String (10), variable	Release build number.
<releaseCheckinMinRule>	Optional	0 - 1	Integer (1)	Release minimum checkin rule. Values: **0** = Wide open. **1** = Area or package audit required before moving to next area. **2** = Area must be blocked (or package frozen) before moving to next area. **3** = User must pass entity check (and rule 0). **4** = Rule 1 and rule 2. **5** = Rule 1 and rule 3. **6** = Rule 2 and rule 3. **7** = Rule 1, rule2, and rule 3.
<releaseCreateUserid>	Optional	0 - 1	String (8), variable	User ID of user who created release.
<releaseDateApproved>	Optional	0 - 1	Date, yyyymmdd	Date that release was approved.
<releaseDateBackedOut>	Optional	0 - 1	Date, yyyymmdd	Date that release was backed out.
<releaseDateBaselined>	Optional	0 - 1	Date, yyyymmdd	Date that release was baselined.
<releaseDateBlocked>	Optional	0 - 1	Date, yyyymmdd	Date that release was blocked.
<releaseDateCreated>	Optional	0 - 1	Date, yyyymmdd	Date that release was created.
<releaseDateDisReceived>	Optional	0 - 1	Date, yyyymmdd	Date that DEV site received all notifications that release was successfully distributed to all remote sites.
<releaseDateDistributed>	Optional	0 - 1	Date, yyyymmdd	Date that release was distributed.
<releaseDateInstalled>	Optional	0 - 1	Date, yyyymmdd	Date that release was installed.
<releaseDateMemoDeleted>	Optional	0 - 1	Date, yyyymmdd	Date that release was memo-deleted.
<releaseDateRejected>	Optional	0 - 1	Date, yyyymmdd	Date that release was rejected.
<releaseDateReverted>	Optional	0 - 1	Date, yyyymmdd	Date that release was reverted.
<releaseDesc>	Optional	0 - 1	String (72), variable	Release description.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<releaseDsnPattern>	Optional	0 - 1	String (5), variable	Release dataset name pattern.
<releaseFromInstallDate>	Optional	0 - 1	Date, yyyymmdd	Install start date.
<releaseFromInstallTime>	Optional	0 - 1	Time, hhmmss	Install start time.
<releaseHighLevelName>	Optional	0 - 1	String (8), variable	Release high-level qualifier.
<releaseHighLevelPath>	Optional	0 - 1	String (1024), variable	Release high-level HFS path.
<releaseImplInst>	Optional	0 - 1	String (72), variable	Release implementation instructions.
<releaseLibsAged>	Optional	0 - 1	String (1)	**Y** = Release libraries are aged. **N** = Release libraries are not aged.
<releaseMemoDeleteUserid>	Optional	0 - 1	String (8), variable	User ID of user who memo-deleted release.
<releaseOtherProblemAction>	Optional	0 - 1	String (72)	Release other problem description.
<releaseProblemActionCode>	Optional	0 - 1	String (1)	Action to be taken if a problem occurs when a package is installed. Values: **1** = Hold production and contact analyst. **2** = Back out change and continue production. **3** = Other
<releaseRejectUserid>	Optional	0 - 1	String (8), variable	User ID of user who rejected release.
<releaseRequestorDept>	Optional	0 - 1	String (8), variable	Release requester department.
<releaseRequestorName>	Optional	0 - 1	String (25), variable	Release requester name.
<releaseRequestorPhone>	Optional	0 - 1	String (15), variable	Release requester phone number.
<releaseRetrieveMinRule>	Optional	0 - 1	Integer (1)	Release minimum retrieve rule. Values: **0** = Retrieve allowed from any area (blocked or unblocked). **1** = Retrieve allowed only from unblocked areas. **2** = Must pass entity check (and rule 0). **3** = Rule 1 and rule 2.
<releaseRevertUserid>	Optional	0 - 1	String (8), variable	User ID of user who reverted release.
<releaseSchedulerCmn>	Optional	0 - 1	String (1)	**Y** = Allow CMN internal release scheduler. **N** = Do not allow CMN internal release scheduler.
<releaseSchedulerManual>	Optional	0 - 1	String (1)	**Y** = Allow manual release scheduler. **N** = Do not allow manual release scheduler.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<releaseSchedulerOther>	Optional	0 - 1	String (1)	**Y** = Allow other release scheduler. **N** = Do not allow other release scheduler.
<releaseReleaseSchedulerType>	Optional	0 - 1	String (8), variable	Release scheduler type. Values: **CMN** = ChangeMan ZMF. The installation jobs are submitted by the ZMF started task at the scheduled install date and time. **MANUAL** = Manual. The installation jobs are submitted as soon as the package approvals are completed. **OTHER** = Other. The installation jobs are inserted into a third-party scheduler via a batch job.
<releaseStatus>	Optional	0 - 1	String (3), variable	Release status. Values: **APR** = Approved **BAK** = Backed out **BAS** = Baselined **BLK** = Blocked **DEL** = Deleted **DEV** = In development **DIS** = Distributed **INS** = Installed **REJ** = Rejected
<releaseSyslibOrder>	Optional	0 - 1	String (1)	Release SYSLIB sort order. Values: **A** = Ascending **D** = Descending
<releaseTimeApproved>	Optional	0 - 1	Time, hhmmss	Time that release was approved.
<releaseTimeBackedOut>	Optional	0 - 1	Time, hhmmss	Time that release was backed out.
<releaseTimeBaselined>	Optional	0 - 1	Time, hhmmss	Time that release was baselined.
<releaseTimeBlocked>	Optional	0 - 1	Time, hhmmss	Time that release was blocked.
<releaseTimeCreated>	Optional	0 - 1	Time, hhmmss	Time that release was created.
<releaseTimeDisReceived>	Optional	0 - 1	Time, hhmmss	Time that DEV site received all notifications that release was successfully distributed to all remote sites.
<releaseTimeDistributed>	Optional	0 - 1	Time, hhmmss	Time that release was distributed.
<releaseTimeInstalled>	Optional	0 - 1	Time, hhmmss	Time that release was installed.
<releaseTimeMemoDeleted>	Optional	0 - 1	Time, hhmmss	Time that release was memo-deleted.
<releaseTimeRejected>	Optional	0 - 1	Time, hhmmss	Time that release was rejected.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<releaseTimeReverted>	Optional	0 - 1	Time, hhmmss	Time that release was reverted.
<releaseToInstallDate>	Optional	0 - 1	Date, yyyymmdd	Install end date.
<releaseToInstallTime>	Optional	0 - 1	Time, hhmmss	Install end time.
<releaseWorkChangeRequest>	Optional	0 - 1	String (16), variable	Release work change request.
<totalReleasePackages>	Optional	0 - 1	Integer (5)	Total number of packages in release.

RLSMRLSE SERVICE SEARCH

The RLSMRLSE SERVICE SEARCH message provides the means to do a comprehensive search for releases and list status flags and other information for selected releases.

The XML service/scope/message tags and attributes for this message are:

```
<service name="RLSMRLSE">
<scope name="SERVICE">
<message name="SEARCH">
```

These tags appear in both requests and replies.

RLSMRLSE SERVICE SEARCH — Request

The only required tag for this service is `<releaseCreators>`, which requires the entry of one or more user IDs.

This service provides several yes/no flags for release status filtering, for example, `<searchForApprovedStatus>`, `<searchForBackedOutStatus>`, and so on. These flags take default values as a group. The default changes based on whether or not you enter explicit values in these tags, as follows:

- If ***no*** status flag has an explicitly typed value, the default for all tags is "Y".
- If ***any*** status flag has an explicitly typed value, the default for the remaining tags is "N".

For more information about yes/no flags, see "Yes/No Flag Tags" on page 12.

The following tags allow multiple values to be entered in a semicolon-delimited list:

- `<releaseCreators>`
- `<releaseList>`
- `<searchForRlseApplNames>`
- `<searchForRlseApprovalEntities>`
- `<searchForRlseSites>`

The following example shows how you might code a request to list release information for a named release-creator. Data structure details for the <request> tag follow the example.

Example XML — RLSMRLSE SERVICE SEARCH Request

```
<?xml version="1.0"?>
<service name="RLSMRLSE">
 <scope name="SERVICE">
  <message name="SEARCH">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
    <releaseCreators>WSER99  </releaseCreators>
   </request>
  </message>
 </scope>
</service>
```

RLSMRLSE SERVICE SEARCH <request> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<releaseAuditReturnCode>	Optional	0 - 1	String (2)	Release audit return code.
<releaseCreators>	Optional	1	String (255), variable	User IDs of release creators. Multiple entries are allowed, delimited by semicolons. Asterisk (*) wildcard may be used.
<releaseDesc>	Optional	0 - 1	String (72), variable	Release description.
<releaseList>	Optional	0 - 1	String (255), variable	Release name list. Multiple release names are allowed, delimited by semicolons. Asterisk (*) wildcard may be used.
<releaseRequestorDept>	Optional	0 - 1	String (8), variable	Release requester department.
<releaseRequestorName>	Optional	0 - 1	String (25), variable	Release requester name.
<releaseRequestorPhone>	Optional	0 - 1	String (15), variable	Release requester phone number.
<releaseSearchParms>	Optional	0 - 1	String (1)	Release parameters. Values: **A** = ADMIN list. List all releases regardless of the release status. **blank** = List only those releases that have fully configured release areas.
<releaseWorkChangeRequest>	Optional	0 - 1	String (16), variable	Release work change request. Asterisk (*) wildcard may be used.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<searchForApprovedStatus>	Optional	0 - 1	String (1)	**Y** = Include approved releases. **N** = Exclude approved releases. ***NOTE:*** Part of release status flag group. If *no* tag in group has explicit value, default is Y. If *any* tag in group has explicit value, default is N.
<searchForBackedOutStatus>	Optional	0 - 1	String (1)	**Y** = Include backed out releases. **N** = Exclude backed out releases. ***NOTE:*** Part of release status flag group. If *no* tag in group has explicit value, default is Y. If *any* tag in group has explicit value, default is N.
<searchForBaselineStatus>	Optional	0 - 1	String (1)	**Y** = Include baseline releases. **N** = Exclude baseline releases. ***NOTE:*** Part of release status flag group. If *no* tag in group has explicit value, default is Y. If *any* tag in group has explicit value, default is N.
<searchForBlockedStatus>	Optional	0 - 1	String (1)	**Y** = Include blocked releases. **N** = Exclude blocked releases. ***NOTE:*** Part of release status flag group. If *no* tag in group has explicit value, default is Y. If *any* tag in group has explicit value, default is N.
<searchForCmnScheduler>	Optional	0 - 1	String (1)	**Y** = Include releases with a scheduler type of CMN. **N** = Exclude releases with a scheduler type of CMN.
<searchForDeletedStatus>	Optional	0 - 1	String (1)	**Y** = Include deleted releases. **N** = Exclude deleted releases. ***NOTE:*** Part of release status flag group. If *no* tag in group has explicit value, default is Y. If *any* tag in group has explicit value, default is N.
<searchForDeliveredStatus>	Optional	0 - 1	String (1)	**Y** = Include delivered releases. **N** = Exclude delivered releases. ***NOTE:*** Part of release status flag group. If *no* tag in group has explicit value, default is Y. If *any* tag in group has explicit value, default is N.
<searchForDevelopmentStatus>	Optional	0 - 1	String (1)	**Y** = Include releases in development. **N** = Exclude releases in development. ***NOTE:*** Part of release status flag group. If *no* tag in group has explicit value, default is Y. If *any* tag in group has explicit value, default is N.
<searchForInstalledStatus>	Optional	0 - 1	String (1)	**Y** = Include installed releases. **N** = Exclude installed releases. ***NOTE:*** Part of release status flag group. If *no* tag in group has explicit value, default is Y. If *any* tag in group has explicit value, default is N.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<searchForManualScheduler>	Optional	0 - 1	String (1)	**Y** = Include releases with a scheduler type of MANUAL. **N** = Exclude releases with a scheduler type of MANUAL.
<searchForOtherScheduler>	Optional	0 - 1	String (1)	**Y** = Include releases with a scheduler type of OTHER. **N** = Exclude releases with a scheduler type of OTHER.
<searchForRejectedStatus>	Optional	0 - 1	String (1)	**Y** = Include rejected releases. **N** = Exclude rejected releases. ***NOTE:*** Part of release status flag group. If *no* tag in group has explicit value, default is Y. If *any* tag in group has explicit value, default is N.
<searchForReleaseApplNames>	Optional	0 - 1	String (255), variable	Release application name. Multiple entries are allowed, delimited by semicolons. Asterisk (*) wildcard may be used.
<searchForReleaseApproval Entities>	Optional	0 - 1	String (255), variable	Release approval entity. Multiple entries are allowed, delimited by semicolons. Asterisk (*) wildcard may be used.
<searchForReleaseSites>	Optional	0 - 1	String (255), variable	Release site. Multiple entries are allowed, delimited by semicolons. Asterisk (*) wildcard may be used.
<searchFromDateApproved>	Optional	0 - 1	Date, yyyymmdd	Start date for searching on release approve date.
<searchFromDateBackedOut>	Optional	0 - 1	Date, yyyymmdd	Start date for searching on release back-out date.
<searchFromDateBaselined>	Optional	0 - 1	Date, yyyymmdd	Start date for searching on release baseline date.
<searchFromDateBlocked>	Optional	0 - 1	Date, yyyymmdd	Start date for searching on release block date.
<searchFromDateCreated>	Optional	0 - 1	Date, yyyymmdd	Start date for searching on release create date.
<searchFromDateInstalled>	Optional	0 - 1	Date, yyyymmdd	Start date for searching on release install date.
<searchFromDateRejected>	Optional	0 - 1	Date, yyyymmdd	Start date for searching on release reject date.
<searchFromDateReverted>	Optional	0 - 1	Date, yyyymmdd	Start date for searching on release revert date.
<searchFromInstallDate>	Optional	0 - 1	Date, yyyymmdd	Start date for searching on release install date.
<searchToDateApproved>	Optional	0 - 1	Date, yyyymmdd	End date for searching on release approve date.
<searchToDateBackedOut>	Optional	0 - 1	Date, yyyymmdd	End date for searching on release back-out date.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<searchToDateBaselined>	Optional	0 - 1	Date, yyyymmdd	End date for searching on release baseline date.
<searchToDateBlocked>	Optional	0 - 1	Date, yyyymmdd	End date for searching on release block date.
<searchToDateCreated>	Optional	0 - 1	Date, yyyymmdd	End date for searching on release create date.
<searchToDateInstalled>	Optional	0 - 1	Date, yyyymmdd	End date for searching on release install date.
<searchToDateRejected>	Optional	0 - 1	Date, yyyymmdd	End date for searching on release reject date.
<searchToDateReverted>	Optional	0 - 1	Date, yyyymmdd	End date for searching on release revert date.
<searchToInstallDate>	Optional	0 - 1	Date, yyyymmdd	End date for searching on release install date.

RLSMRLSE SERVICE SEARCH — Reply

The XML reply to a RLSMRLSE SERVICE SEARCH request returns zero to many `<result>` data elements. Each result lists status flags and other information for a release.

The standard `<response>` data element follows any `<result>` tags in the reply and indicates the success or failure of the request. Successful requests have a return code of 00. Unsuccessful requests have a return code of 04 or higher. Because it is the final data element in the XML reply message, the `<response>` tag serves as an end-of-list marker.

The following example shows what the reply message might look like. Data structure details for the `<result>` tag follow the example.

Example XML — RLSMRLSE SERVICE SEARCH Reply

```
<?xml version="1.0"?>
<service name="RLSMRLSE">
 <scope name="SERVICE">
  <message name="SEARCH">
    <result>
    <release>DEMOREL1</release>
    <releaseDesc>general testing</releaseDesc>
    <releaseStatus>DEV</releaseStatus>
    <releaseRequestorName>DAVE BRADY</releaseRequestorName>
    <releaseRequestorPhone>763-416-9999</releaseRequestorPhone>
    <releaseWorkChangeRequest>ERO</releaseWorkChangeRequest>
    <releaseRequestorDept>DEVL</releaseRequestorDept>
    <addReleaseInterfacingApprovers>N</addReleaseInterfacingApprovers>
    <isReleaseAreasConfigured>Y</isReleaseAreasConfigured>
    <releaseSyslibOrder>A</releaseSyslibOrder>
    <auditEnforceIHASetting>Y</auditEnforceIHASetting>
    <auditIgnoreHigherAreas>N</auditIgnoreHigherAreas>
    <isReleaseLinked>N</isReleaseLinked>
    <isAutoFixDevPackages>Y</isAutoFixDevPackages>
    <isAutoFixFrzPackages>Y</isAutoFixFrzPackages>
    <isAutoFixAprPackages>Y</isAutoFixAprPackages>
    <isAllReleaseSitesInstalled>N</isAllReleaseSitesInstalled>
```

```
<isReleaseSiteBackedOut>N</isReleaseSiteBackedOut>
<isReleaseInstallBuildJclPending>N</isReleaseInstallBuildJclPending>
<isReleaseInstallPending>N</isReleaseInstallPending>
<isReleaseRevertPending>N</isReleaseRevertPending>
<isReleaseBackoutPending>N</isReleaseBackoutPending>
<isReleaseApprovalPending>N</isReleaseApprovalPending>
<isReleaseBlockPending>N</isReleaseBlockPending>
<releaseSchedulerCmn>Y</releaseSchedulerCmn>
<releaseSchedulerManual>Y</releaseSchedulerManual>
<releaseSchedulerOther>N</releaseSchedulerOther>
<releaseSchedulerType>CMN</releaseSchedulerType>
<releaseProblemActionCode>1</releaseProblemActionCode>
<releaseAuditMinRule>0</releaseAuditMinRule>
<releaseCheckinMinRule>0</releaseCheckinMinRule>
<releaseBlockingMinRule>0</releaseBlockingMinRule>
<releaseRetrieveMinRule>0</releaseRetrieveMinRule>
<releaseApproveMinRule>0</releaseApproveMinRule>
<releaseFromInstallDate>20081230</releaseFromInstallDate>
<releaseFromInstallTime>120000</releaseFromInstallTime>
<releaseToInstallDate>20080125</releaseToInstallDate>
<releaseToInstallTime>190000</releaseToInstallTime>
<releaseDateCreated>20080125</releaseDateCreated>
<releaseTimeCreated>142124</releaseTimeCreated>
<releaseCreator>WSER99</releaseCreator>
<releaseImplInst>Implementation Instructions</releaseImplInst>
<totalReleasePackages>00000</totalReleasePackages>
<releaseHighLevelName>CMNDEV.A</releaseHighLevelName>
<releaseDsnPattern>HRAL</releaseDsnPattern>
</result>
.
.
.
<response>
<statusMessage>CMR8700I - Release Search service completed
 <statusMessage>
<statusReturnCode>00</statusReturnCode>
<statusReasonCode>8700</statusReasonCode>
</response>
</message>
</scope>
</service>
```

RLSMRLSE SERVICE SEARCH <result> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<addReleaseInterfacingApprovers>	Optional	0 - 1	String (1)	**Y** = Add related approvers. **N** = Do not add related approvers.
<auditEnforceIHASetting>	Optional	0 - 1	String (1)	**Y** = Enforce IHA setting. **N** = Do not enforce IHA settings.
<auditIgnoreHigherAreas>	Optional	0 - 1	String (1)	**Y** = Ignore higher areas. **N** = Do not ignore higher areas. **C** = If the current release has only a single path through it, process as if "Y" had been specified. If the current release has multiple paths through it, process as if "N" had been specified.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<isAllReleaseSitesInstalled>	Optional	0 - 1	String (1)	**Y** = All release sites are installed. **N** = All release sites are not installed.
<isAutoFixAprPackages>	Optional	0 - 1	String (1)	**Y** = Automatically cleanup APR packages. **N** = Do not automatically cleanup APR packages.
<isAutoFixDevPackages>	Optional	0 - 1	String (1)	**Y** = Automatically cleanup DEV packages. **N** = Do not automatically cleanup DEV packages.
<isAutoFixFrzPackages>	Optional	0 - 1	String (1)	**Y** = Automatically cleanup FRZ packages. **N** = Do not automatically cleanup FRZ packages.
<isReleaseApprovalPending>	Optional	0 - 1	String (1)	**Y** = Release approval is pending. **N** = Release approval is not pending.
<isReleaseAreasConfigured>	Optional	0 - 1	String (1)	**Y** = Release areas are configured. **N** = Release areas are not configured.
<isReleaseBackoutPending>	Optional	0 - 1	String (1)	**Y** = Release backout is pending. **N** = Release backout is not pending.
<isReleaseBlockPending>	Optional	0 - 1	String (1)	**Y** = Release block is pending. **N** = Release block is not pending.
<isReleaseInstallBuildJclPending	Optional	0 - 1	String (1)	**Y** = Release install JCL builds are pending. **N** = Release install JCL builds are not pending.
<isReleaseInstallPending>	Optional	0 - 1	String (1)	**Y** = Release install is pending. **N** = Release install is not pending.
<isReleaseLinked>	Optional	0 - 1	String (1)	**Y** = Release is linked. **N** = Release is not linked.
<isReleaseRevertPending>	Optional	0 - 1	String (1)	**Y** = Release revert is pending. **N** = Release revert is not pending.
<isReleaseSiteBackedOut>	Optional	0 - 1	String (1)	**Y** = Release site is backed out. **N** = Release site is not backed out.
<release>	Optional	0 - 1	String (8), variable	Release name.
<releaseApplName>	Optional	0 - 1	String (4), variable	Release application name.
<releaseApprovalEntity>	Optional	0 - 1	String (8), variable	Release approval entity.
<releaseApproveMinRule>	Optional	0 - 1	Integer (1)	Release minimum approval rule. Values: **0** = No approvals required. **1** = Approvals required before checkin. **2** = Approvals required before advancing to the next area (check-off). **3** = Rule 1 and rule 2.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<releaseAuditMinRule>	Optional	0 - 1	Integer (1)	Release area minimum audit rule. Values: **0** = Audit optional. **1** = Audit required. RC < 20 (audit failure). **2** = Audit required. RC <= 12 (out-of-sync errors within audited area). **3** = Audit required. RC <= 8 (out-of-sync errors with respect to next areas/final areas in prior releases/baseline). **4** = Audit required. RC <= 4 (no out-of-sync errors but some duplicates exist). **5** = Audit required. RC = 0 (no out-of-sync errors and no warnings).
<releaseAuditReturnCode>	Optional	0 - 1	String (2)	Release audit return code.
<releaseBackoutUserid>	Optional	0 - 1	String (8), variable	User ID of user who backed out release.
<releaseBlockUserid>	Optional	0 - 1	String (8), variable	User ID of user who blocked release.
<releaseBlockingMinRule>	Optional	0 - 1	Integer (1)	Release minimum blocking rule. Values: **0** = Wide open. **1** = Audit required. **2** = User must pass entity check (and rule 0). **3** = Rule 1 and rule 2.
<releaseBuildNumber>	Optional	0 - 1	String (10), variable	Release build number.
<releaseCheckinMinRule>	Optional	0 - 1	Integer (1)	Release minimum checkin rule. Values: **0** = Wide open. **1** = Area or package audit required before moving to next area. **2** = Area must be blocked (or package frozen) before moving to next area. **3** = User must pass entity check (and rule 0). **4** = Rule 1 and rule 2. **5** = Rule 1 and rule 3. **6** = Rule 2 and rule 3. **7** = Rule 1, rule2, and rule 3.
<releaseCreator>	Optional	0 - 1	String (8), variable	User ID of user who created release.
<releaseDateApproved>	Optional	0 - 1	Date, yyyymmdd	Date that release was approved.
<releaseDateBackedOut>	Optional	0 - 1	Date, yyyymmdd	Date that release was backed out.
<releaseDateBaselined>	Optional	0 - 1	Date, yyyymmdd	Date that release was baselined.
<releaseDateBlocked>	Optional	0 - 1	Date, yyyymmdd	Date that release was blocked.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<releaseDateCreated>	Optional	0 - 1	Date, yyyymmdd	Date that release was created.
<releaseDateDisReceived>	Optional	0 - 1	Date, yyyymmdd	Date that DEV site received all notifications that release was successfully distributed to all remote sites.
<releaseDateDistributed>	Optional	0 - 1	Date, yyyymmdd	Date that release was distributed.
<releaseDateInstalled>	Optional	0 - 1	Date, yyyymmdd	Date that release was installed.
<releaseDateMemoDeleted>	Optional	0 - 1	Date, yyyymmdd	Date that release was memo-deleted.
<releaseDateRejected>	Optional	0 - 1	Date, yyyymmdd	Date that release was rejected.
<releaseDateReverted>	Optional	0 - 1	Date, yyyymmdd	Date that release was reverted.
<releaseDesc>	Optional	0 - 1	String (72), variable	Release description.
<releaseDsnPattern>	Optional	0 - 1	String (5), variable	Release dataset name pattern.
<releaseFromInstallDate>	Optional	0 - 1	Date, yyyymmdd	Install start date.
<releaseFromInstallTime>	Optional	0 - 1	Time, hhmmss	Install start time.
<releaseGenerateMinRule>	Optional	0 - 1	Integer (1)	Release area minimum generate rule. Values: **0** = Wide open. **1** = User must pass entity check.
<releaseHighLevelName>	Optional	0 - 1	String (8), variable	Release high-level qualifier.
<releaseHighLevelPath>	Optional	0 - 1	String (1024), variable	Release high-level HFS path.
<releaseImplInst>	Optional	0 - 1	String (72), variable	Release implementation instructions.
<releaseLibsAged>	Optional	0 - 1	String (1)	Release libraries aged. Values (Y/N)/
<releaseMemoDeleteUserid>	Optional	0 - 1	String (8), variable	User ID of user who memo-deleted release.
<releaseOtherProblemAction>	Optional	0 - 1	String (72)	Release other problem description.
<releaseProblemActionCode>	Optional	0 - 1	String (1)	Action to be taken if a problem occurs when a package is installed. Values: **1** = Hold production and contact analyst. **2** = Back out change and continue production. **3** = Other
<releaseRejectUserid>	Optional	0 - 1	String (8), variable	User ID of user who rejected release.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<releaseRemoteSite>	Optional	0 - 1	String (8), variable	Release site name.
<releaseRequestorDept>	Optional	0 - 1	String (8), variable	Release requester department.
<releaseRequestorName>	Optional	0 - 1	String (25), variable	Release requester name.
<releaseRequestorPhone>	Optional	0 - 1	String (15), variable	Release requester phone number.
<releaseRetrieveMinRule>	Optional	0 - 1	Integer (1)	Release minimum retrieve rule. Values: **0** = Retrieve allowed from any area (blocked or unblocked). **1** = Retrieve allowed only from unblocked areas. **2** = Must pass entity check (and rule 0). **3** = Rule 1 and rule 2.
<releaseRevertUserid>	Optional	0 - 1	String (8), variable	User ID of user who reverted release.
<releaseSchedulerCmn>	Optional	0 - 1	String (1)	**Y** = Allow CMN internal release scheduler. **N** = Do not allow CMN internal release scheduler.
<releaseSchedulerManual>	Optional	0 - 1	String (1)	**Y** = Allow manual release scheduler. **N** = Do not allow manual release scheduler.
<releaseSchedulerOther>	Optional	0 - 1	String (1)	**Y** = Allow other release scheduler. **N** = Do not allow other release scheduler.
<releaseReleaseSchedulerType>	Optional	0 - 1	String (8), variable	Release scheduler type. Values: **CMN** = ChangeMan ZMF. The installation jobs are submitted by the ZMF started task at the scheduled install date and time. **MANUAL** = Manual. The installation jobs are submitted as soon as the package approvals are completed. **OTHER** = Other. The installation jobs are inserted into a third-party scheduler via a batch job.
<releaseStatus>	Optional	0 - 1	String (3), variable	Release status. Values: **APR** = Approved **BAK** = Backed out **BAS** = Baselined **BLK** = Blocked **DEL** = Deleted **DEV** = In development **DIS** = Distributed **INS** = Installed **REJ** = Rejected
<releaseSyslibOrder>	Optional	0 - 1	String (1)	Release SYSLIB sort order. Values: **A** = Ascending **D** = Descending

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<releaseTimeApproved>	Optional	0 - 1	Time, hhmmss	Time that release was approved.
<releaseTimeBackedOut>	Optional	0 - 1	Time, hhmmss	Time that release was backed out.
<releaseTimeBaselined>	Optional	0 - 1	Time, hhmmss	Time that release was baselined.
<releaseTimeBlocked>	Optional	0 - 1	Time, hhmmss	Time that release was blocked.
<releaseTimeCreated>	Optional	0 - 1	Time, hhmmss	Time that release was created.
<releaseTimeDisReceived>	Optional	0 - 1	Time, hhmmss	Time that DEV site received all notifications that release was successfully distributed to all remote sites.
<releaseTimeDistributed>	Optional	0 - 1	Time, hhmmss	Time that release was distributed.
<releaseTimeInstalled>	Optional	0 - 1	Time, hhmmss	Time that release was installed.
<releaseTimeMemoDeleted>	Optional	0 - 1	Time, hhmmss	Time that release was memo-deleted.
<releaseTimeRejected>	Optional	0 - 1	Time, hhmmss	Time that release was rejected.
<releaseTimeReverted>	Optional	0 - 1	Time, hhmmss	Time that release was reverted.
<releaseToInstallDate>	Optional	0 - 1	Date, yyyymmdd	Install end date.
<releaseToInstallTime>	Optional	0 - 1	Time, hhmmss	Install end time.
<releaseWorkChangeRequest>	Optional	0 - 1	String (16), variable	Release work change request.
<revertReasons>	Optional	0 - 1	String (72), variable	Release revert reasons.
<totalReleasePackages>	Optional	0 - 1	Integer (5)	Total number of packages in release.

RLSMRLSE SERVICE TEST

The RLSMRLSE SERVICE TEST message tests the contents of a release against all of the packages that may place a component in that release. The reply message does not return any data, only a response indicating success or failure.

The XML service/scope/message tags and attributes for this message are:

```
<service name="RLSMRLSE">
<scope name="SERVICE">
<message name="TEST">
```

These tags appear in both requests and replies.

RLSMRLSE SERVICE TEST — Request

The following example shows how you might code a request to test the contents of a release. Data structure details for the <request> tag follow the example.

NOTE Job cards are only required if you are using the auto-cleanup subtags:
<cleanupComponentFromDiffPkg>
<cleanupEmptyPackage>
<cleanupNotCheckedInComponent>
For example, job cards would be required if an auto-cleanup operation needed to demote a component prior to deletion.

Example XML — RLSMRLSE SERVICE TEST Request

```
<?xml version="1.0"?>
<service name="RLSMRLSE">
 <scope name="SERVICE">
  <message name="TEST">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
    <release>S4712COM</release>
   </request>
  </message>
 </scope>
</service>
```

RLSMRLSE SERVICE TEST <request> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<cleanupComponentFromDiffPkg>	Optional	0 - 1	String (1)	**Y** = If different versions of a component exist in different packages, the version not checked into the release will be deleted from the package. **N** = Do not delete components with different versions.
<cleanupEmptyPackage>	Optional	0 - 1	String (1)	**Y** = Automatically cleanup empty packages. **N** = Do not cleanup empty packages.
<cleanupNotCheckedInComponent>	Optional	0 - 1	String (1)	**Y** = Automatically cleanup components that were not checked into release. **N** = Do not cleanup components that were not checked into release.
<jobCard01>	Optional	0 - 1	String (72), variable	Job card 1.
<jobCard02>	Optional	0 - 1	String (72), variable	Job card 2.
<jobCard03>	Optional	0 - 1	String (72), variable	Job card 3.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<jobCard04>	Optional	0 - 1	String (72), variable	Job card 4.
<jobCards>	Optional	0 - 4	String (72), variable	Deprecated. Use tags <jobCard01> through <jobCard04> instead.
<release>	Required	1	String (8), variable	Release name.

RLSMRLSE SERVICE TEST — Reply

The XML reply to a RLSMRLSE SERVICE TEST request does not return any `<result>` data elements. The `<response>` data element indicates the success or failure of the request. Successful requests have a return code of 00. Unsuccessful requests have a return code of 04 or higher.

The following examples show what a reply message might look for a successful and unsuccessful request.

Example XML — RLSMRLSE SERVICE TEST Replies

Successful Request

```
<?xml version="1.0"?>
<service name="RLSMRLSE">
 <scope name="SERVICE">
  <message name="TEST">
   <response>
    <statusMessage>CMR1507I - Release S4711010/GENLEDGR and package
      components match.</statusMessage>
    <statusReturnCode>00</statusReturnCode>
    <statusReasonCode>1507</statusReasonCode>
   </response>
  </message>
 </scope>
</service>
```

Unsuccessful Request

```
<?xml version="1.0"?>
<service name="RLSMRLSE">
 <scope name="SERVICE">
  <message name="TEST">
   <response>
    <statusMessage>CMR0001A - Invalid Release status BAS, S4712COM must be in
     DEV,BLK,APR,REJ status.</statusMessage>
    <statusReturnCode>08</statusReturnCode>
    <statusReasonCode>0001</statusReasonCode>
   </response>
  </message>
 </scope>
</service>
```

RLSMRLSE SITES LIST

The RLSMRLSE SITES LIST service gathers the remote site list by getting the remote site data from all the packages attached to the release. It then combines those lists and returns the release site list.

The XML service/scope/message tags and attributes for this message are:

```
<service name="RLSMRLSE">
<scope name="SITES">
<message name="LIST">
```

These tags appear in both requests and replies.

RLSMRLSE SITES LIST — Request

The following example shows how you might code a request to list all of the release site information for a release. Data structure details for the `<request>` tag follow the example.

Example XML — RLSMRLSE SITES LIST Request

```
<?xml version="1.0"?>
<service name="RLSMRLSE">
 <scope name="SITES">
  <message name="LIST">
   <header>
    <subsys>4</subsys>
    <test> </test>
    <product>CMN</product>
   </header>
  <request>
    <release>S4712010</release>
   </request>
  </message>
 </scope>
</service>
```

RLSMRLSE SITES LIST <request> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<release>	Required	1	String (8), variable	Release name.
<siteName>	Optional	0 - 1	String (8), variable	Site name.

RLSMRLSE SITES LIST — Reply

The XML reply to a RLSMRLSE SITES LIST request does returns zero to many `<result>` data elements. Each result lists dates and contact information for a release site.

The standard <response> data element follows any <result> tags in the reply and indicates the success or failure of the request. Successful requests have a return code of 00. Unsuccessful requests have a return code of 04 or higher. Because it is the final data element in the XML reply message, the <response> tag serves as an end-of-list marker.

The following example shows what the reply message might look like. Data structure details for the <result> tag follow the example.

Example XML — RLSMRLSE SITES LIST Reply

```
<?xml version="1.0"?>
<service name="RLSMRLSE">
 <scope name="SITES">
  <message name="LIST">
   <result>
    <release>S4712010</release>
    <siteName>SERT4</siteName>
    <installDate>20131231</installDate>
    <fromInstallTime>080000</fromInstallTime>
    <toInstallTime>235900</toInstallTime>
    <contactName>Hung Nguyen</contactName>
    <contactPhone>808-555-1214</contactPhone>
    <alternateContactName>Wenwei Zheng</alternateContactName>
    <alternateContactPhone>808-555-1212</alternateContactPhone>
    <siteStatus>DEV</siteStatus>
   </result>
   <response>
    <statusMessage>CMR8700I - LIST Sites service completed</statusMessage>
    <statusReturnCode>00</statusReturnCode>
    <statusReasonCode>8700</statusReasonCode>
   </response>
  </message>
 </scope>
</service>
```

RLSMRLSE SITES LIST <result> Data Structure

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<alternateContactName>	Optional	0 - 1	String (25), variable	Alternate analyst name.
<alternateContactPhone>	Optional	0 - 1	String (15), variable	Alternate analyst phone number.
<contactName>	Optional	0 - 1	String (25), variable	Originating analyst name.
<contactPhone>	Optional	0 - 1	String (15), variable	Originating analyst phone number.
<fromInstallTime>	Optional	0 - 1	Time, hhmmss	Start time for installation time range.
<installDate>	Optional	0 - 1	Date, yyyymmdd	Install date.
<release>	Optional	0 - 1	String (8), variable	Release name.

Subtag	Use	Occurs	Data Type & Length	Values & Dependencies
<siteName	Optional	0 - 1	String (8), variable	Site name.
<siteStatus>	Optional	0 - 1	String (3)	Determined site status.
<toInstallTime>	Optional	0 - 1	Time, hhmmss	End time for installation time range.

Index