System-Link User's Guide

Release 9

INFOR™

Publication Information

Release: Infor ERP XA Release 9

Publication Date: March 2010

Table of Contents

To the Reader

Consumer demand for the universal availability of information is growing; the expectation that information should be accessible from wherever it is needed, rather than only being available from a single, fixed source.

A business system must be able to support communication with as many disparate systems as possible: workstations, Internet browsers, cell phones, PDAs, pagers, or business-to-business transactions involving servers and Microsoft COM, Enterprise JavaBeans, CICS, SQL, HTTP, etc. XA System-Link provides XA the ability to participate in this new world of intercomputing.

This guide assumes a working knowledge of XML, which is beyond the scope of this guide.

In this guide:

What is System-Link?	➢ Describes the high-level System-Link concepts.
Accessing System-Link via a Web Server	➢ Describes how to connect to System-Link over an HTTP connection.
Accessing System-Link via the Local Program Interface	➢ Describes how to connect to System-Link from a System i server program using program calls.
Accessing System-Link via System-Link MQ	➢ Describes how to connect to System-Link over WebSphere MQ.
The System-Link Protocol in Detail	➢ Describes the System-Link command protocol.
Copying a System-Link request from the Browser	➢ Describes how to provide the System-Link request automatically from the menu bar.
Using the System-Link Simulator	➢ Describes a method for testing your System-Link request before putting it into production.
Examples	➢ Illustrates the System-Link syntax using typical business examples.
XML Syntax for Requests and Responses	➢ Provides the detailed syntax that external applications will require in order to communicate with XA via System-Link.
Document Type Definitions (DTD)	➢ Shows a compact form of what is detailed in Examples
Supported Objects	➢ Pointer to obtaining lists of objects supported by System-Link.

Notes:
- o If you intend to use System-Link MQ (System-Link via WebSphere MQ), see the "System-Link MQ Installation and Configuration Guide".
- o Link-Manager now includes and uses an embedded web server, which will run on your System i as part of normal installation and operation. As a result, the installation of a separate web server for use by System-Link or the Simulator is no longer required. It is anticipated that this "plug-and-play" web server solution will meet the needs for the majority of installations. However, System-Link may still be configured to use an external web server (e.g. IBM WebSphere Application Server), for those organizations requiring a more extensive web server feature set.
- o If you wish to use System-Link with Integrator objects or Infor ERP XA primary objects that have been extended with Integrator, you must have Enterprise Integrator installed. These objects will not be usable through System-Link with Standard Integrator. In the case of extended Infor ERP XA primary objects, this means that <u>the entire object</u> (not just its extension) <u>will be unavailable through System-Link without installing Enterprise Integrator</u>.
- o For simplicity in the remainder of this book Infor ERP XA will be referred to as XA.

Chapter 1.   What is System-Link?

System-Link is a bridge between outside systems and XA. The term *outside system* could mean one or more of any number of programs and platforms, including:

- A web page
- An RPG program
- A Java program on a System i or an auxiliary machine
- Other ERP systems on other platforms

The calling program can be any system that can send and receive XML. As long as it can get its XML to System-Link it can interoperate with XA. It should come as no surprise, therefore, that there are a number of ways to call in to System-Link. For example:

- A web page (HTML) or Java program would communicate via your web server

- An RPG program can communicate via either your web server or directly using a local program call.

- Other ERP systems may interact with each other and System-Link via WebSphere MQ.

In short, the way System-Link is called may depend on language, platform and business needs. RPG, HTML and Java may each have a preferred technique for calling System-Link. When different platforms are involved, the communications link (LAN, internet, WebSphere MQ…) may dictate how System-Link is called. It is not important that you call System-Link in the "right" way, because there often is no one right way. It is important to understand the calling options available and the relative advantages of each.

At a high level, there are three ways to call System-Link: the local program interface (LPI), the web server and a message queuing system. The following diagram illustrates how three different programs might interact with System-Link via one or more of these methods:
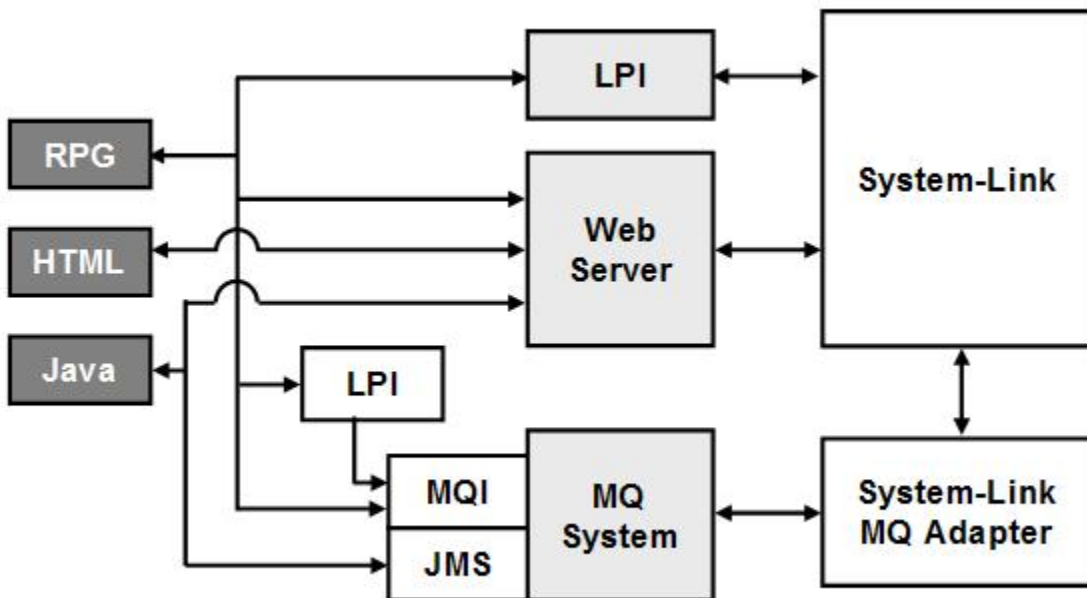
Web Server

One of the easiest ways to use System-Link – and one of the quickest ways to obtain business benefits – is to use it with web pages. It is very easy to create a secure web page to inquire into XA data. Many companies supply customers, vendors, and partners with a user ID and password so they can get up-to-the-minute transaction data from the company's web site. In such a scenario, the communication between the web page and System-Link would be through the web server. A company could also access System-Link from the web server using programs written in RPG or Java. Since the client framework includes an embedded web server, this method is also relatively "plug and play".

Local Program Interface

Often companies wish to interact with XA using their own RPG programs. In such a case, using the web server is not required. The LPI allows an RPG program to communicate directly with System-Link.

Message Queuing System

A message queue system provides a communications layer allowing synchronous or asynchronous communications between multiple processes. A program using WebSphere MQ can interact with System-Link by placing a request on a specific request queue, and then awaiting the response on its own response queue.

The method with which a program interacts with System-Link MQ depends on the language of the program:

- Java programs can access WebSphere MQ directly via JMS

- RPG programs can access WebSphere MQ directly via MQI (provided by IBM)

- RPG programs may use System-Link MQ without dealing directly with WebSphere MQ by using the Local Program Interface (LPI) provided by Infor. From the programmer's perspective, this interface is identical to the LPI described earlier. The difference is that one server program calls System-Link directly, while the other server program calls System-Link via WebSphere MQ.

All of these methods yield similar results, so the decision is one of programmer preference. Programmers who already have WebSphere MQ and are familiar with MQI or JMS may prefer those interfaces. Programmers who already use the LPI to communicate to System-Link directly may prefer to use the same interface to communicate with WebSphere MQ.

Link Manager is the facility that allows a company to distribute the server load across a variety of system resources. These system resources include the System i, IXS cards and LPARs in the System i, NT or Linux servers on the LAN, etc. Link Manager also controls the deployment of the System-Link MQ Adapter, which serves as a bridge between WebSphere MQ and System-Link.

Chapter 2.   Accessing System-Link via a Web Server

Link-Manager has an embedded web server that is installed during the normal installation process. This web server runs on your System i, and can be accessed by the base URL http://<your System i name>:<your secondary port>, where

- *<your System i name>* is the name of your System i (e.g. "my400.mycompany.com"), and
- *<your secondary port>* is client framework's secondary port number. Unless you have explicitly changed the port during installation, this value will be "36001". (If required, the secondary port number can be validated by consulting the QPRINT files associated with the MPX processes started by the UJOB. The port number will appear as a parameter to the processes.)

The base URL for the above example would therefore be "http://my400.mycompany.com:36001".

Interfacing with the System-Link servlet via HTTP is done by passing your System-Link request in the body of a POST message. The message body contains a single key-value pair: the key "SystemLinkRequest" and the value consisting of your System-Link request document. (Alternately, if the key "SystemLinkRequestURL" is used, the value should consist of the URL where your System-Link request document can be obtained.)

An example web page using the System-Link servlet is the System-Link Simulator (a request testing and validation tool that is described later in this Guide), which can be found at http://<your System i name>:<your secondary port>/SystemLink. The source for this web page can be found in the IFS folder "/XA/Web/SystemLink".

The embedded web server is considered the "default" for System-Link, and is designed to provide a turnkey solution that will meet the needs of the majority of installations. If your requirements dictate that a separate web server be used (e.g. IBM WebSphere Application Server), then the System-Link servlet can also be run in that alternate web server. The installation and configuration of an alternate web server is beyond the scope of this document. Please consult the information provided by your web server vendor, as well as the installation guides found in the same location where you found this book.

Chapter 3.   Accessing System-Link via the Local Program Interface

Overview

System-Link provides an XML-based interface to the XA applications. This enables applications to interact with XA via XML from a variety of platforms. The System-Link Local Program Interface (LPI) makes it possible to perform this same integration from server applications without requiring communications through a web server or WebSphere MQ.

The same System-Link request that the server programmer would send to the web server can be sent instead to the LPI. The LPI will return the same System-Link response that the web server would return. The only difference is that the System-Link documents are passed via program parameters.

LPI Comparison

There are two RPG programs that provide implementations of the Local Program Interface: one that interfaces with System-Link directly and one that interfaces with System-Link via WebSphere MQ. The following comments describe their differences:

1. LPI (direct) – This program makes a direct call to System-Link. No installation of WebSphere MQ is required. The communication is entirely synchronous: the System-Link server associated with the requested environment must be running at the time that the call is made. If the server is not available, an error will be returned.

2. LPI (via MQ) – This program communicates with System-Link via WebSphere MQ and the System-Link MQ Adapter. This version requires the installation and configuration of WebSphere MQ and Directory Services. The communication can be asynchronous in that the System-Link server associated with the requested environment does not have to be running at the time the call is made. However, if the server is not available the caller will block until the request is completed.

Note: if you intend to use LPI via MQ, please read the next chapter "Accessing System-Link via System-Link MQ". It contains further relevant installation and usage information.

Interacting with the Local Program Interface

The steps involved in using the LPI are as follows:

- The application program creates a System-Link request.

- The LPI is called with the System-Link request as a parameter. The size of the request parameter is limited to 64K characters.

- The System-Link server processes the request and returns the System-Link response to the LPI.

- The LPI sends the System-Link back to the application program. A message is returned if an error condition is encountered.

Program parameters

The program name of the Local Program Interface is PSVPSR1R. It has the following parameters:

Parameter	Description	Usage	Size	Type
P#XREQ	System-Link request	I	65,535	A
P#XRSP	System-Link response	O	65,535	A
P#MRQR	More query results (*YES/*NO)	O	8	A
P#ERFD	Internal errors (*YES/*NO)	O	8	A
P#MSID	External errors (PSX0122/PSX0130)	O	7	AV

P#XREQ   This is the System-Link request. The request cannot be longer than 64K bytes.

P#XRSP   This is the System-Link response. If the response is longer than 64K bytes PSX0130 is returned indicating that the response was too large.

P#MRQR   A return value of *YES* indicates that there are more records than what have been returned so far. Note: for this parameter to ever return *YES* your request must include *retainResult='true'*.

P#ERFD   When the System-Link response contains the *<Message type=error>* tag, this parameter is set to *YES*. This indicates that the response contains error messages.

P#MSID   There are three values for this parameter:

- (blank)   The request was processed normally

- PSX0122   There was an error in processing. For example, the System-Link server was not started.

- PSX0130   The response was too large. You should change your request in order to reduce the response. For example, insert a maxReturned value when requesting a list of records.

Example 1: Request

The following XML represents a request to change the quantity of requisition 1234 to 101.000 in environment P0. (The user ID is shown as UUUUUUUUU and the password is shown as PPPPPPPPP.)

```
<?xml version='1.0' encoding='ISO-8859-1'?><!DOCTYPE System-Link SYSTEM 'SystemLinkRequest.dtd'>
<System-Link>
  <Login principal='UUUUUUUUU' credentials='PPPPPPPPP' maxIdle='900000'

properties='com.mapics.cas.domain.EnvironmentId=P0,
com.mapics.cas.domain.SystemName=USATLD06,com.mapics.cas.user.LanguageId=en'/>

    <Request sessionHandle='*current' workHandle='*new' broker='EJB' maxIdle='900000'>
    <Update name='update_req_1' domainClass='com.mapics.pm.Requisition'>
     <DomainEntity>
      <Key><Property path='requisition'><Value><![CDATA[1234]]></Value></Property></Key>
      <Property path='quantity'><Value><![CDATA[101.000]]></Value></Property>
     </DomainEntity>
    </Update>
    </Request>
</System-Link>
```

To send this request to the Local Program Interface, the following command would be used:

call psvpsr1r ('<?xml version="1.0" encoding="'ISO-8859-1"?><!DOCTYPE System-Link SYSTEM "SystemLinkRequest.dtd"> <System-Link> <Login principal="UUUUUUUUU" credentials="PPPPPPPPP" maxIdle="900000" properties="com.mapics.cas.domain.EnvironmentId=P0, com.mapics.cas.domain.SystemName=USATLD06, com.mapics.cas .user.LanguageId=en"/> <Request sessionHandle="*current" workHandle="*new" broker="EJB" maxIdle="900000">< Update name="update_req_1" domainClass="com.mapics.pm.Requisition"> <DomainEntity> <Key> <Property path=" requisition"> <Value><![CDATA[1234]]></Value></Property></Key><Property path="quantity"><Value><![CDATA[101 .000]]></Value> </Property> </DomainEntity> </Update> </Request> </System-Link>' ' ' ' ' ' ' ' ')

This command can be described in sections as follows.

Call the program

```
call psvpsr1r
```

This is the call to the program.

First parameter: P#XREQ

```
call psvpsr1r ('<?xml version= "1.0"…</System-Link>'
```

The first parameter is the System-Link request shown above. Most of it has been removed, as shown by the ellipsis (…) to simplify the illustration.

Single quotes define the parameter. Since the System-Link request also contains quotes, they are represented as double-quotes. For example, there are double-quotes around the version (i.e., "1.0") versus the single quotes (i.e., '1.0') in the System-Link request at the beginning of the examples topic. Double-quotes need to be used in the XML when it is inside a parameter. The LPI will change them to single quotes when it passes the request to System-Link.

The remaining parameters are all blank: four sets of quote-blank-quote separated by blanks.

```
call psvpsr1r ('<?xml version= "1.0"…</System-Link>' ' ' ' ' ' ' ' ' ')
```

Example 2: Response

If the requisition is found and there are no errors, the following System-Link response is returned:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
  <System-Link version="1.0" hostVersion="1.0">
  <LoginResponse><SessionHandle value="-6816b5eb:f0aa27ee6b:-7e6f" /></LoginResponse>
  <Response sessionHandle="-6816b5eb:f0aa27ee6b:-7e88" workHandle="-6816b5eb:f0aa27ee6b:-7e6d"
/>
  </System-Link>
```

The returned parameters would be as follows:

Parameter	Value	Description
P#XRSP	<?xml version="1.0" … </System-Link>	This is the returned System-Link response
P#MRQR	*NO	The response is complete
P#ERFD	*NO	There are no internal errors
P#MSID	(blank)	The request was processed normally

Example 3: Response with errors

Sometimes a request will generate an error condition. For example, if the requisition number in our example did not exist, the following System-Link response would be returned:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<System-Link version="1.0" hostVersion="1.0">
  <LoginResponse><SessionHandle value="-6816b5eb:f0aa27ee6b:-7e4c" /></LoginResponse>
  <Response sessionHandle="-6816b5eb:f0aa27ee6b:-7e88" workHandle="-6816b5eb:f0aa27ee6b:-7e4a">
    <UpdateResponse name="updateObject_Requisition">
      <Exception name="com.pjx.eScript.RequestDataException">
        <Text><![CDATA[ Error occurred during update]]></Text>
        <Message type="error">
          <Text><![CDATA[ Error occurred during update of com.mapics.pm.Requisition object]]></Text>
        </Message>
        <Message type="error">
          <Text><![CDATA[ (E) Requisition 1234 no longer exists.]]></Text>
          <DetailedText><![CDATA[Explanation . . :   The record you are trying to change or delete no longer
            exists in the database. It might have been deleted by another user or job. Your action: Refresh your
            screen and the record should disappear from the list. If not please contact your XA support
            representative.]]></DetailedText>
        </Message>
        <Message type="error">
          <Text><![CDATA[ (E) The requisition record could not be locked.]]></Text>
          <DetailedText><![CDATA[Explanation . . : You attempted to access a requisition record, but the record is
            being maintained by someone else. Your action . . : Select another requisition to work with or try
            again.]]></DetailedText>
        </Message>
        <Message type="information">
          <Text><![CDATA[ Check for XML errors]]></Text>
          <DetailedText><![CDATA[ Check your XML for errors]]></DetailedText>
        </Message>
      </Exception>
      <Exception name="com.pjx.eScript.TransactionException">
        <Text><![CDATA[ Transaction failed due to nested exceptions]]></Text>
        <Message type="information">
        <Text><![CDATA[ The transaction failed due to the nested exceptions indicated.]]></Text>
        </Message>
      </Exception>
    </UpdateResponse>
  </Response>
</System-Link>
```

The returned parameters would be as follows:

Parameter	Value	Description
P#XRSP	<?xml version="1.0" … </System-Link>	This is the returned System-Link response
P#MRQR	*YES	The response is complete
P#ERFD	*YES	There are internal errors
P#MSID	(blank)	The request was processed normally

You would need to parse the System-Link response in order to find the errors. Parsing XML is described in more detail below.

Programming to the Local Program Interface (LPI)

One of the input parameters to the LPI is the System-Link request. The LPI passes the System-Link request to System-Link where it is processed. System-Link returns a System-Link response to the LPI, which passes it back to the calling program in an output parameter (Figure A).
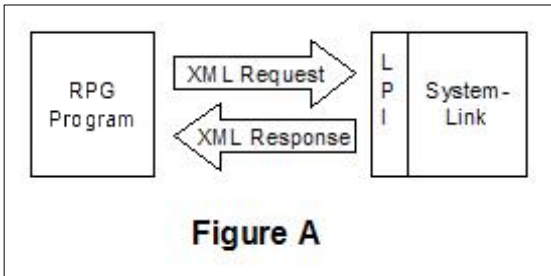
Figure A

Creating these programs involves two steps that may be new to an RPG programmer. First, the System-Link request must be assembled. System-Link provides tools to help create the System-Link request. An option on the Navigation menu provides the System-Link request for the list or card file being displayed, and it can be changed and tested, if necessary, using the Simulator. The request must contain four items that may or may not be variables: the user ID, password, environment and language. In addition, there will usually be one or more input variables, such as customer, vendor or item. All variables must be included in the System-Link request that is sent to the LPI.

Second, the results must be parsed. The RPG program must be able to extract data from the XML string and place it in variables in the RPG program.

Multiple techniques can be used to assemble the System-Link request and parse the System-Link response. RPGLE contains features that facilitate working with string data. We have written some programs to assemble the System-Link request using messages. We have written another program to illustrate how the response can be parsed. These programs can be downloaded[1]. The remainder of this chapter describes these programs.

Example programs

To illustrate how the LPI can be used we will use a set of programs that print a list of PO's for a vendor. The vendor number is an input parameter of the RPG program, which will request a maximum of ten records. If the list is not complete, it will request up to five more, and continue doing so until all the records have been returned. It will then print the results.

The programs you can download are as follows (refer to Figure B):

URSLPI0C

NOTE: You may choose to download these programs, make some necessary changes, and then run them. To simplify that process we have provided this CL program. It will create the messages you need during runtime. Should you choose to employ this message file technique, you would create permanent messages, and a CL program like this one would not be necessary. This chapter describes what the supplied programs do, and then describes what you would do differently to use the same techniques in production.

This CL program writes two messages to a message file in QTEMP. The first is the QueryList request, which is the initial request for PO's. It contains five variables, one of which is the input parameter (i.e., the vendor number). The other four

Figure B

are user ID, password, environment and system name. We will discuss how you can enter your own values for these later. The second message is a FetchNext request, which will be used to get additional PO's (if any). It contains three variables: the session and work handles, and the maximum number of records to be retrieved.

---
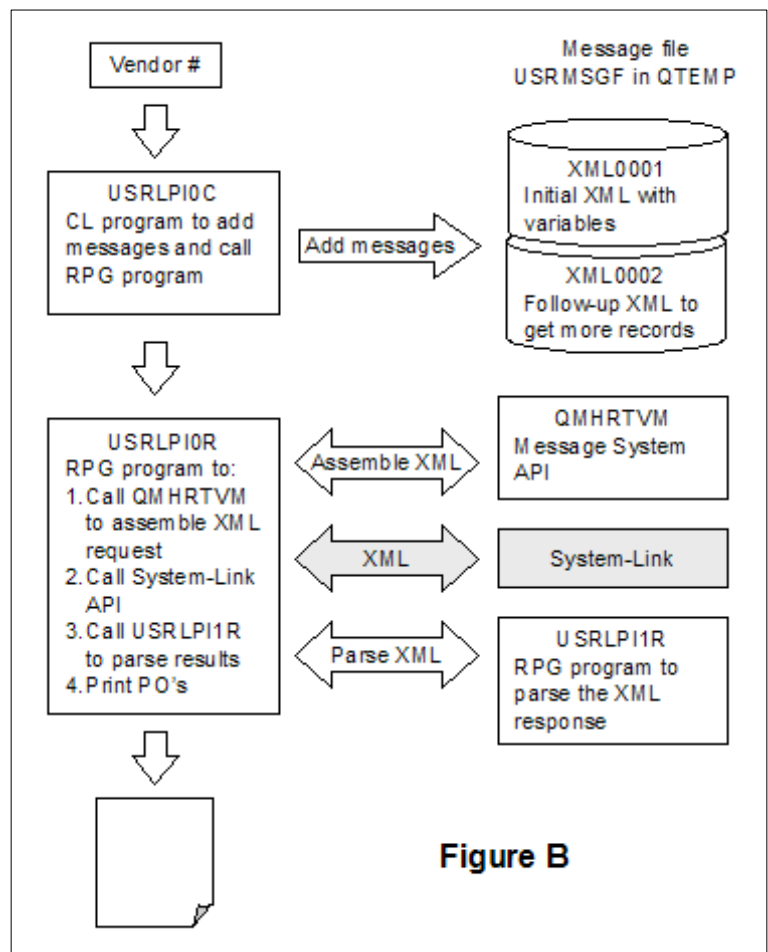
[1] These are not supported programs and are provided for illustration purposes only.

USRLPI0R

This is an RPG program that calls the LPI. It can call the direct LPI or it can call the MQ Series LPI, which is a different program with the same parameters[2]. This program also receives the System-Link response and prints the PO's.

Assembling the System-Link Request

One of the parameters of the LPI is P#XREQ, which holds the System-Link request. It must be a complete request, without any substitution variables. Thus, one of the functions of this program is to replace variables in the System-Link request with actual values. Since the XML is stored in a message, we can use QMHRTVM. This is a system API that can retrieve the message from the message file and replace substitution variables with data. This is exactly what we want to do to assemble our System-Link request.

The parameters for QMHRTVM include the message ID (which contains the System-Link request in the first message created by our CL program) and the message data. Our program sets the message data to the following predefined values:

> - User ID: USER
> - Password: PASSWORD
> - Environment: MM
> - System name: SYSTEM

The data for the fifth substitution variable (the vendor) is supplied by an input parameter. QMHRTVM returns the XML string, which is put into P$XREQ. This is the parameter that will be passed to the LPI.

Calling the LPI

As stated earlier, there are two different LPI's with the same parameters: one for calling System-Link directly and one for calling System-Link via WebSphere MQ. This program has a variable called LPI Option (wLPIOpt) that can be set to any of three values:

> - SystemLink   Call PSVPSR1R (the direct LPI)
> - MQSeries   Call PSVSMI1R (the WebSphere MQ LPI)
> - MQIsample   For use with the LPI provided by IBM (see program USRLPI2R, discussed below)

Based on the value of the LPI Option, the appropriate LPI is called.

Getting more records

The System-Link request is set to retrieve a maximum of ten records. You can increase this by simply replacing the maxReturned value in the System-Link request with whatever value you choose. If you need to retrieve additional records then you will need to use the FetchNext request.

In order to perform the FetchNext request the parsing program (USRLPI1R – discussed below) must be called. The parsing program will retrieve the session and work handles, and pass them back to USRLPI0R, which then uses the same technique described above to assemble the System-Link request. QMHRTVM is called again, but this time the message ID is the second message created by our CL program, and the data to be inserted is the session and work handles, as well as a new maxReturned value (five records instead of the original ten).

Printing the data

The program prints the data in the normal way. However, to be able to print the data it must be able to recognize it. The System-Link response is a long string of character data. In order to print a PO number, for example, our program needs to be able to extract it from the XML string. To do this, it calls USRLPI1R.

USRLPI1R

This program consists of a series of subroutines. The USRLPI0R program calls it when it needs some data. For example, it will call it to get the first (or next) domain entity, which is a record. It will call it to get the first (or next) key field for that domain entity. It will call it to get the first (or next) non-key field for that domain entity, and so forth. The program is called field by field, one at a time. In addition to business data, this program is also called to retrieve the session and work handles needed for the FetchNext request.

---

[2] There is also an MQ interface (MQI) provided by IBM. See discussion of program USRLPI2R.

Because the System-Link response is a long string of character data, the parsing program makes extensive use of scan and substring functions. Every scan is based on character positions in the System-Link response. The beginning and end tags are stored as constants for use in the scan operation. The following simplified example will illustrate how the parsing function works.

Function: get first domain entity's domain class (from subroutine GetDmnEnt)

Step 1: Starting at position 1, <u>scan</u> the System-Link response (p#xrsp) looking for the domain entity tag (DmnBegS). The resulting variable (wPos) will hold the starting position of the domain entity tag. (The starting position for the scan is stored in a field (cDmnNext) and increased as additional records are read.)

Step 2: Add 27 to the position (wPos + 27). This is because the constant representing the domain entity tag is 27 positions long. Place the result in a variable (wStr) which represents the first character of the data we want. In this case, that data is the domain class "com.mapics.pm.PurchaseOrder".

Step 3: Starting from the result in step 2 (i.e., the first position of the data we want) <u>scan</u> the System-Link response (p#xrsp) for the domain entity end tag (DmnBegE). The resulting variable (wEnd) will hold the starting position of the end tag.

Step 4: Now we have the starting position of the data we want and the starting position of the end tag (which is the first position after the data). Subtracting one from the other gives us the length of the data, so we can use the <u>substring</u> function to place the data in the domain class variable (p#dmcl).

All the parsing uses this same technique. The result is that the substring of the XML is placed in a program variable where it can be manipulated by RPG.

One item to note is that the XML data is all character data. This means that a numeric value in the database, such as a quantity, appears as an alphanumeric value in the System-Link response. It needs to be changed back before it is stored in the variable. To do this, the atof function is used.

Using the sample programs as a model

The sample programs are provided to you to use them as a model to create you own programs. You will need to do the following:

o   Create your own messages. There is an option on the Navigation menu to give you the System-Link request for whatever view or card file you have displayed. You can use the Simulator to adjust that System-Link request, or create your own, and test it. Once you have the System-Link request you want you can save it in a permanent message. Consider whether you want to perform requests in 'bunches' (i.e., using the FetchNext technique) in which case you need a pair of messages, one for the initial query and one for additional records. This makes good sense when you have some logic to determine whether additional records are needed. Alternatively, you can request all records with the initial request (maxReturned = -1). *Warning: Retrieving a large amount of information in one request may exceed the size of the LPI buffer…64K. Use it with caution.*

o   Build your request. Use the USRLPI0R program as a model. You may want to make the signon information parameters. Be sure the switch is set the way you want it (i.e., to call System-Link directly, if that's what you want to do). Check the *Build initial System-Link request statements* to be sure they reference your messages, parameters, etc.

o   Parse the results. There are two basic approaches to parsing XML data. One makes a pass through the XML and processes the desired fields one-by-one as they appear. Our sample programs use this single-step method. A second approach has two steps: it first makes a pass through the XML and simply places all fields encountered into a table. The desired field information is then extracted from the resulting table for processing in the second step. One advantage of this two-step technique is that you can write the XML parsing program (step one) once and reuse it in different programs requiring different field values. The disadvantage is that it can be inefficient, such as when you have a large number of fields (resulting in a large table) and only want the first one.

o   Process the response. Program USRLPI0R shows how the data can be parsed (by calling the parsing program) and printed as required. You may want to write the data to a file, or do something else with it.

Special notes on sample program USRLPI2R

This sample program is a simplified implementation of LPI using System-Link MQ. System-Link MQ uses WebSphere MQ as a communication medium between the caller (in this case the LPI program) and System-Link. USRLPI2R is based on a sample program provided by IBM that uses IBM's Message Queue Interface (MQI):

- Program name: AMQ3REQ4
- Source file name: QRPGLESRC
- Library name: QMQMSAMP

Using this program as a starting point, we have made some changes to conform to the specific requirements of System-Link MQ. The resulting program is USRLPI2R. The changes are summarized as follows:

- Queue name – The System-Link queue name is XSA.INPUT.QUEUE. This cannot be changed.

- Queue manager name – The System-Link queue manager name is COM.PSI.MAINQM. This cannot be changed.

- Work variables – We have provided as work variables the Unicode expressions for some of the System-Link tags.

- Parameters – We have defined the LPI parameters P#XREQ, P#XRSP, etc. and converted them to Unicode.

- Print to log – We have changed the printing technique to conform to the USRLPI0R program.

- Mainline – We have divided the four main sections into subroutines, then added the mainline program to call the subroutines. We did this to make the program easier to comprehend for the programmer.

- Evaluating the response – We added a section to evaluate the System-Link response to determine whether there are a) more records to query and b) any errors.

Other than these changes, the program should look very similar to the IBM sample program.

For further information on using System-Link MQ, please consult the next chapter in this Guide.

Chapter 4.   Accessing System-Link via System-Link MQ

System-Link MQ allows an outside program to send System-Link requests and receive System-Link responses via queues in WebSphere MQ (formerly MQ Series). There are three primary steps to using System-Link MQ:
1. Installation and configuration of WebSphere MQ, and IBM Directory Services,
2. Testing your System-Link MQ configuration, and
3. Designing and implementing the code in your program that interfaces with WebSphere MQ.

The first two steps are covered in detail in the *System-Link MQ Installation and Configuration Guide*. Please follow that Guide prior to continuing with step 3.

How you proceed with the third step – design and implementation – depends both upon how you intend to interface with MQ and upon the language in which your program will be written:
- o   If your program is written in Java, then you will be calling WebSphere MQ directly, using its JMS interface.
- o   If your program is written in RPG, then you can either use the LPI (via MQ) program provided, or can call WebSphere MQ directly using its MQI interface.

If you intend to use LPI via MQ as your program interface to System-Link MQ, then the actual interface to WebSphere MQ (as well as the System-Link MQ details) will be hidden from your program. Please consult the preceding chapter "Accessing System-Link via the Local Program Interface" for implementation details.

General Interface Program Flow

Whether you are using JMQ or MQI to interface with System-Link MQ, the basic overall steps will be the same:
1. Create a connection and a session with the queue manager / queue connection factory
2. Obtain a reference to the System-Link request queue
3. Create a new message structure for your request
4. Create a temporary / dynamic queue to receive your response, and place a reference to this in the message header
5. Insert your System-Link request into the message structure
6. Place the message on the System-Link request queue
7. Wait for a response message on the response queue that you created in step 4
8. Read your System-Link response from the response message
9. Delete your temporary / dynamic response queue (explicit for JMS, implicit for MQI)
10. End your session and connection

Calling System-Link MQ via JMS

JMS is a Java language standard interface supported by the major message queuing systems (including WebSphere MQ). General information, tutorials, and examples on using JMS can be found at http://java.sun.com, and are beyond the scope of this book. One of the downloadable sample programs, "eScriptJMSTester.java", demonstrates how to call System-Link MQ using JMS.

The JMS elements specific to System-Link MQ are as follows:
- o   Subtree from the base LDAP URL: "ou=mq, o=psi, c=us"
- o   Queue Connection Factory name: "cn=XSAQueueCF"
- o   System-Link Request Queue name: "cn=XSAQueue"

Calling System-Link MQ via MQI

MQI is an IBM WebSphere-MQ-specific interface, with support for RPG. General information on using MQI to interface with WebSphere MQ can be found at IBM's website, and is beyond the scope of this book. Due to the extensive nature of MQI, we recommend reviewing the downloadable sample program USRLPI2R. This sample program is a simple LPI implementation, and uses MQI to communicate with System-Link MQ. Further details on USRLPI2R can be found at the end of the preceding chapter.

The WebSphere MQ elements specific to System-Link MQ are as follows:
- o   Queue Manager name: "COM.PSI.MAINQM"
- o   System-Link Request Queue name: "XSA.INPUT.QUEUE"
- o   Model for Dynamic Response Queue: "SYSTEM.DEFAULT.MODEL.QUEUE" (under COM.PSI.MAINQM)

Chapter 5. System-Link Response Forwarding

With System-Link Response Forwarding, one can send System-Link responses to recipient programs that did not explicitly request them. (In effect, this emulates a "push-based" system: there still needs to be a request sent to System-Link that initiates this response.)

Aside from the creation of the actual recipient programs (the details of which will vary from installation to installation), one must make System-Link aware of the external recipient details, as well as identify when this recipient is to receive a copy of a given response. (The latter is specified as part of the System-Link request. Please see **Response Forwarding Commands** in **Chapter 10.** for further details.) The former configuration task is handled through System-Link business objects that are accessible via the XA card on the main browser.

System-Link Destinations

For environments on which System-Link is installed, the Settings tab on the Environment card will contain a "System-Link Destinations" object:
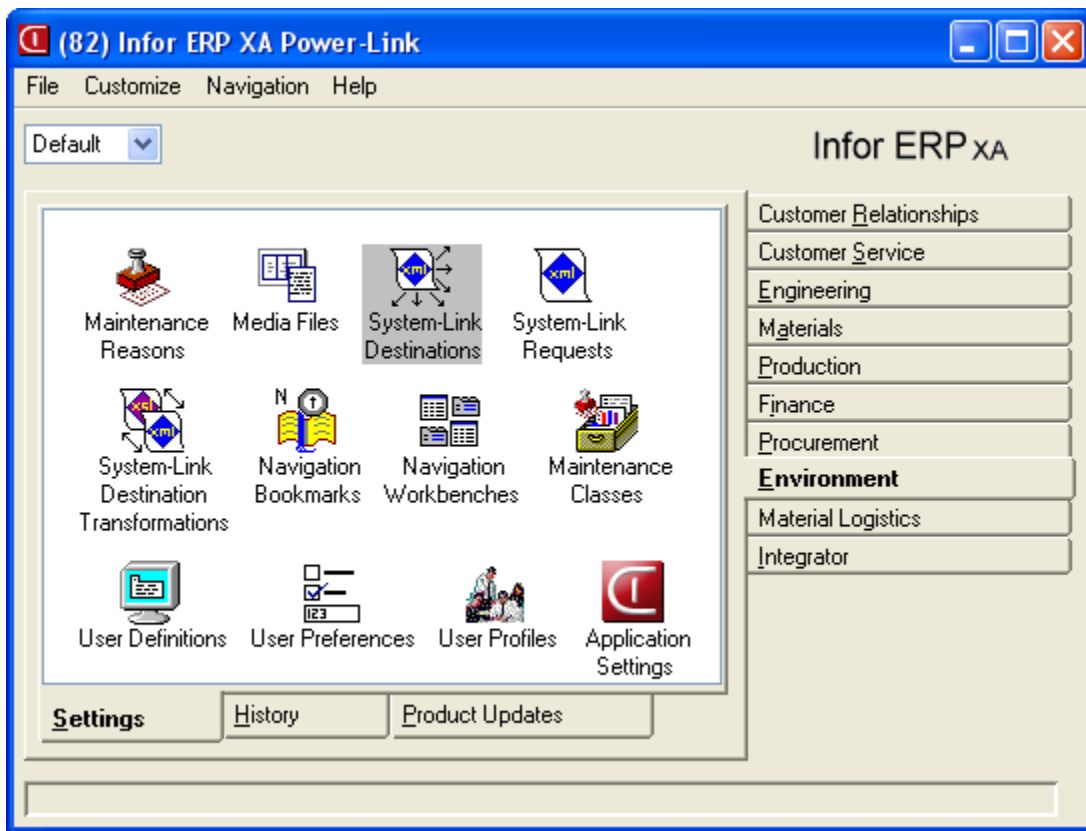
As with other business objects, you can get a list browser and an object browser over the System-Link Destination objects. (Note: the "checkmark" action icon on the right: this is the icon for the "Test Recipient" action, which will be covered shortly.)
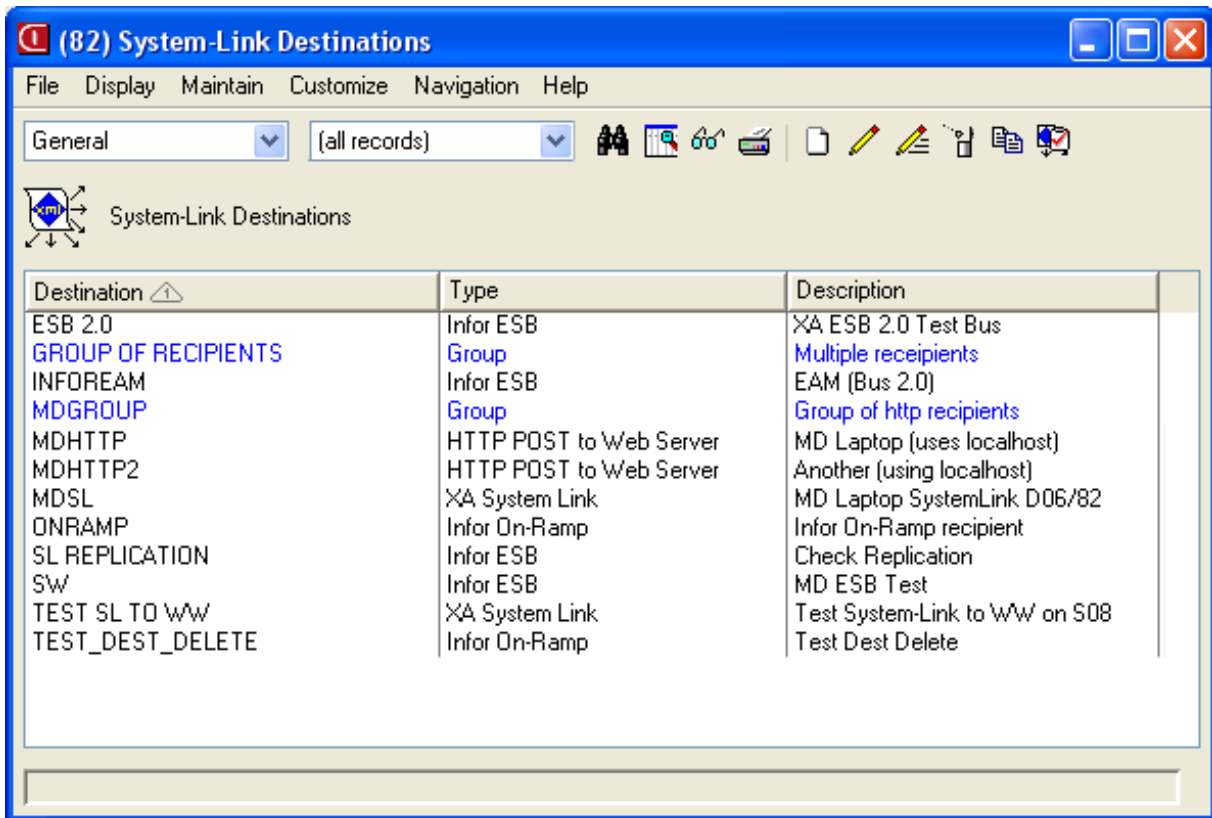
At the present time, there are these different types of destinations (an example of each shown above):

- "Infor ESB": the recipient system is listening on the Infor Enterprise Services Bus.
- "Infor On-Ramp": the recipient system is listening on the Infor On-Ramp.
- "XA System-Link": the recipient program is listening on another XA System-Link server.
- "Group": the destination consists of multiple (non-group) recipients.
- "HTTP": the recipient program receives the response via an HTTP POST operation.
- "LPI": the recipient RPG program is programmatically "bound" to the SLC process for that environment, and listening for an upcall.[3]
- "MQ": the recipient program is listening on a specified WebSphere MQ queue.
- "Socket": the recipient program is listening on a specified socket on a specified host.

The three properties shown in the above list browser are those common to all destination types. (The purposes for the "type" and "description" properties are fairly obvious.) The value for the "Recipient ID" property is used as a "key" in the System-Link request document to refer to a given destination. In addition, each destination type shows the System-Link Destination Transformations and the System-Link Transformations available for that destination type and request. Please see the "**System-Link Destination Transformations**" and "**System-Link Transformations**" sections in this chapter for further details.

---

[3] The recipient-specific LPI implementation is currently under development. Details on this interface will be given in a subsequent version of the System-Link User's Guide.

The following are the object browsers for each type of destination, showing the properties that are specific to each:
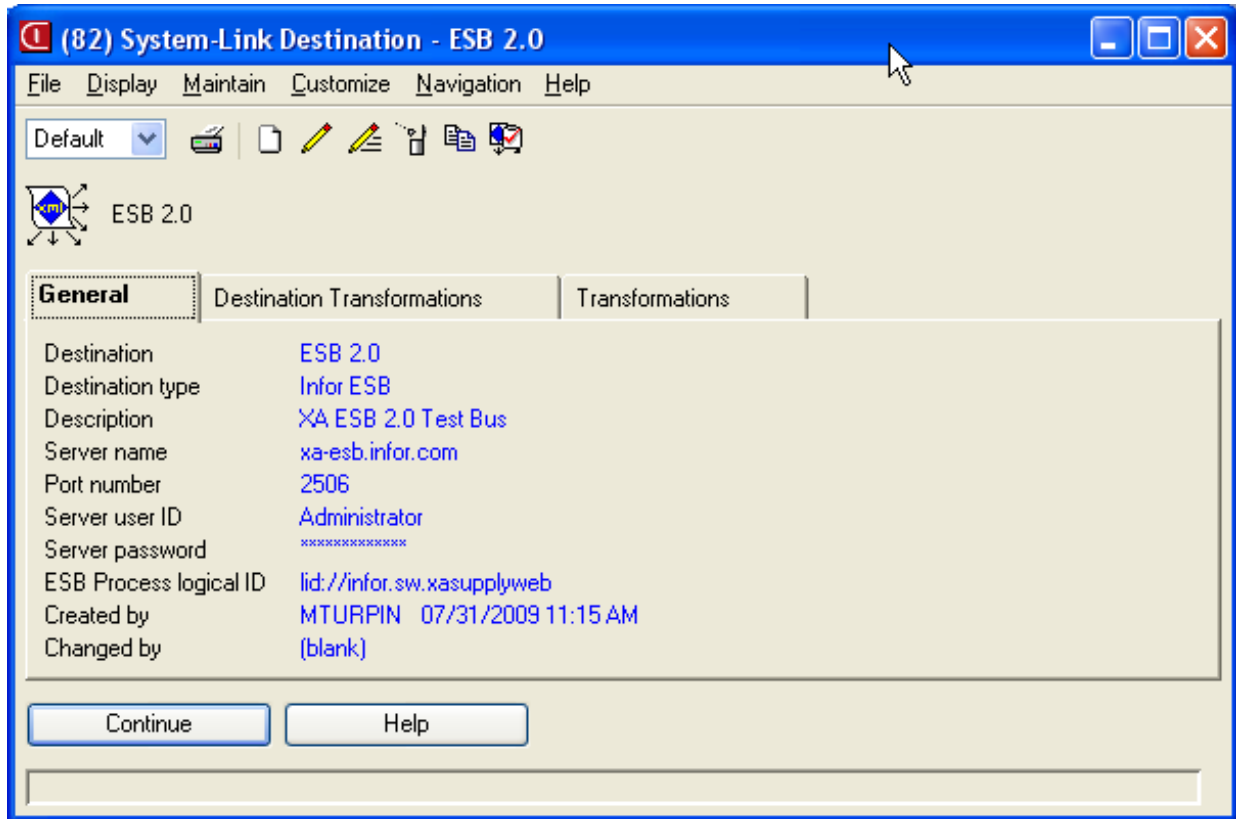
Server name: the name of the machine running the Infor ESB server
Port number: the listener port number configured for the Infor ESB server instance
Server user ID: the user ID used to connect to the Infor ESB server (not necessarily the same as used to access the server machine)
Server password: the password associated with the user ID above
ESB Process Logical ID: the LID used during a Process BOD sent from this destination

More details about connecting XA to the Infor ESB are covered in the "Integrating XA & SupplyWEB" integration guide.
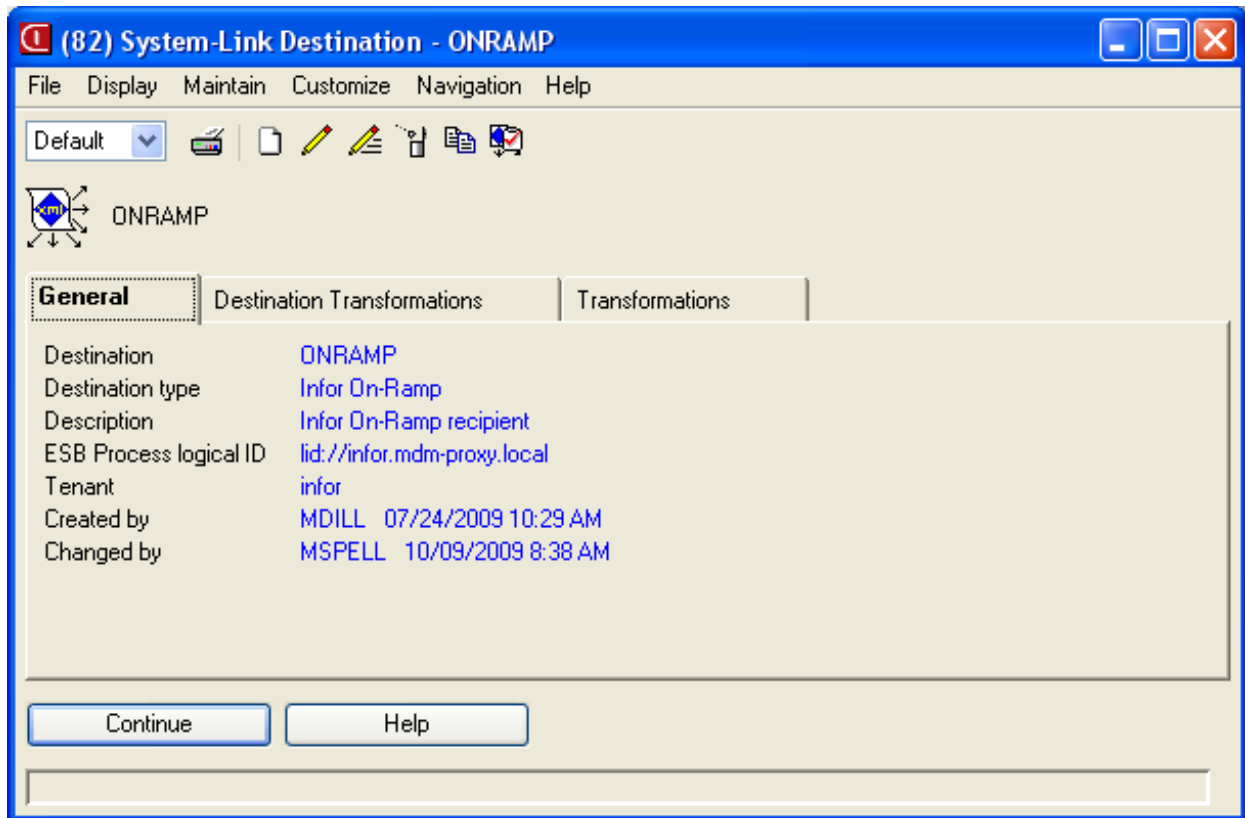
ESB Process Logical ID: the name of the component or subscriber that will handle the transaction transmitted through the Infor On-Ramp.
Tenant: the name of the instance of the database supporting the Infor On-Ramp.
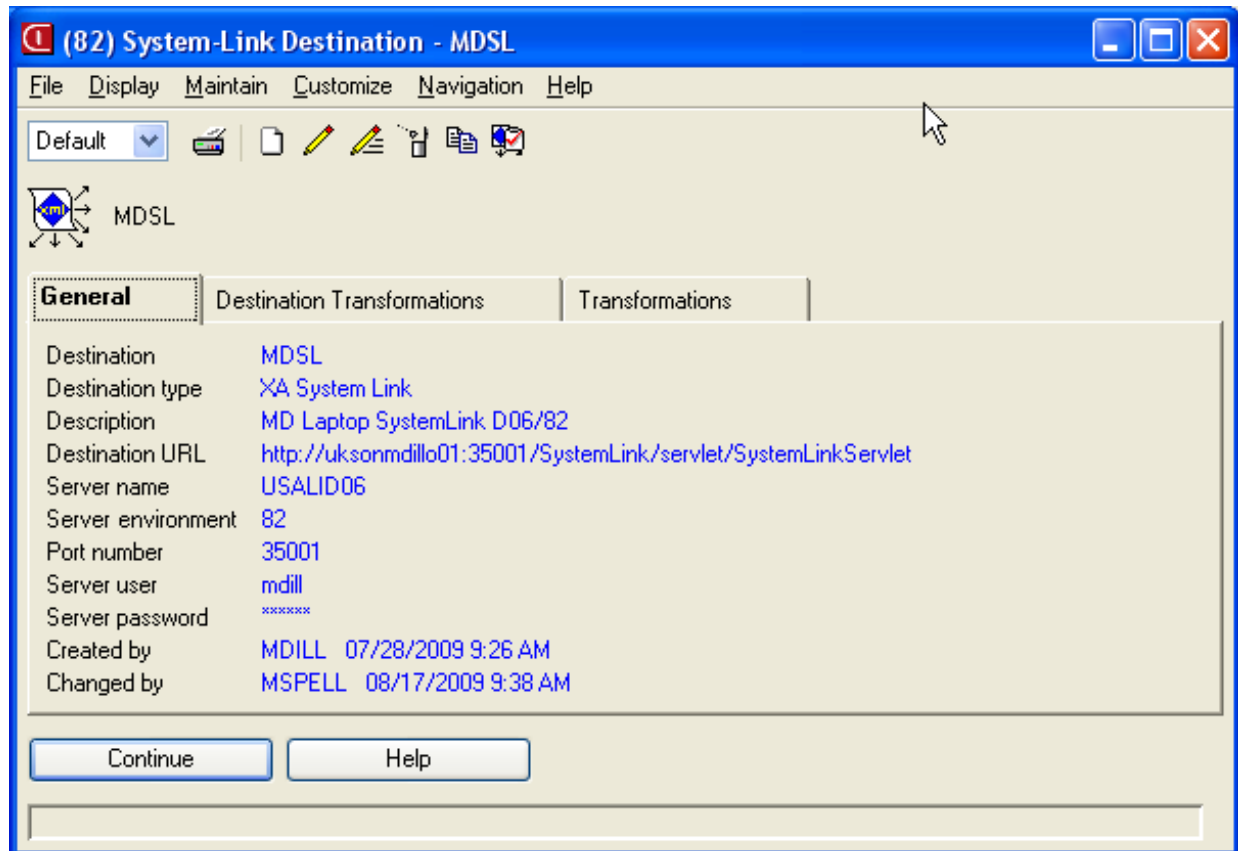
<u>Destination URL</u>: the URL of the HTTP server expecting the response.

<u>Server name</u>: the name of the machine running the XA recipient server.

<u>Server environment</u>: the identifier for the recipient XA environment.

<u>Port number</u>: the listener port number configured for the XA recipient server instance.

<u>Server user</u>: the user ID used to connect to the XA recipient server (not necessarily the same as used to access the server machine).

<u>Server password</u>: the password associated with the user ID above.
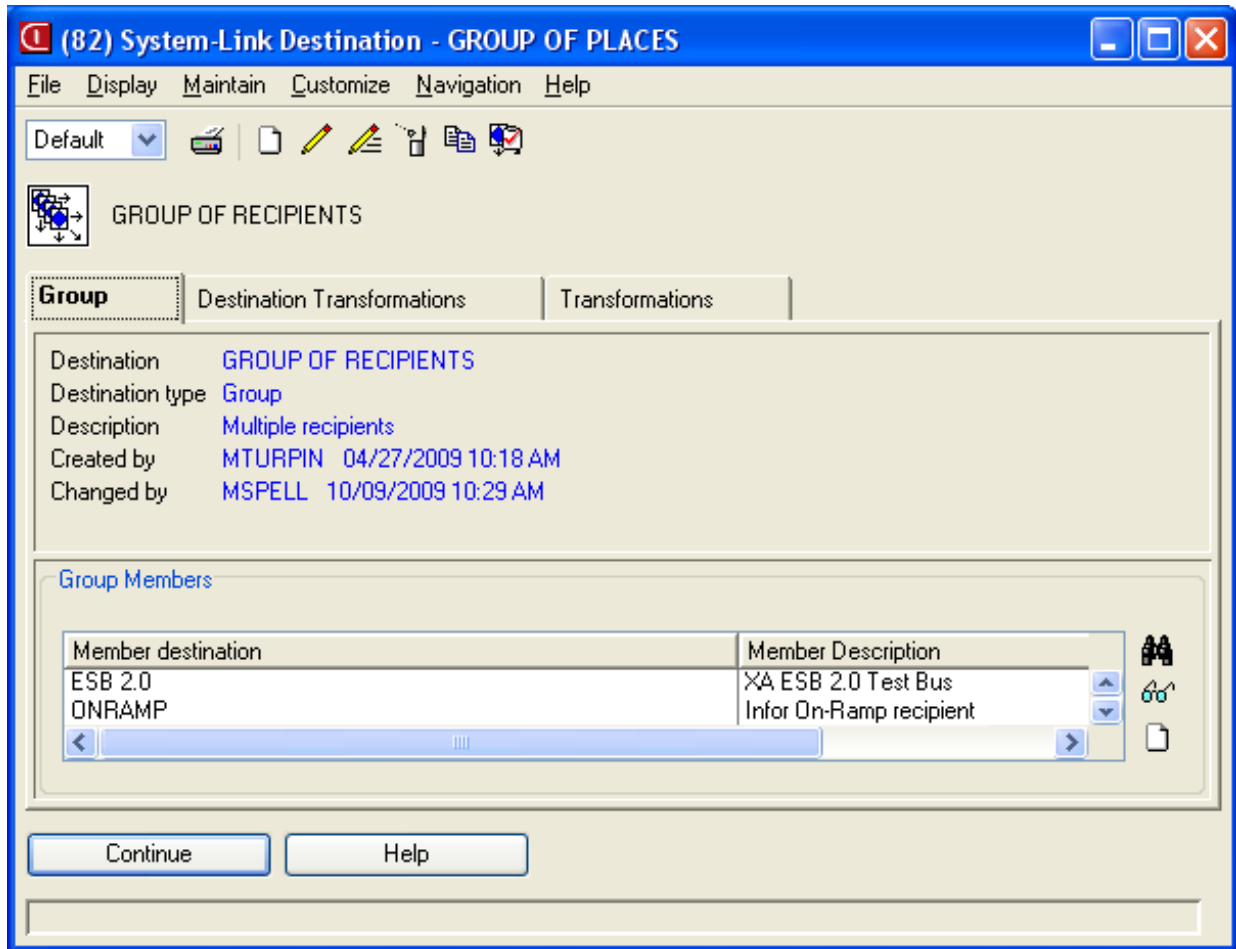
The Group destination type contains an editable list of member destinations. These members can be of any destination type except for Group.
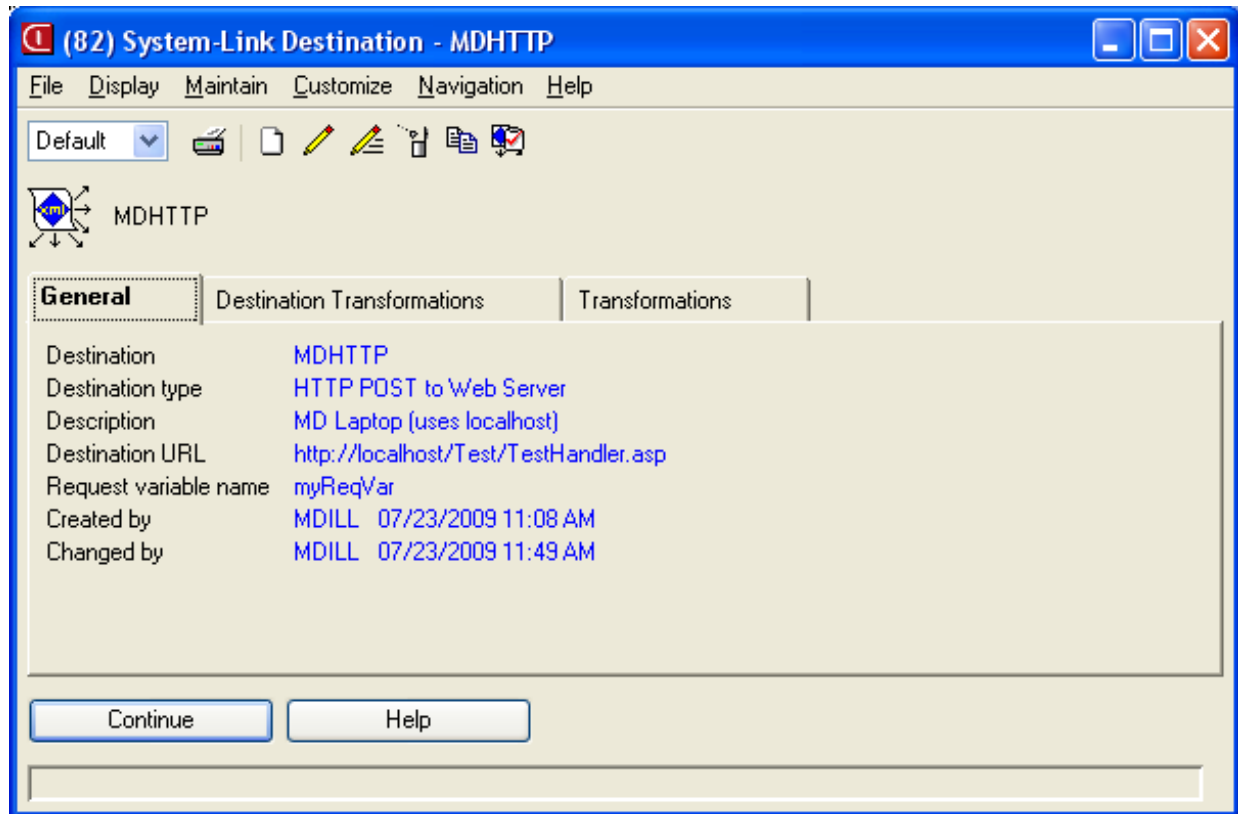
Destination URL: the URL of the HTTP server expecting the response
Request variable name: (optional) if the server is expecting the response to be the value for a variable in the POST body (as opposed to simply being the body itself), this variable name is specified here
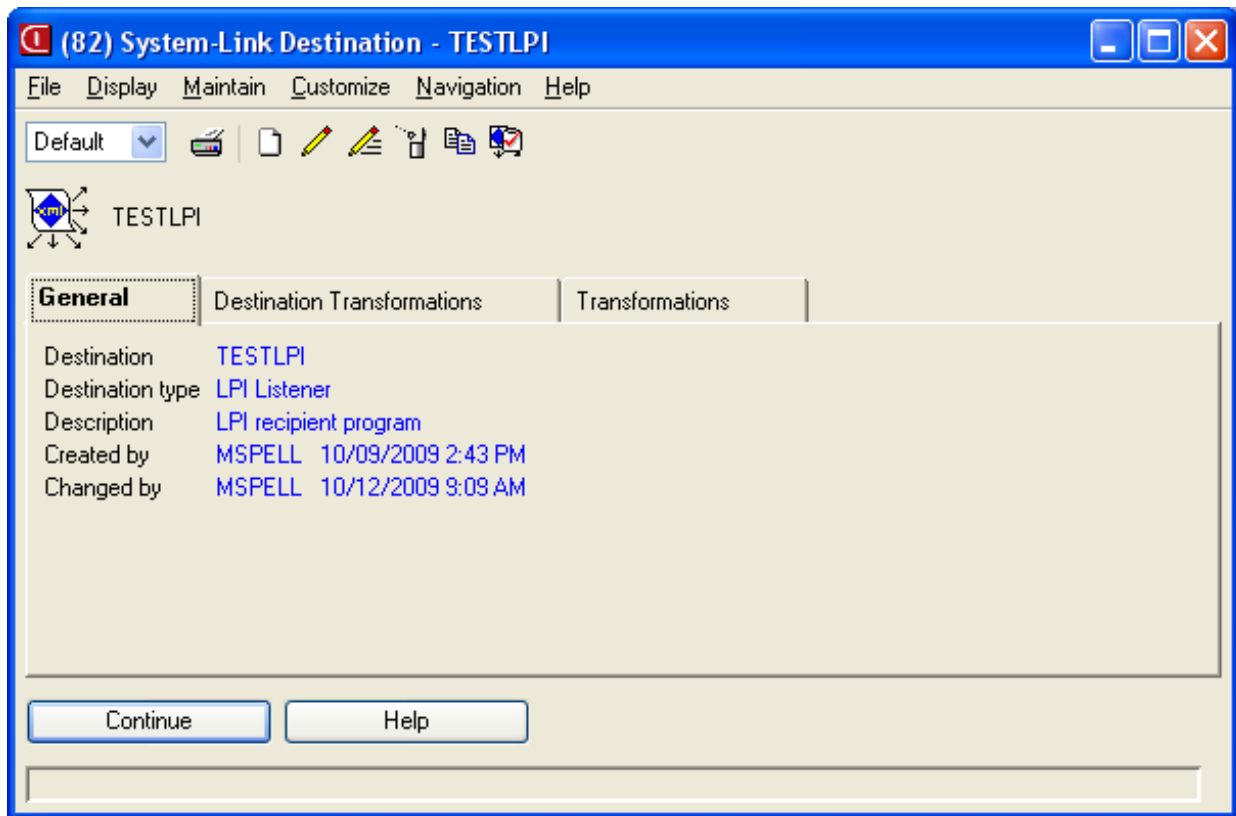
Note that, as an interface that is totally encapsulated within System-Link, there are no LPI-specific properties required.
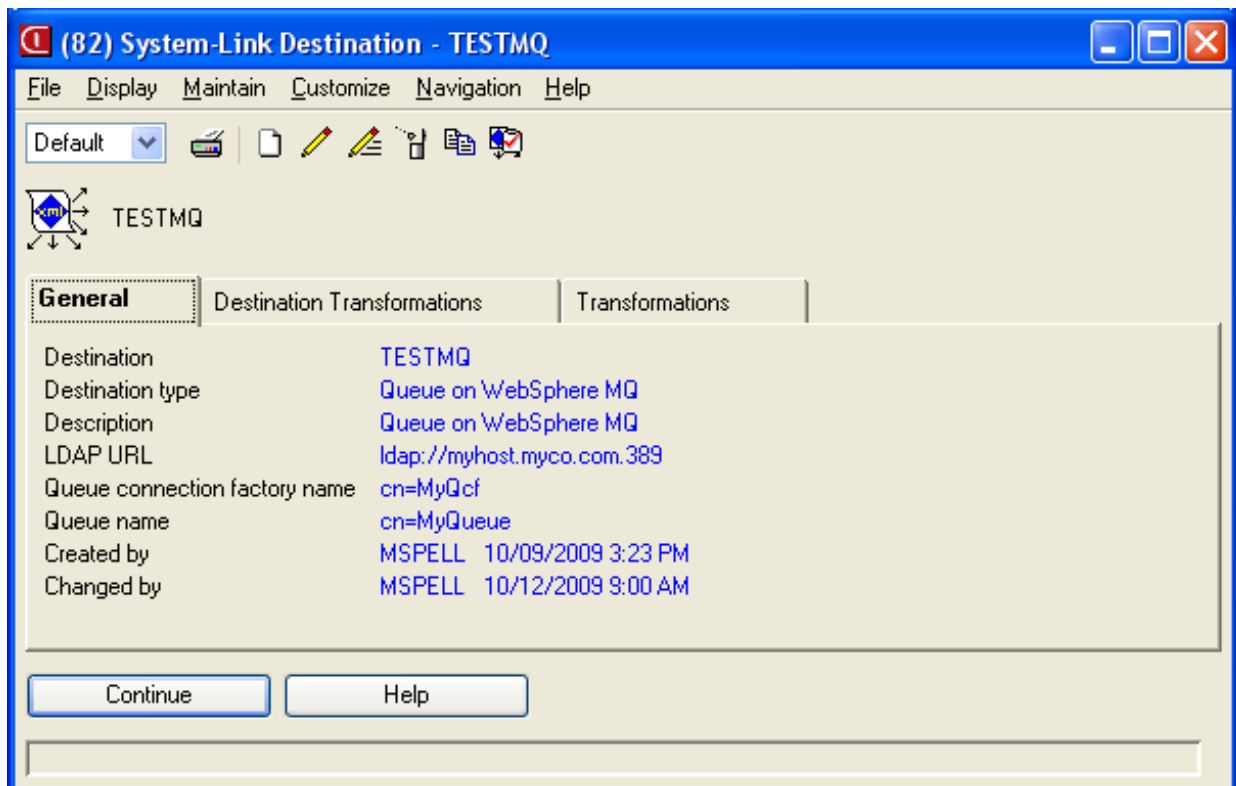
<u>LDAP URL</u>: the URL of the LDAP server on which the queue connection factory and queue are bound.

Queue connection factory name: name of the queue connection factory
Queue name: name of the queue on which the response should be placed
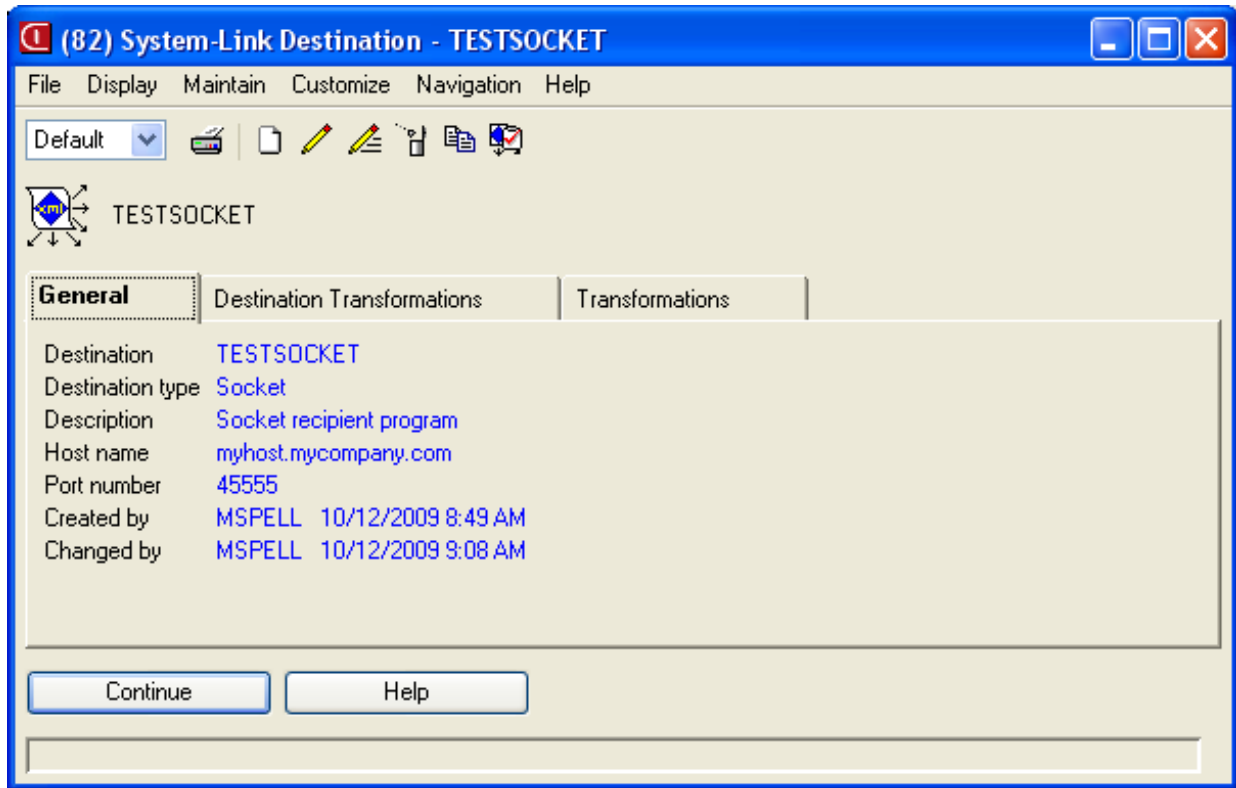
Host name: name of the recipient's host machine
Port number: number of the port on which the recipient is listening

System-Link Requests

For environments on which System-Link is installed, the Settings tab on the Environment card also contains a "System-Link Requests" object. A System-Link request provides information required to form requests to send information to another system. Each System-Link request contains descriptive information about the request, control information for how to process the request, audit controls to log the request, and the original and overridden versions of the request XML.

XA provides a default request for each transaction supported by a BOD. Users can use or modify the XA requests or create additional requests. The Request ID for XA-provided requests begins with "XA_" and contains a verb and noun indicating the type of request.

The list browser over the System-Link Requests object shows the request ID, description, and whether the body of the XML for the request can be overridden.

Request IDs for user-defined requests cannot begin with XA. Also, XA reserves the request ID of *INBOUND for a predefined request.

The object browser shows properties specific to a selected request. The General card contains processing information for the request and a list of request transformations defined for the request. This card combines information from files SLRHDR, SLRTXT, SLRUXT, and SLRTRF.
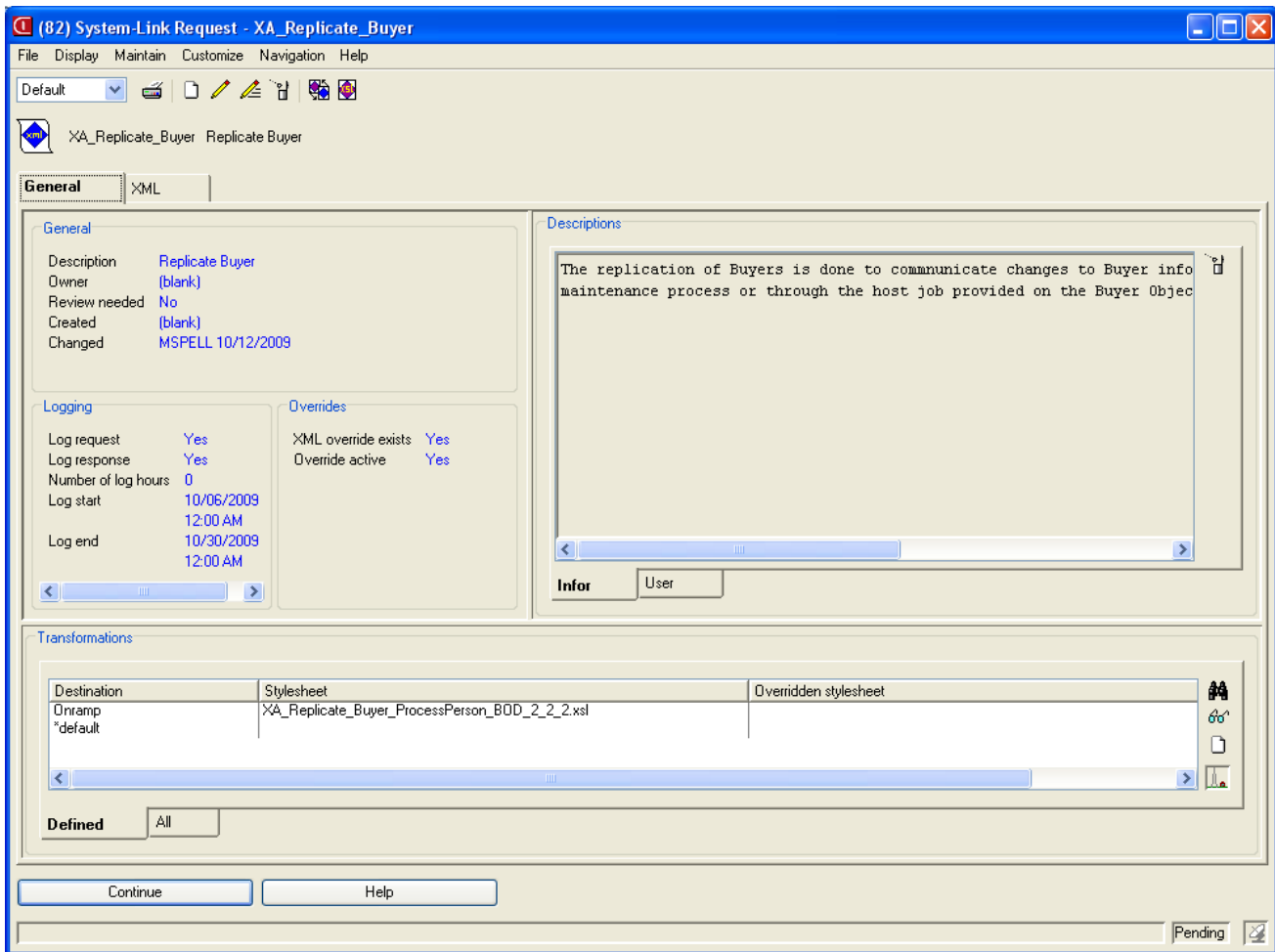
Owner: the ID of the user who has responsibility for the request. The owner cannot be changed after the request is created. XA sets the owner for all requests provided by XA to (blank).

Review needed: Yes indicates that the request requires review. XA sets this attribute to Yes automatically for any new versions of an XA-supplied request.

Log request: indicates whether transmission of the request is logged in the Transaction Status object.

Log response: indicates whether the response to the request is logged in the Transaction Status object.

Number of log hours: the number of hours for the duration of logging. This time, combined with the Log start date and time, can be used to determine the log end date and time automatically.

Log start: the day and time for the log activity to begin.

Log end: the day and time for the log activity to end. If specified, the Number of log hours can be used with the Log start values to calculate the Log end day and time automatically.

XML override exists: Yes indicates overridden XML content on the XML tab is active or ready for use. The default value is No. This attribute is available if user-entered XML content exists for this request.

Override active: Yes indicates the overridden XML is used for the body of the request instead of the original XML. XA automatically sets this attribute to Yes if override XML is entered for the request.

The Descriptions section contains the text from the SLRTXT and SLRUTX files.

The Transformations section lists all System-Link destination transformations which apply to this request. The related transformations come from the SLRTRF file.

The XML card shows the request XML provided by XA from the XML Requests file (SLRXML) and an area for entry of custom XML that overrides the predefined XML for the body of the request. Any custom XML entered is stored in the User Overridden XML Requests file (SLSUXM). For XML requests provided by XA, the Request XML Body can be overridden by XML entered in the Overridden XML Body. The overridden Request XML Body does not replace the original Request XML Body but is used in the request transaction if the XML override exists and Override active attributes are set to Yes.
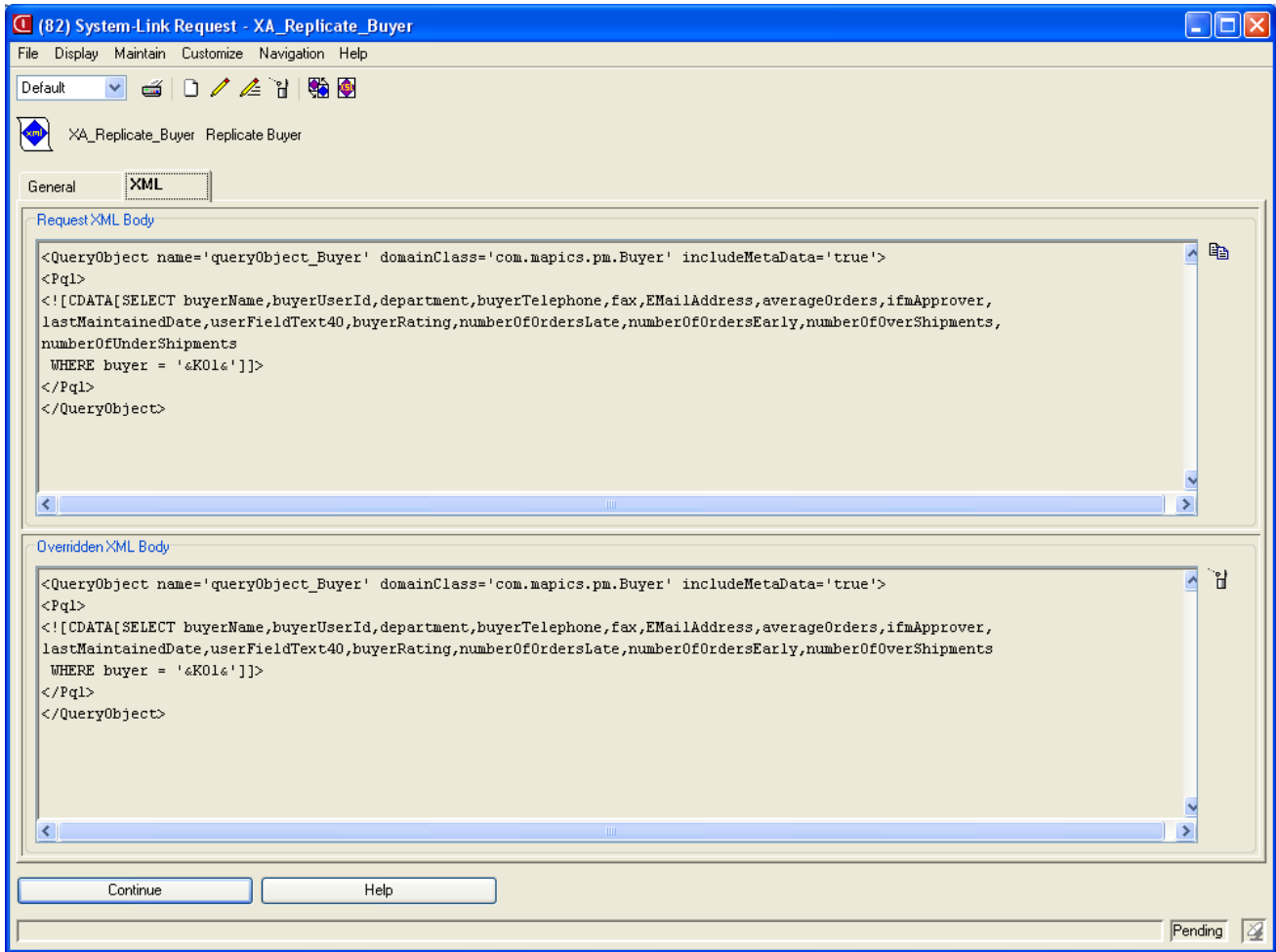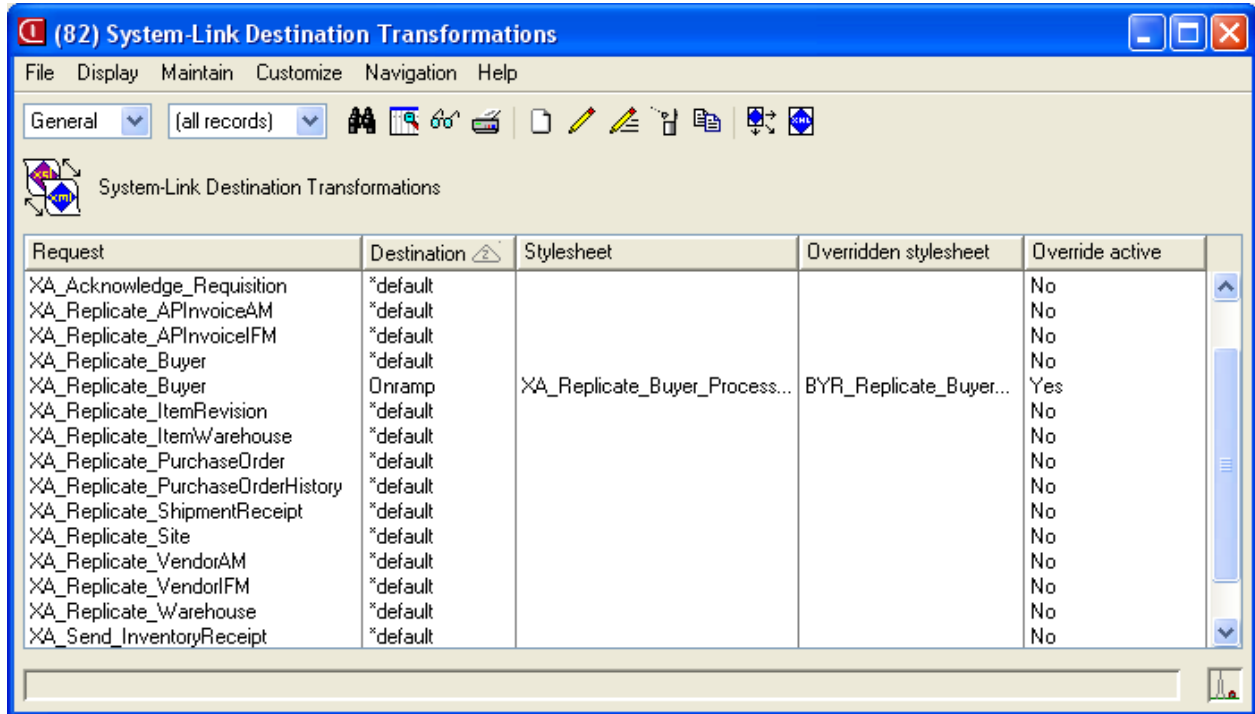
XA provides the XML for the login, header, and footer sections of the System-Link request and for testing the request. The XML in these sections of the request cannot be changed. Users can create System-Link requests, but the XML for the login, header, footer, and test sections defaults to the XA-provided XML and cannot be changed. Users provide the XML for the body of the non-XA requests.

System-Link Destination Transformations

For each combination of System-Link destination and request, a "System-Link Destination Transformation" object must exist. A System-Link destination transformation establishes the link between the System-Link request and the System-Link destination and specifies the stylesheet to be applied to transform the request for the particular destination.

The list browser over the System-Link Destination Transformations object shows the request ID, destination for the request, and the stylesheet for transforming the request. The list also indicates if the transformation includes an override stylesheet and if the override is active.

The object browser shows properties specific to a selected request and destination. The General card identifies the request, the destination, the applicable stylesheet, and details regarding transformation processing.
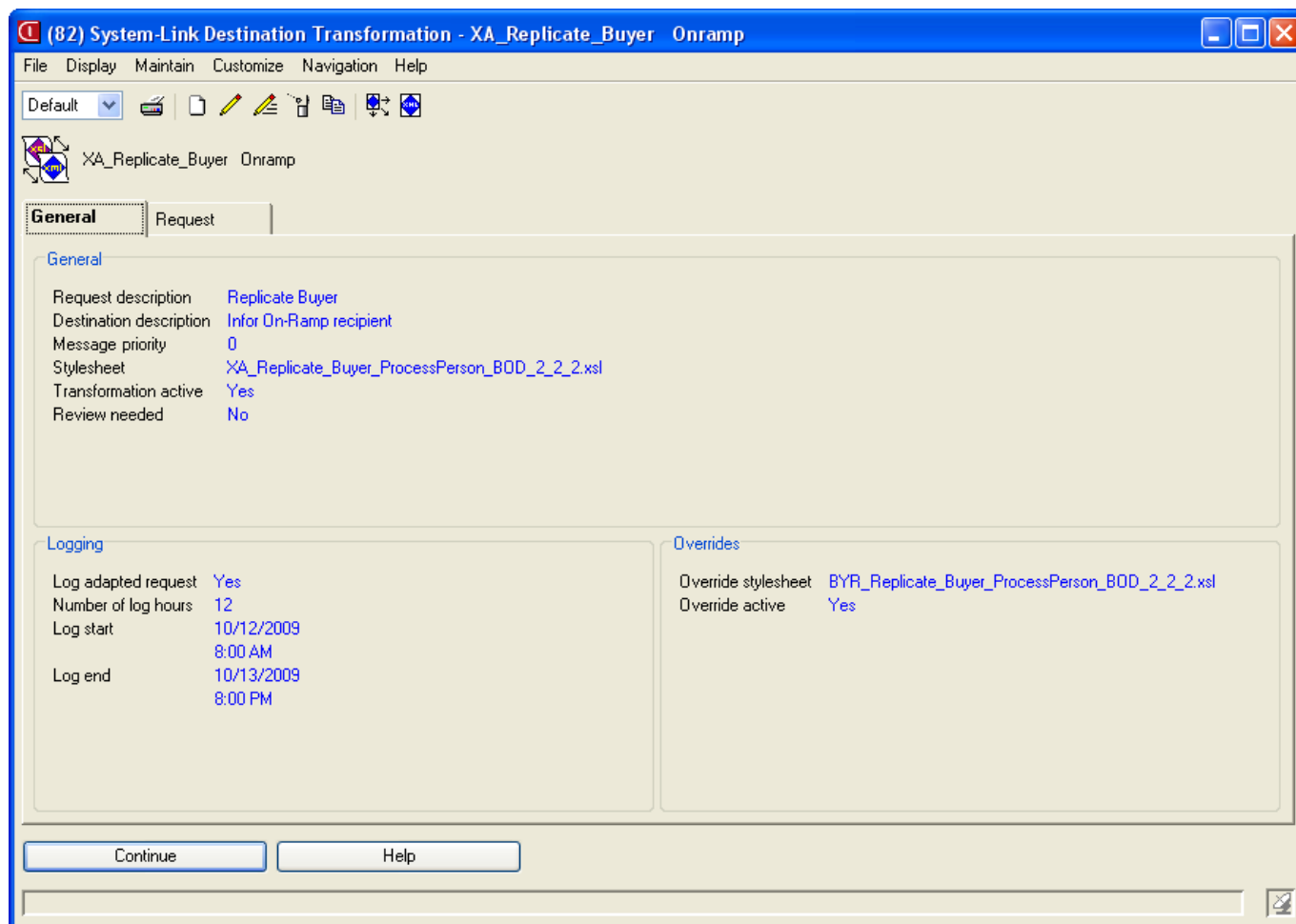
Request description: the description for the request to be processed by the transformation. This information is required for an inbound request.

Destination description: the description for the destination to be processed by the transformation.

Message priority: the level of importance for processing the transformed output by the remote system.

Stylesheet: the file name of the default stylesheet assigned to the transformation. The stylesheet can be overridden. In Create or Change mode, use the Verify Stylesheet button to check that an override stylesheet file name is a valid file in the IFS directory Infor/AMALIBz/AMFLIBx/UserTransformations. Any user-defined stylesheets must be stored in this IFS location to be used to transform a request.

Transformation active: Yes indicates the request is available for delivery to the specified destination.

Review needed: Yes indicates that the transformation requires review. XA sets this attribute to Yes automatically for any new versions of an XA-supplied transformation.

Log adapted request: indicator for whether the response to the request is logged in the Transaction Status object.

Number of log hours: the number of hours for the duration of logging. This time, combined with the Log start date and time, can be used to determine the log end date and time automatically.

Log start day and time for the log activity to begin.

Log end: the day and time for the log activity to end. If specified, the Number of log hours can be used with the Log start values to calculate the Log end day and time automatically.

Override stylesheet: the stylesheet to be used as an alternate for transforming the request when delivered to the destination.

Override active: Yes indicates the overridden stylesheet is to be used to transform the request for this destination.

The Request card provides access to the XML that comprises the message.
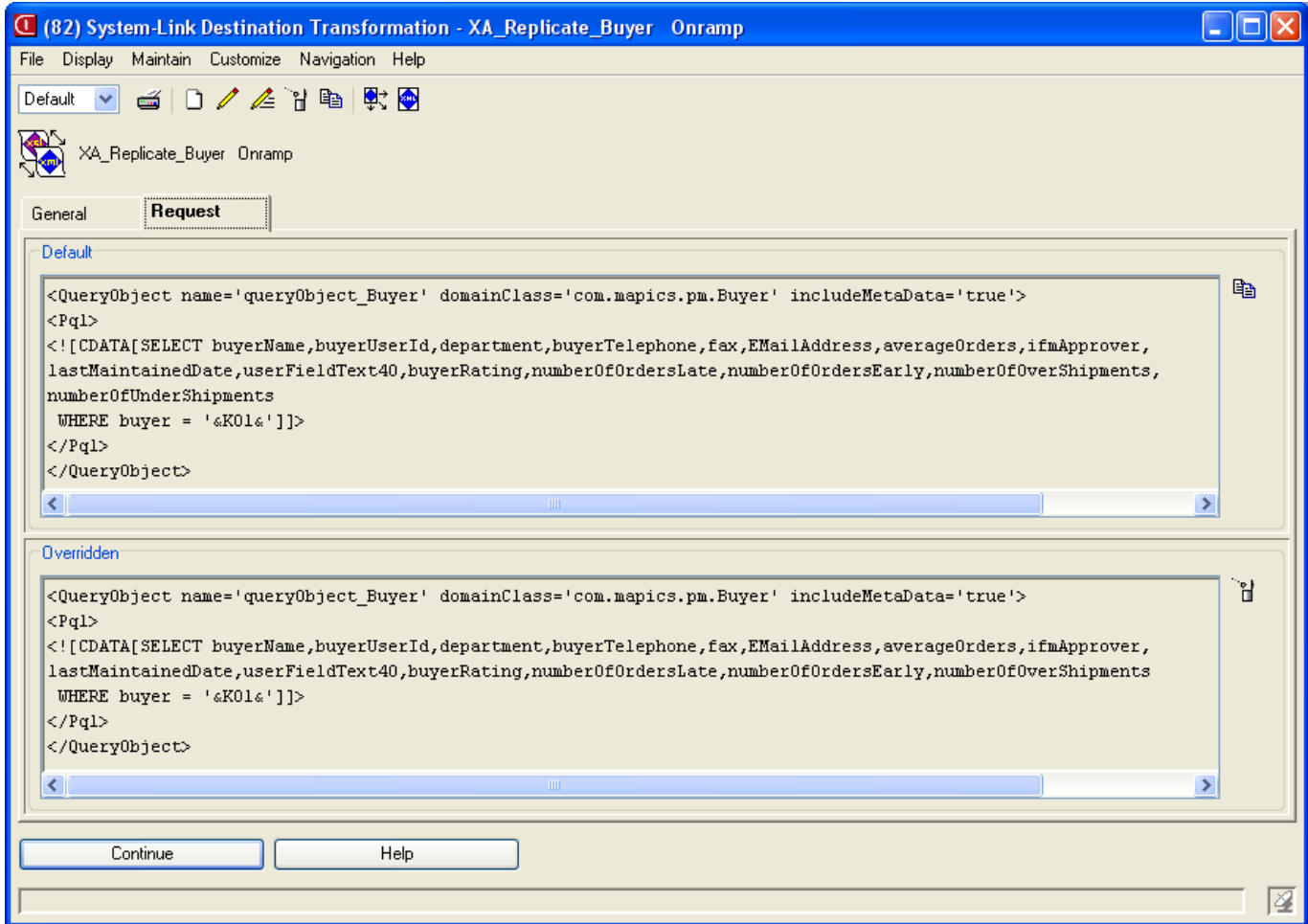
System-Link Transformations

For each XA-provided System-Link request, a "System-Link Transformation" object must exist. The System-Link transformation contains default options for processing, or transforming, the request into a BOD. These processing options apply to any BODs created from an XA-provided request for the destinations specified for that request.

XA supplies the System-Link transformations for use only with XA-provided System-Link requests. System-Link transformations do not apply to user-defined System-Link requests.

Transformations cannot be maintained, but they can be displayed from the Transformations card in the detailed information for a System-Link destination or from the System-Link Transformations option on the Display menu in the System-Link Destinations object.

The list browser over the System-Link Transformations card shows the request ID and the default stylesheet for transforming the request.
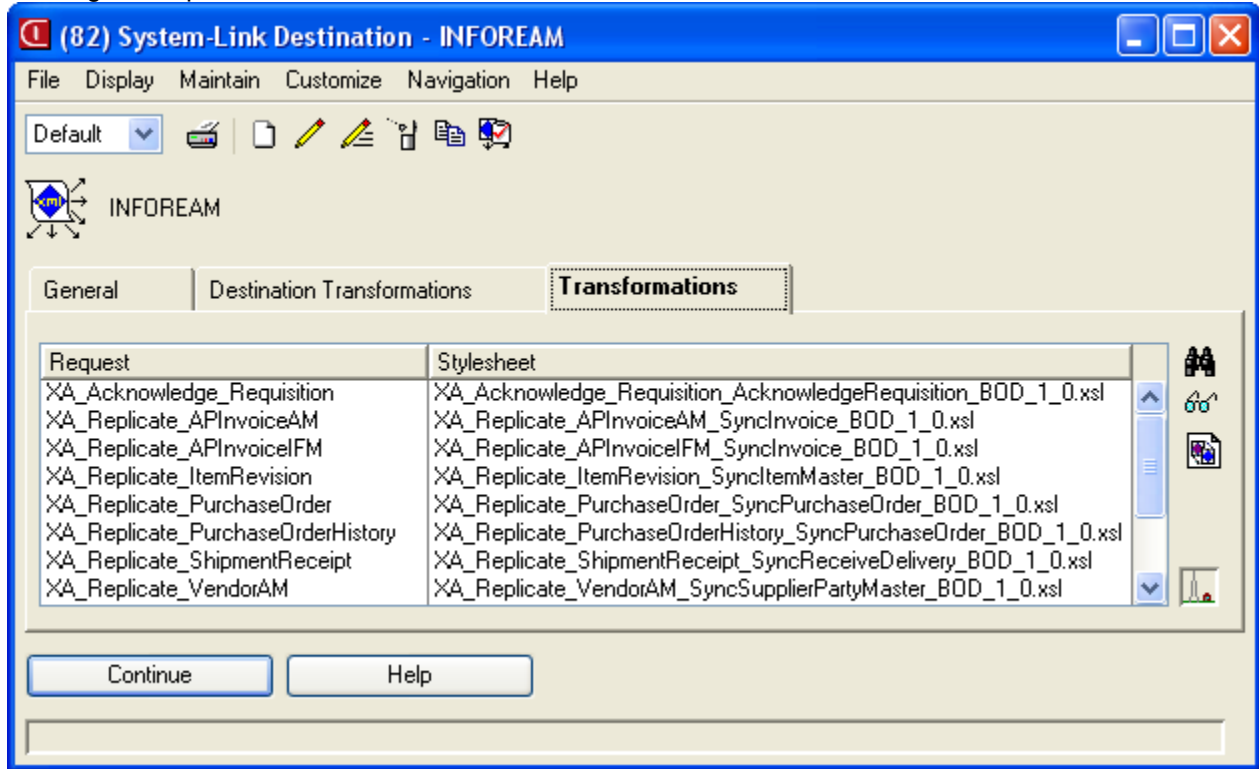
The object browser shows properties specific to a selected request and destination. The General card identifies the request, the destination, the applicable stylesheet, and details regarding processing of the transformation.
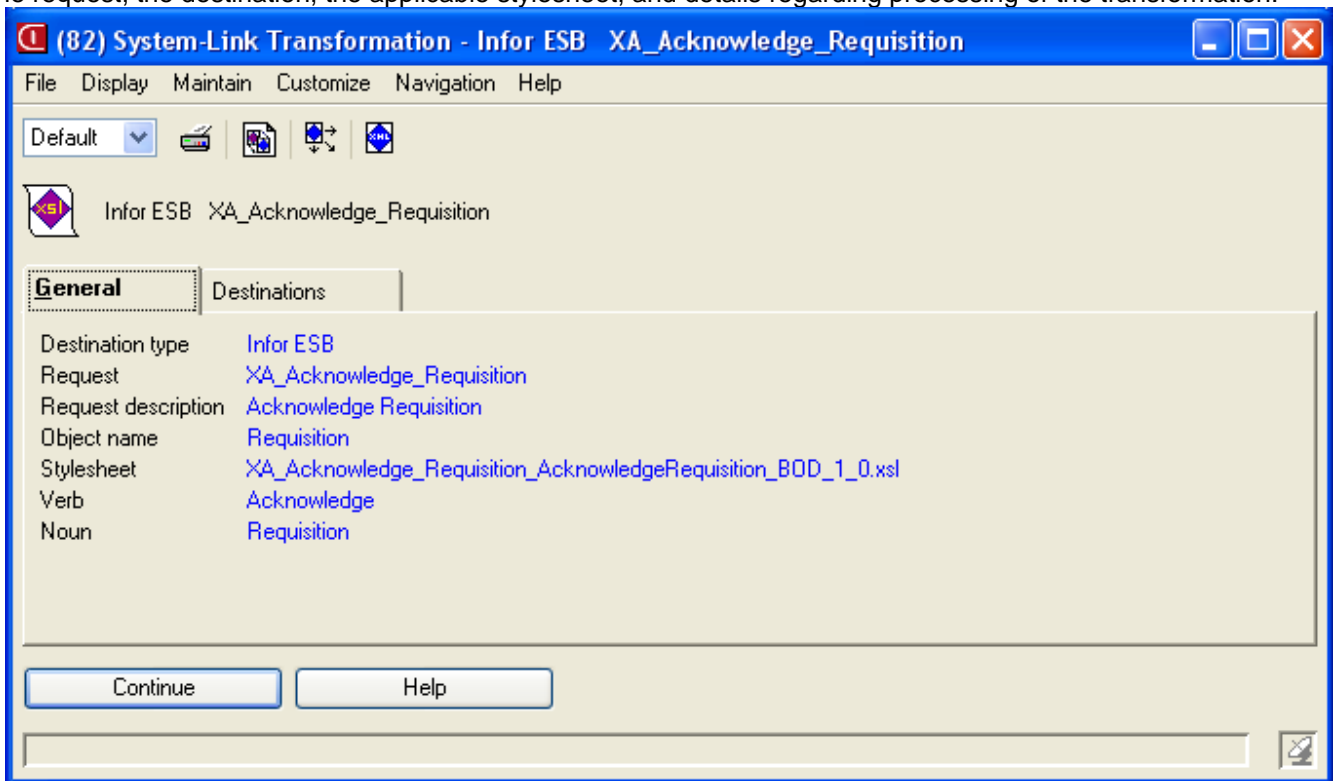
Destination type: the category that classifies the destination used in this transformation. This attribute contains the same categories as the Recipient type attribute specified by System-Link recipients.
Request: the identifier for the request to be processed by the transformation.

Request description: the text specifying the purpose of the request.
Object name: the identifier of the main XA application object from which information is sent or to which information is received.
Stylesheet: the identifier of the default stylesheet assigned to the transformation. The stylesheet can be overridden in the System-Link destination transformation.
Verb: the action performed by the XML request. The verb is for reference purposes and applies only to transformations to Infor BODs.
Noun: the data being acted upon by the XML request. The noun is for reference purposes and applies only to transformations to Infor BODs.

The Destinations card lists the destinations for the selected request that are processed by the transformation.
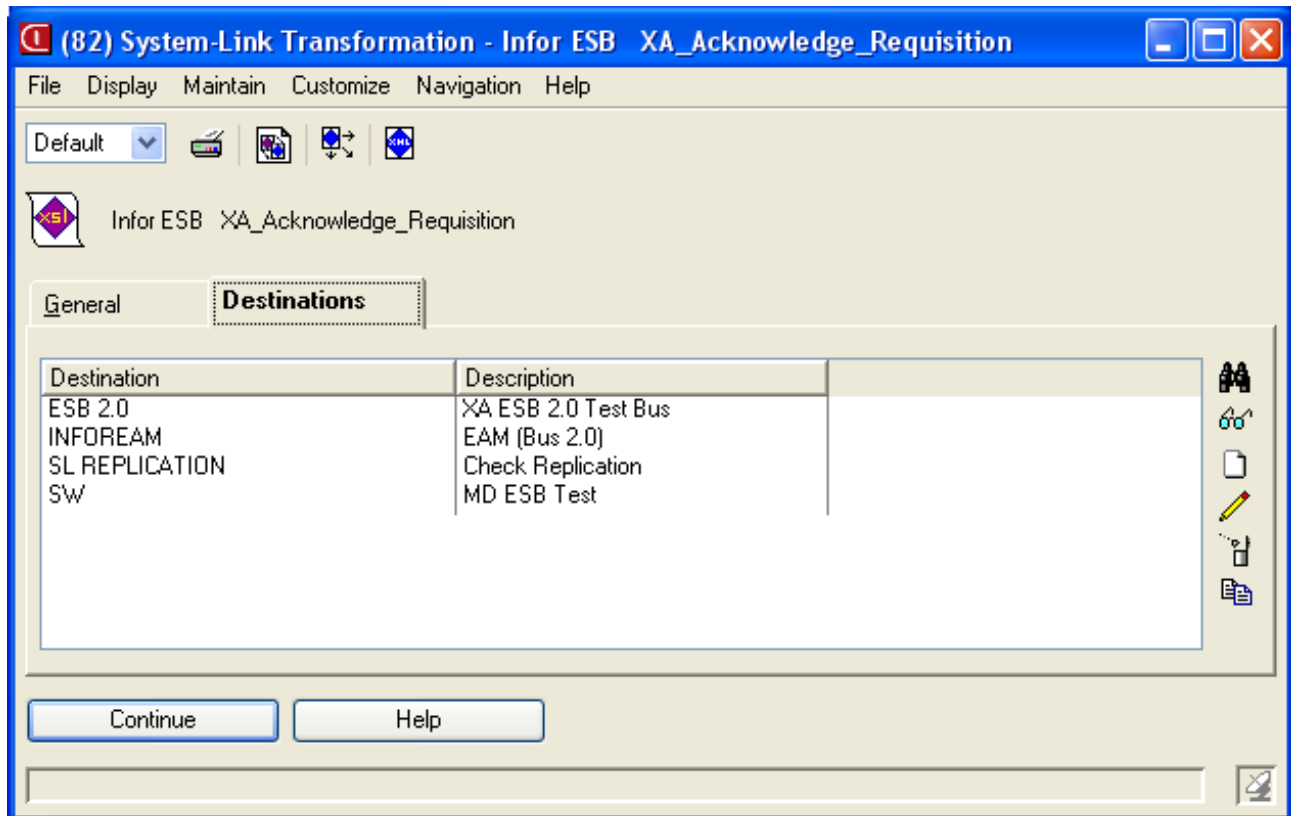
Transformation Mechanism

Transformations are performed once per stylesheet against the generated System-Link response. System-Link processes the transformation a single time for each stylesheet because the transformed document could be sent to multiple destinations. The document sent to each destination should contain information specific to that destination. The transformed document varies for each destination due to substitution variables that differ among destinations. These variables are primarily, but not exclusively, obtained from the settings specified for the destination. The variable values can be inserted into the transformed document by specifying the substitution variable in the stylesheet and are delimited by the "%" character. Not all destination types use substitution variables. The different requirements for substitution variables used with different destination types are described in the following sections. For example, the following code specifies the Session handle value for a System-Link destination of type HTTP.

<Request sessionHandle="%SESSIONHANDLE%" workHandle+"*new" broker="EJB" maxIdle="1000">

Additionally, some of these substitution variables might require data from the generated document (for example, the bodId on Infor Bus 2.0 and Infor On-Ramp). To provide this data, reference values should be added in the stylesheet. The reference values can be added by specifying a comment in the stylesheet that contains the text TX_REF:{Reference}={Value}. Again, not all destination types rely on this mechanism. The following example specifies the ReferencID value used when building the BOD_ID substitution variable for Infor Bus 2.0 destinations.

Note that the entire example should be entered on a single line, but the line is wrapped here due to formatting limitations in this document.

<xsl:comment>TX_REF:ReferenceID=<xsl:value-of select="System-Link/Response/QueryObjectResponse/DomainEntity/Key/Property[@path='order']>/Value"/></xsl:comment>

System-Link HTTP post

A destination of this type performs the login and session maintenance automatically using the values specified in the System-Link Destination details. When creating the stylesheet for this type of destination, it is necessary to assume that the login has been performed. Therefore, all tags that require the sessionHandle attribute should specify this value using the SESSIONHANDLE substitution variable.

Substitution variables

SESSIONHANDLE	The session handle obtained from the login. Use this value wherever the sessionHandle attribute is required.

Transformation references

There are no transformation references.

Infor Bus 2.0

Connection to the Bus will be performed automatically. The substitution variables should only be used inside the ApplicationArea section because only this area can differ between destinations. These details will be obtained from the Bus properties table and from the destination details.

Substitution variables

TO_LOGICAL_ID	The logical ID of the destination. For Process or Get verbs, this value will be specified in the destination. For all other verbs, this value will be lid://default.
FROM_LOGICAL_ID	The logical ID of the sender. This value will be obtained from the Bus properties.
CREATION_DATE_TIME	The current time as a formatted timestamp in GMT.
BOD_ID	The generated BOD ID has a pre-generated format which includes the value obtained by referring to the transformation reference "ReferenceID."
REFERENCE_ID	Currently the same value as BOD_ID.

Transformation references

ReferenceID	The format of the reference ID used to generate the BOD ID is related to the key of the business object. This transformation reference is used to specify the value to be used.

Example

The following example is a fragment of the stylesheet used to send a Purchase Order with the Process verb. The ApplicationArea of the example should be used for most Infor BOD level 1.0 transformations..

```
<ProcessPurchaseOrder … >
  <xsl:comment>TX_REF:ReferenceID=<xsl:value-of select="System-
  Link/Response/QueryObjectResponse/DomainEntity/Key/Property[@path='order']/Value"/>
  </xsl:comment>
  <ApplicationArea>
    <Sender>
      <LogicalID>%FROM_LOGICAL_ID%</LogicalID>
      <ReferenceID>%REFERENCE_ID%</ReferenceID>
    </Sender>
    <CreationDateTime>%CREATION_DATE_TIME%</CreationDateTime>
    <BODID>%BOD_ID%</BODID>
  </ApplicationArea>
```

Infor On-Ramp

The substitution variables should only be used inside the ApplicationArea section and verb sections because only these areas can differ between destinations. These details will be obtained from the Bus properties table and from the destination details.

Substitution variables

TO_LOGICAL_ID	The logical ID of the destination. For Process or Get verbs, this value will be specified in the destination. For all other verbs, this value will be lid://default.
FROM_LOGICAL_ID	The logical ID of the sender. This value will be obtained from the Bus properties.
CREATION_DATE_TIME	The current time as a formatted timestamp in GMT.
BOD_ID	The generated BOD ID has a pre-generated format which includes the value obtained by referring to the transformation reference "ReferenceID."
MESSAGE_ID	A system-generated identifier for the BOD that is added to the message headers. This value currently is not used in the ApplicationArea.
TENANT_ID	The identifier for the tenant. Currently, this variable has a fixed value of "infor."
ORGANIZATION_NODE	The identifier for the organization. This value is obtained directly from the transformation references.

Transformation references

ReferenceID	The format of the reference ID used to generate the BOD ID is related to the key of the business object. This transformation reference is used to specify the value to be used.
AccountingEntityID	The accounting entity ID is optional. It is not required when the object does not relate to a single accounting entity, but spans multiple entities. Used when building the BOD ID.
LocationID	The location ID is optional. It is not required when the object does not relate to a single location. Used when building the BOD ID.
Revision	The revision of the business object is optional. Used when building the BOD ID.

Example

The following example is a fragment of the stylesheet used to send a buyer in the form of a Person, using the Sync noun. Note that none of the optional transformation references are defined.

```
<SyncPerson …>
  <xsl:comment>TX_REF:ReferenceID=<xsl:value-of select="System-
  Link/Response/QueryObjectResponse/DomainEntity/Key/Property[@path='buyer']/Value"/>
  </xsl:comment>
  <ApplicationArea>
    <Sender>
        <LogicalID>%FROM_LOGICAL_ID%</LogicalID>
        <ComponentID>ERP</ComponentID>
                </Sender>
                <CreationDateTime>%CREATION_DATE_TIME%</CreationDateTime>
                <BODID>%BOD_ID%</BODID>
        </ApplicationArea>
        <DataArea>
                <Sync>
                        <TenantID>%TENANT_ID%</TenantID>
                        <AccountingEntityID>%ACCOUNTING_ENTITY_ID%</AccountingEntityID>
                        <ActionCriteria>
                                <ActionExpression><xsl:attribute name="actionCode">
                                        <xsl:call-template name="ActionCode"/></xsl:attribute>
                                </ActionExpression>
                        </ActionCriteria>
                </Sync>
    …
</SyncPerson>
```

The Test Destination Action

Once a recipient's configuration data is entered, the "Test Destination" action provides a way to validate that the connection can be made. Note that this validation will require that the associated recipient program be present and active (and ready to "catch" the test response). This implies that the supporting systems (e.g. WebSphere MQ for "MQ Listeners", and System-Link for "LPI Listeners") should be installed and running as well.

Chapter 6.   The System-Link Protocol in Detail

XA System-Link uses an XML-based request/response protocol that is transported via an HTTP POST request (with MIME type "text/xml"), a WebSphere MQ message, or an RPG program call.
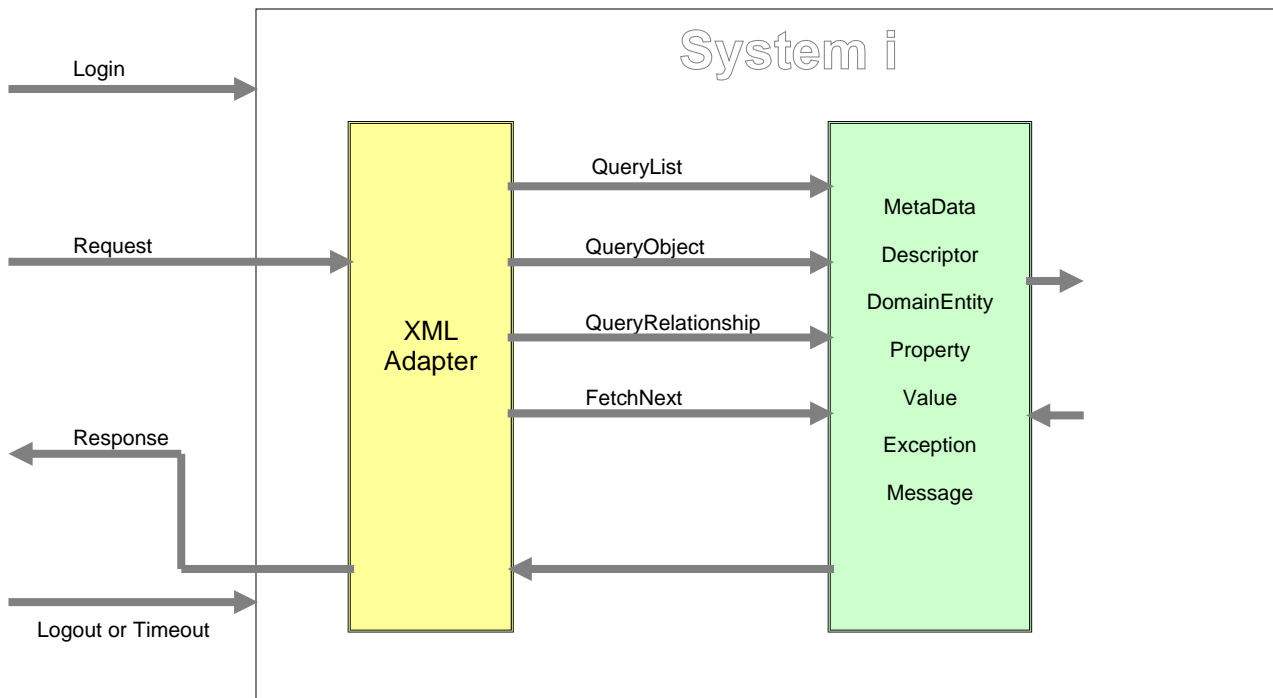
The base unit of a dialog with XA System-Link is the "session". A session consists of a "login" action, one or more "request" actions, and a "logout" action. The login determines user rights via XA security, and specifies other session parameters (such as, XA environment, national language, currency, etc.). Logout, as usual, simply terminates the session. The "request" action(s) contain all of the "real work" of a session, which is performed in the context of a "work area". The work area can be visualized as an isolated "scratch pad" containing all temporary and non-persisted data. A given session can have one or more work areas, but a single work area cannot span sessions.

A key feature of both sessions and work areas is their "timeout" mechanism. The session timeout is specified at login, whereas the work area has its timeout specified at the beginning of the request action. This feature allows both sessions and work areas to persist beyond the processing of a single System-Link request script (or to "idle"), which has the following consequences:

- The login, request, and logout actions can be split across multiple request/response pairs. This allows you to avoid incurring login overhead more than once per dialog. In addition, this allows System-Link to be installed in such a way that the login/logout and the request operations are sent to different ports (e.g. allowing SSL for login/logout while handling the normal request data in the clear).
- Matching the session and work area idle times more closely to the expected usage pattern allows session and work area reuse while minimizing both server overhead and security window size.
- Since a XA login may require up to several seconds to complete, the performance benefit from reusing one login session (as opposed to creating a new login session for every request) can be substantial.

Inside a given request action, the data query and maintenance requests ("actions") are specified. Query action "tags" can be used to request a list of XA domain objects, a single domain object, or a single domain object followed by one or more lists of related domain objects. Maintenance action tags can be used to create, copy, update, or delete an XA domain object. Any given request can contain one or more actions. Each response to a given action will be associated with the action by name, and will contain either the results of the action, or one or more exception messages detailing the errors that occurred. (More information about the format and types of exception messages can be found in Examples under "Exception tag".)

Queries can be specified using one of two methods: PQL or custom definitions. The "Paragon Query Language",

or "PQL" shares much of its syntax with SQL, and therefore should be fairly easy to understand for those already familiar with SQL. When viewing PQL, the SQL-literate person will often run across the following notable difference (example: a SELECT clause fragment for the Domain Class "PurchaseOrder"):
  "SELECT order, relatedBuyer.buyerName, relatedVendor.vendorName1 … "

In the above, "order" specifies that the PurchaseOrder property "order" should be displayed. Likewise, "relatedBuyer.buyerName" and "relatedVendor.vendorName1" specify that "buyerName" and "vendorName1" also be displayed. However, these two properties are not present on PurchaseOrder, but rather are obtained via known relationships with that given PurchaseOrder. In this case, "buyerName" would be obtained from the related Buyer, and "vendorName1" would be obtained from the related Vendor.

An alternative way of specifying a query can be using custom definitions. System-Link can determine which data to retrieve based on a view name rather than a SELECT clause in PQL. Similarly, a subset name replaces the WHERE clause and a sort name replaces the ORDER BY clause. The advantage of using custom definitions instead of PQL is that the results of the XML request can be controlled by simply customizing a definition.

Global Restrictions

A named action (e.g. QueryObject, FetchNext, Update) cannot have the same name as any other named action in the same System-Link document. However, names can be reused across documents for a given System-Link session and work area. (Reuse of names does limit the use of "named objects", however. Named objects are detailed later in this document.)

Chapter 7. Copying a System-Link request from the Browser

The process of creating a System-Link request by hand can be both tedious and subject to errors. To assist you, XA provides the System-Link request automatically from the menu bar.
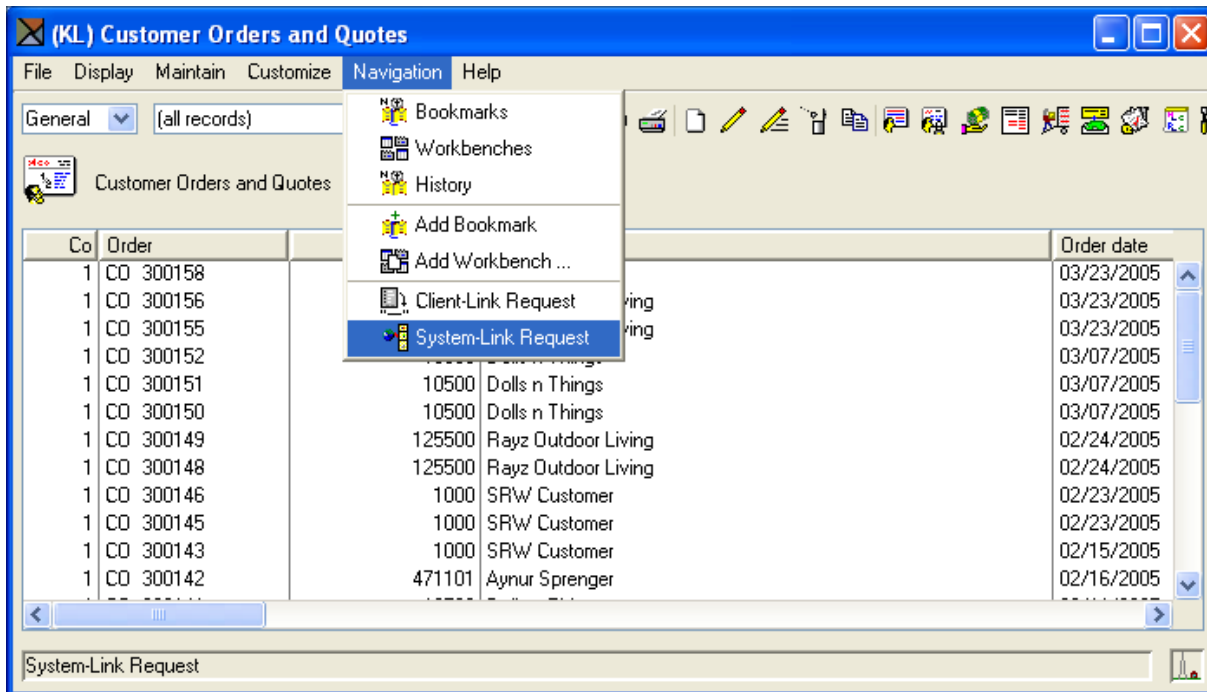
On a list window or card file, select the **System-Link Request** option from the Navigation menu.

The System-Link Request window appears.
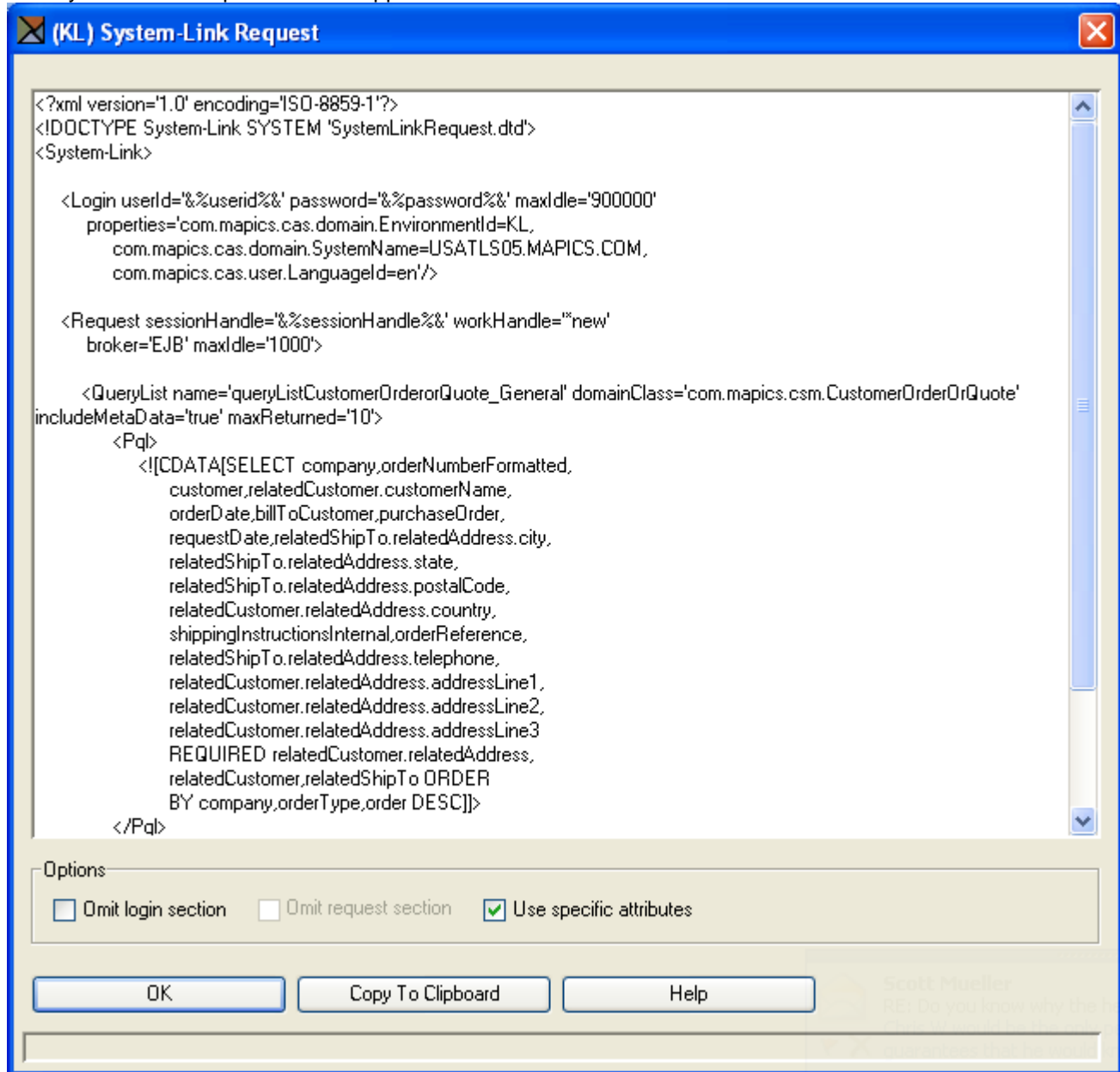
This window shows the System-Link request using the data from the list window or card file. When you ask for the System-Link request for a view, the attributes included in the request are only those listed on the banner or as columns in the view.

Similarly, the System-Link request for a card file includes all the attributes on all the cards in the file, plus any in the banner. If there are customizable list cards in the card file, the request includes the data on those list cards as well.

System-Link allows you to specify the attributes discretely or by the custom definition. In the example above, the attributes for a list of items are identified in the PQL: Item, Description, Item class, etc. (The unusual capitalization and lack of spaces in the attribute names is necessary.)

The System-Link request window appears.
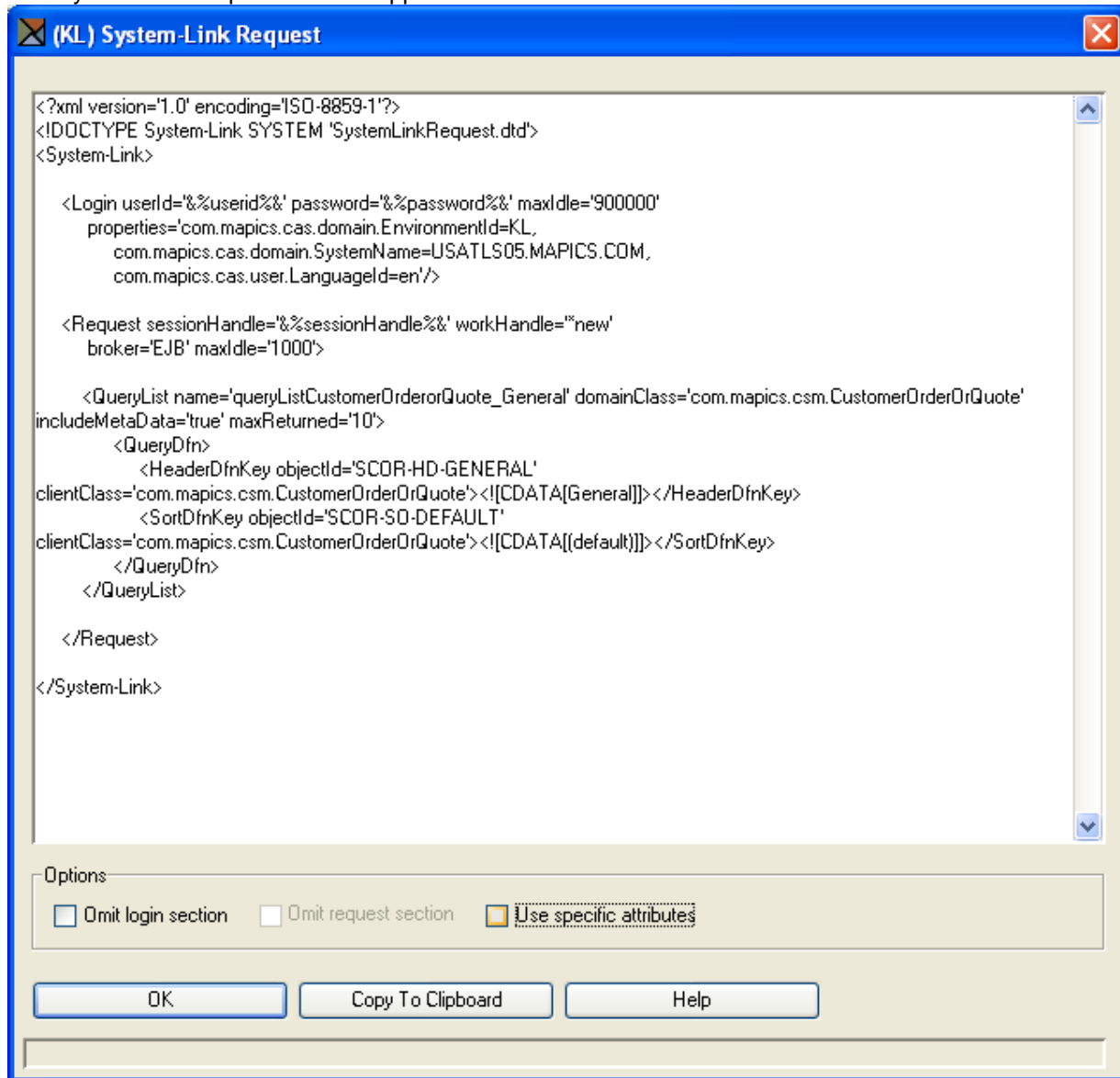
The System-Link request is shown in sections. The login section includes the user ID, password, environment and other login information. The user ID and password are not displayed. Instead, you see the words **userid** and **password** surrounded by percent signs (%) and ampersands (&). These signs indicate a substitution variable. It is assumed that the calling program will substitute the user ID and password for the variables when it sends the System-Link request.

The third section is the heart of the System-Link request. It defines the action (display, create, etc.) and the object, as well as the fields involved. The fields can be defined using custom definitions. When the *System-Link request* menu option is selected from a list window, the view, subset and sort definitions used at the time will be included in the request. This means that the System-Link request will retrieve the same data that you see in the view or card file. User input for prompted subsets becomes part of the System-Link request.

In the illustration shown above, the view definition is *Sample* and the sort definition is *(default)*. When the *System-Link request* option was selected from the menu, no subset was being used.

Every custom definition has a 32-byte token that is its internal system identifier. Using tokens instead of names can make requests language-independent. This is useful when using one System-Link request to support users with multiple languages on the same system. For example, you can specify the correct definition whether it is called *Default* or *Valor por omission* or *Valeur par défaut*. System-Link will display both the name and the token for custom definitions.

To use the System-Link request, click **Copy to Clipboard**. If you want to copy less than the entire XML request, you can highlight part of it and press **Ctl-C** to copy the highlighted portion to the clipboard.

A successful login produces a session. A session can be left open to be reused for future requests, eliminating the need for the user to log in each time. To assist you in doing this, System-Link can strip off the login XML from the request. To do this, click the **Omit login section** checkbox, which is located on the lower part of this window. The System-Link Request window appears again without the login section.
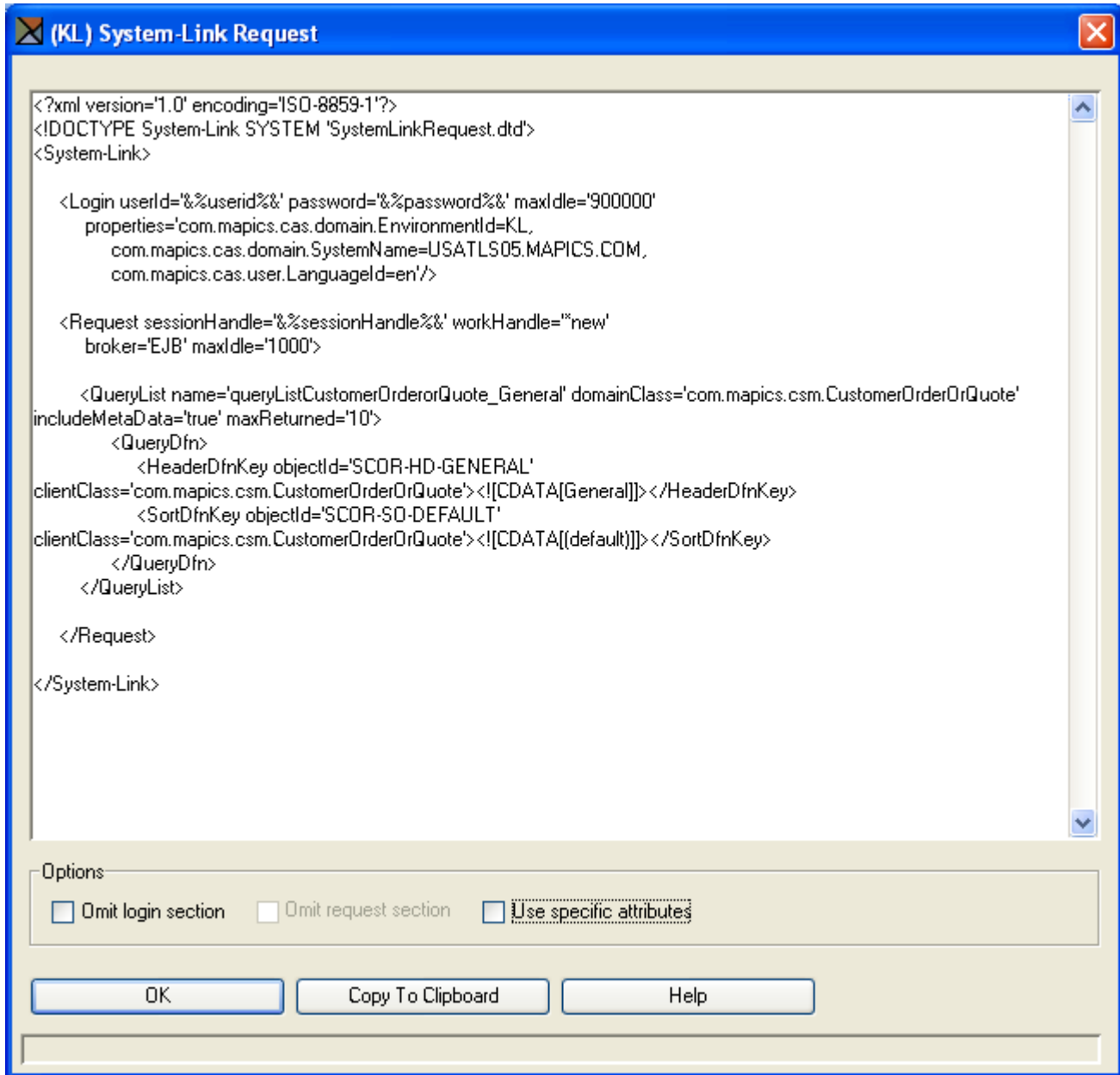
Some System-Link requests are simple, such as requesting a list of items or orders. Others can be complex, such as requesting a single order with all its line items, the releases for each one, the associated warehouse data and various overrides, holds, etc. Complex requests like this need to be assembled from parts of individual requests.

When you omit the login section, XA asks if you want to strip off the request section as well. This leaves only the data section, which is all you need to assemble a complex request. To omit the request section of the XML, click **Omit request section** checkbox. The System-Link Request window appears again without either the login or the request sections.

The advantage of using custom definitions to define System-Link requests is that the requests become very easy to change. For example, adding a field to a view will cause any System-Link request using that view to retrieve the new field. Technical support personnel will appreciate how much easier this makes the maintenance of System-Link requests.

However, there will be some cases where the System-Link request needs to be "hard-coded". (In R6 eZ-Link this was the only option available.) To do this, simply select the Use specific properties checkbox at the bottom of the window. The custom definition entities will be replaced by PQL, with each field listed and each subset criterion identified. (The unusual capitalization and lack of spaces in the field names is necessary.) The PQL will specify the same fields as the view definition, the same filtering criteria as the subset definition, and the same sort order as the sort definition. There is no difference whatsoever in the data retrieved.


```
(KL) System-Link Request

<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE System-Link SYSTEM 'SystemLinkRequest.dtd'>
<System-Link>

    <Login userId='&%userid%&' password='&%password%&' maxIdle='900000'
        properties='com.mapics.cas.domain.EnvironmentId=KL,
            com.mapics.cas.domain.SystemName=USATLS05.MAPICS.COM,
            com.mapics.cas.user.LanguageId=en'/>

    <Request sessionHandle='&%sessionHandle%&' workHandle='*new'
        broker='EJB' maxIdle='1000'>

    <QueryList name='queryListCustomerOrderorQuote_General' domainClass='com.mapics.csm.CustomerOrderOrQuote'
includeMetaData='true' maxReturned='10'>
        <Pql>
            <![CDATA[SELECT company,orderNumberFormatted,
                customer,relatedCustomer.customerName,
                orderDate,billToCustomer,purchaseOrder,
                requestDate,relatedShipTo.relatedAddress.city,
                relatedShipTo.relatedAddress.state,
                relatedShipTo.relatedAddress.postalCode,
                relatedCustomer.relatedAddress.country,
                shippingInstructionsInternal,orderReference,
                relatedShipTo.relatedAddress.telephone,
                relatedCustomer.relatedAddress.addressLine1,
                relatedCustomer.relatedAddress.addressLine2,
                relatedCustomer.relatedAddress.addressLine3
                REQUIRED relatedCustomer.relatedAddress,
                relatedCustomer,relatedShipTo ORDER
                BY company,orderType,order DESC]]>
        </Pql>
```

Options
☐ Omit login section   ☐ Omit request section   ☑ Use specific attributes

OK   Copy To Clipboard   Help

As far as performance and other operational characteristics, there is no advantage or disadvantage to using PQL vs. custom definitions. Other than the maintenance issues already mentioned, it is a matter of preference.

When selecting the *System-Link request* option from a card file menu, the card file definition will be used to specify the list of fields to retrieve. Alternatively, PQL can specify the fields. In either case, all fields from all cards in the card file, plus those in the banner area, will be included.

If the card file is in maintenance mode, the System-Link request will use a maintenance action (i.e., create or update).

As of this writing, the template value will be *(none)* even if the card file was in create mode and a template was applied. The <ApplyTemplate> tag is provided as a placeholder for you to be able to fill in the template name, if any.
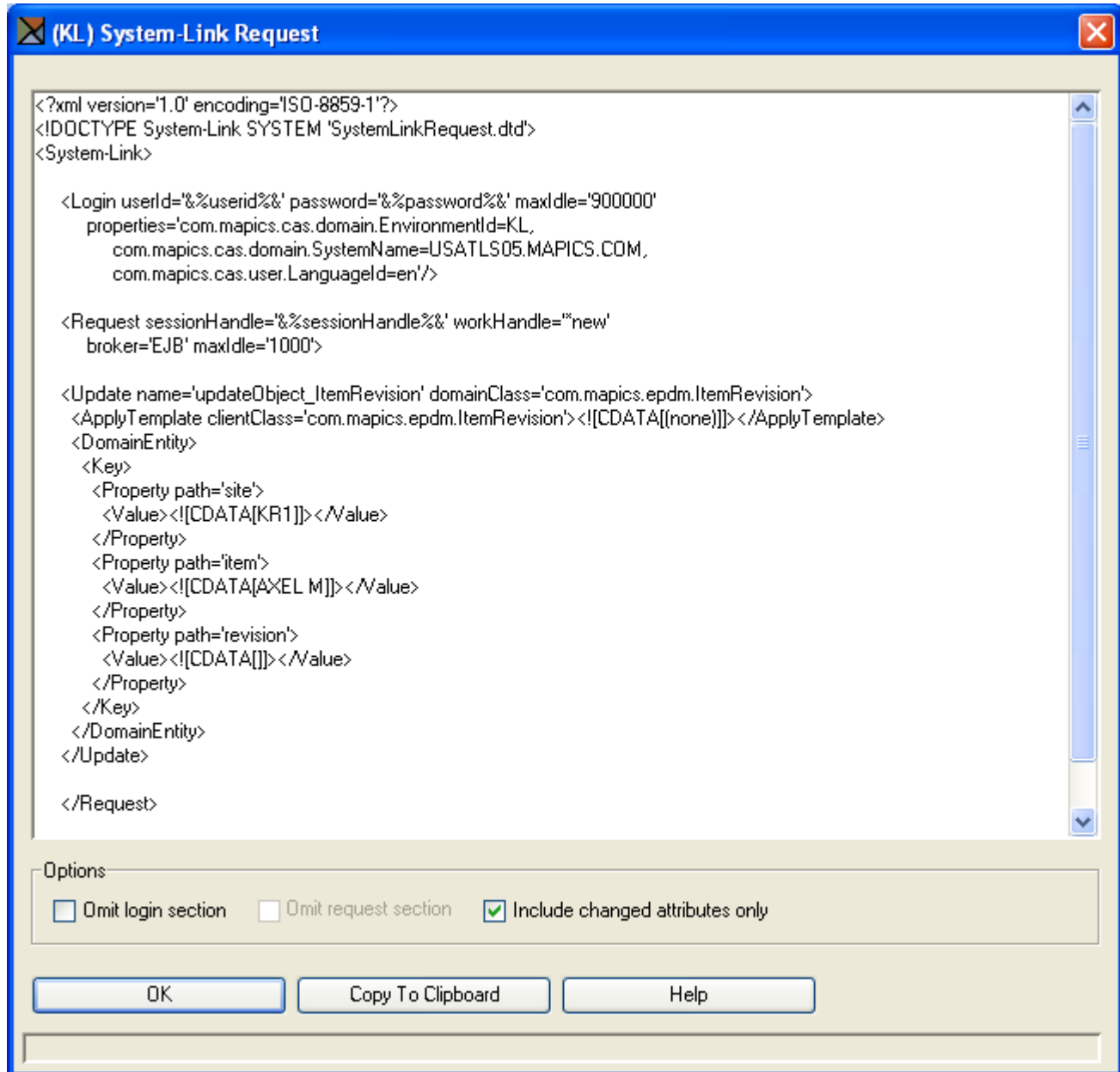
Since the list of fields in a card file can be quite long, System-Link provides the ability to include just the changed fields in the request. The illustration below shows the update action for an item revision record where the department was changed to *AA1A*. Note that the key fields (i.e., properties) are the only fields included other than *department*. (The department field was keyed into the card file but not entered, when the menu option was chosen to display this window.)

The illustration below shows what happens when the Include changed fields only checkbox is not checked. All fields on the banner and the cards in the card file are included. As you can see from the scroll bar, there is an enormous amount of information. You can get it if you need it, but the default is to show just the changed fields.
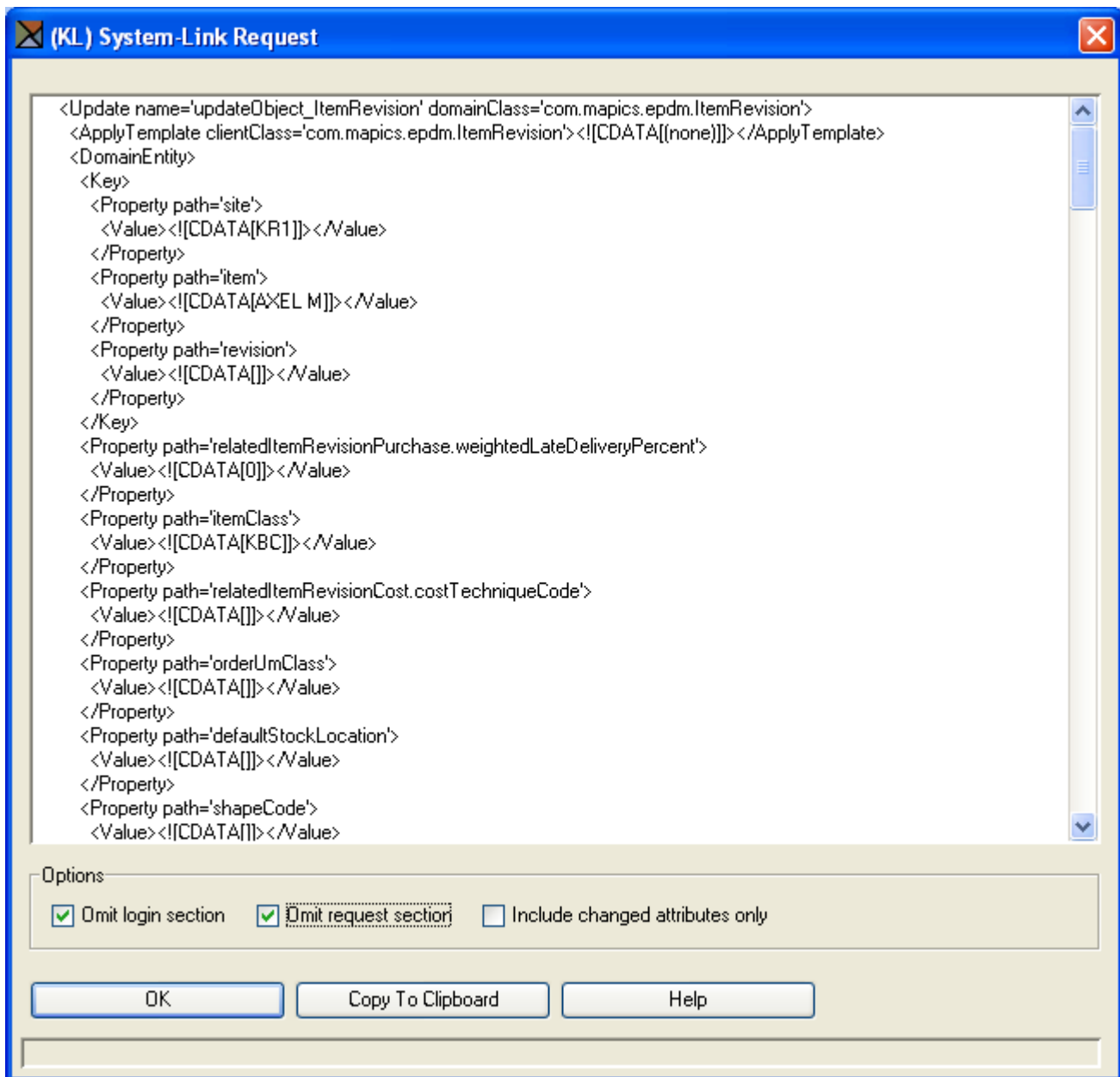
The objective of this feature is to provide you with a starting point for your System-Link request. You can copy it to the clipboard and paste it into your program, document or the System-Link Simulator.

Chapter 8. Using the System-Link Simulator

The System-Link Simulator provides an easy way to test your System-Link request before you putting it into production.

When you install System-Link, you can access the System-Link Simulator at http://yourwebserver.com/SystemLink/Simulator.html, where (yourwebserver) is what you named your web server.

Note*:* In order for the Simulator to work properly, you must use Internet Explorer 5.5 or greater.

On the Simulator, there are four tabs: ***Request, Response, Stylesheet***, and ***View***. Only one tab is displayed at a time. The tab currently displayed has a larger font than the other tabs.
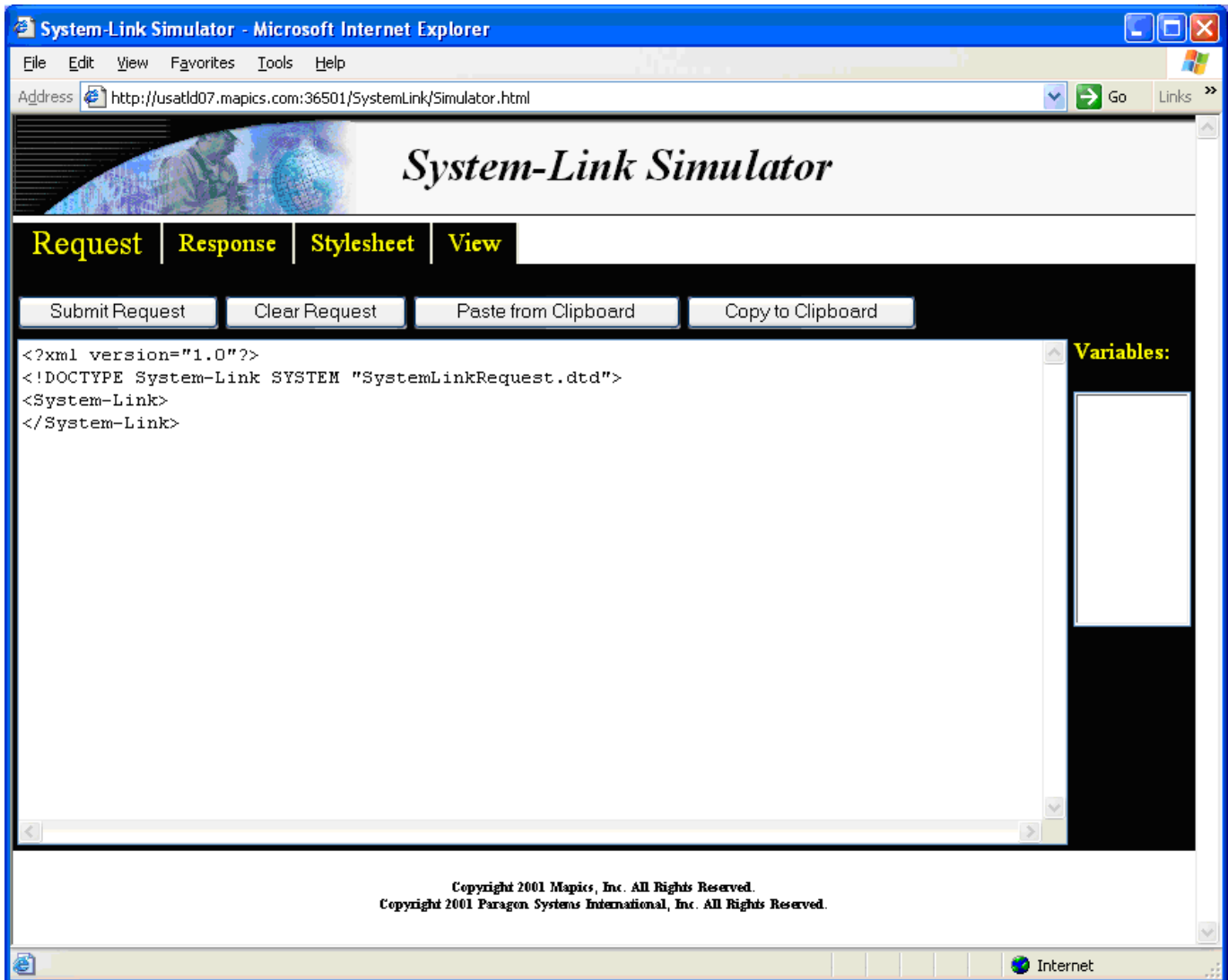
When the Simulator is first displayed, the *Request* tab is on top. To manage the work space, use the buttons located below the tabs:

Submit Request	Submits the System-Link request to the server. The Simulator automatically displays the Response tab while it waits for the System-Link response to return from the server.
Clear Request	Clears the work space on the Request tab. Without this button you would need to use the cursor to highlight the entire work space and press the delete key.
Paste from Clipboard	Clears the workspace, then copies the request on the clipboard to the Simulator.
Copy to Clipboard	It will copy the entire contents of the work space to the clipboard.

Note**:** When you want to copy less than the entire request, you can use your cursor to select (highlight) the part you want to copy. Place your cursor at the beginning of what you want to copy, then hold the **Shift** key down while using the keyboard cursor keys to highlight the text. Once the text is selected, you can use **Edit/Copy** or **Ctl+C** to copy it to the clipboard**.**

When you want to copy the entire System-Link request, use **Copy to Clipboard**. Do not use the **Select All** option from the Edit menu. The **Select All** option selects the entire html document, including the graphics, the buttons, the header, etc.

Using variables in the Request

Variables allow you to insert data into a System-Link request before submitting it to the server. For example, you can have a System-Link request retrieve item data. The variable could be the item number that allows the same System-Link request to work for any item.

Variables are defined using the following syntax:

     &%name%&

where *name* indicates the variable name. Variable names can include any letter or number (no special characters, no blanks) and are case-sensitive. The following are valid variable names:

     userid, variable1, DueDate, 0001.

These would appear in the XML as

     &%userid%&, **&%variable1%&**, **&%DueDate%&** and **&%0001%&**

Reserved variable names are **styleSheet, SystemLinkRequest,** and **SystemLinkRequestURL**.

When you sign on to a web page using a user ID and password, the web page captures the user ID, password and session handle as variables. The web page might capture additional variables, such as product or order numbers, dates, etc. When you use **send**, the variables are sent to the servlet (the URL includes &userId=JSMITH), which merges them with the request before submitting it to the server. In this way, the System-Link request is made flexible enough to support a wide variety of requests.

The Simulator makes it easy for you to test the System-Link request with variables. The **Variables** box to the right of this text area shows the variables and their values. When you first paste a System-Link request (one that includes the login section) into the Simulator, the userid and password appear in the **Variables** box.

To assign values to the variables, click the variable name in the **Variables** box. Two things happen when you do this:

1. A window appears below the variable name list where you can type in the value. As you type them in, the value appears next to the name in the **Variables** box.

2. In the text area, the variable is highlighted so you can find it in your XML. If you have the same variable in multiple places, only the first one is highlighted.

When you open the Simulator, there are no variable values. The values you enter are stored in the Simulator until you refresh it or close it. This means, for example, that you can use the same user ID and password for testing multiple System-Link requests without having to re-enter them each time.

When a System-Link request is created by the XA Browser, the user ID and password are automatically made into substitution variables, (shown in bold in the following example):

```
<Login userId='&%userid%&' password='&%password%&' maxIdle='900000'
     properties='com.mapics.cas.domain.EnvironmentId=6C,
         com.mapics.cas.user.LanguageId=enu'/>
```

You can create additional substitution variables by entering the variable name surrounded by percent signs and ampersands. In the following request, the WHERE clause asks for buyer 550 (shown in bold):

```
  <QueryObject name='queryObject_Buyer_Default'
 domainClass='com.mapics.pm.Buyer'    includeMetaData='true' >
        <Pql>
           <![CDATA[SELECT buyer,buyerName,buyerTelephone,fax,EMailAddress,
              buyerRating,department,buyerUserId, WHERE buyer='550']]>
        </Pql>
     </QueryObject>
```

The buyer number can be changed to be a variable called *buyer* (or anything else you choose). Variables are not shown in quotation marks. When you add the percent signs and ampersands, it looks like this (shown in bold) in the following example:

```
  <QueryObject name='queryObject_Buyer_Default'
 domainClass='com.mapics.pm.Buyer'    includeMetaData='true' >
        <Pql>
           <![CDATA[SELECT buyer,buyerName,buyerTelephone,fax,EMailAddress,
              buyerRating,department,buyerUserId, WHERE buyer=&%buyer%&]]>
        </Pql>
     </QueryObject>
```

As soon as the last ampersand of **&%buyer%&** is entered, the new variable name appears in the *Variables* box. The new variable has exactly the same behavior as that described above for any other variable. The parentheses, next to the name in the *Variables* box, are empty because no value has been assigned. (The original *550* in the example above is ignored.)

There is no limit to the number of variables you can have in your XML. The *Variables* box shows ten at a time. If you have more than that, a scroll bar appears. There is no limit to how long a variable value can be in the Simulator. However, some browsers have a limit on how long a URL can be (for instance, 255 characters). Variables can be included in a URL. The variables plus the rest of the URL must fit within the limit determined by the browser.

Variables:

```
userid (JSmith)
password (1X2Y3Z)
buyer ()
```
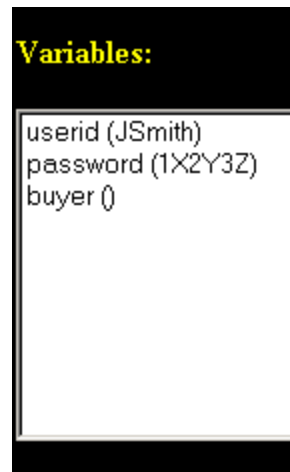
The Response tab

The **Response** tab shows the System-Link response returned from the server. It is blank when you first start the Simulator. The only way to get a System-Link response is to submit a System-Link request.

You cannot maintain the XML on this tab. However, all or part of it can be highlighted using the cursor to copy it to the clipboard.
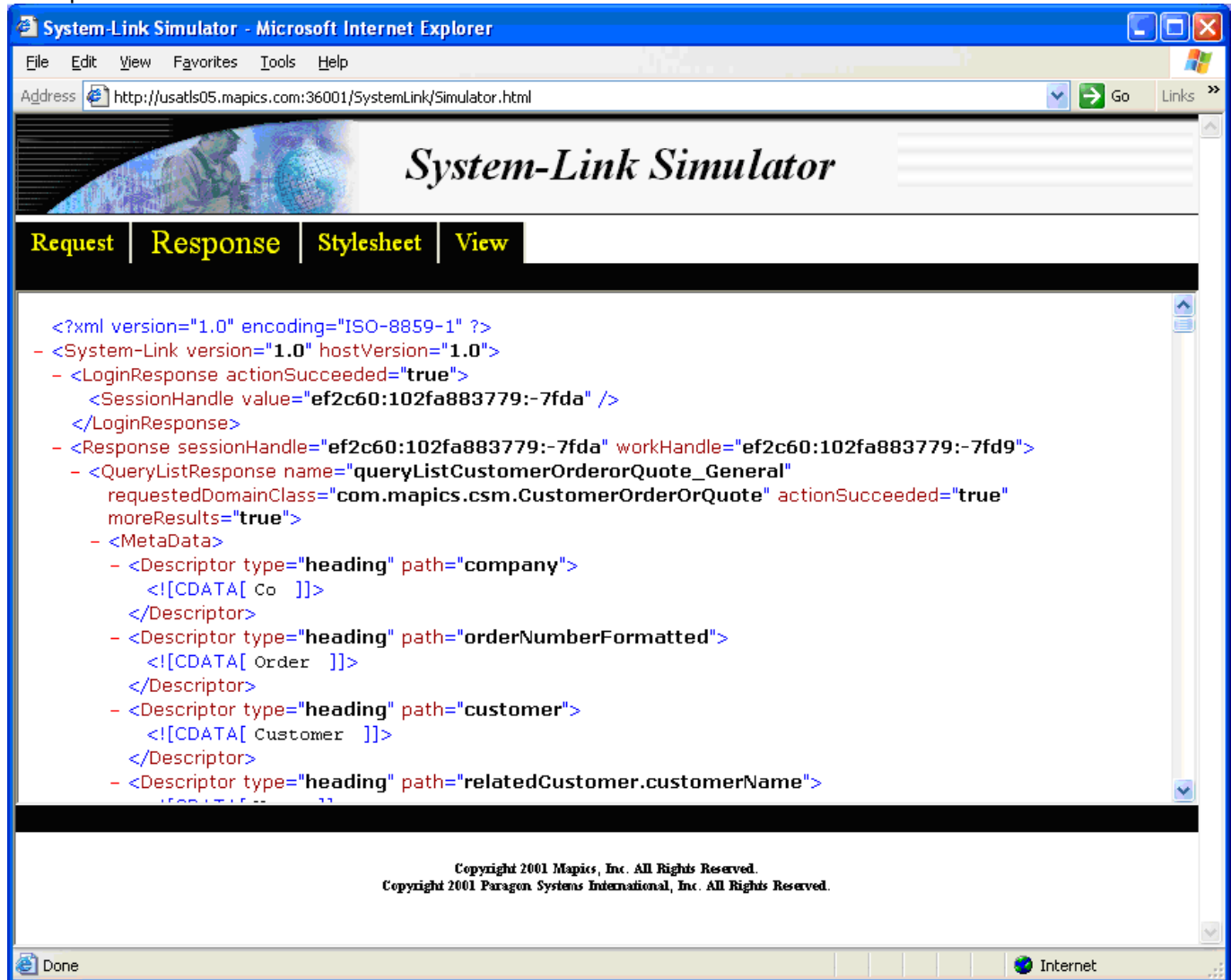
You can look at the response to verify that it is displaying what you want. This is the XML that is returned to your program. If it is not what you want, you can switch back to the **Request** tab to change the System-Link request. You can then resubmit the request and check the response on the **Response** tab again. You can do this as many times as necessary.

The **Response** tab shows any errors. Errors, due to mistakes in the request, can be corrected; and the request can be resubmitted to get a new response. You can also force errors intentionally, such as to test application edits.

The XML on the **Response** tab can be expanded and collapsed. When it is first displayed, all nodes are expanded. When the XML is lengthy, it can be difficult to understand. It is helpful to collapse some of the nodes in order to see the "big picture". The minus sign (-) means the node can be collapsed. The plus sign (+) means the node can be expanded.

The illustration to the right shows an example of the MetaData, Key, and Property path nodes collapsed.

```
<?xml version="1.0" ?>
- <System-Link version="1.0" hostVersion="1.0">
  - <LoginResponse>
      <SessionHandle value="11a775:ea7a8d9889:-7f9e" />
    </LoginResponse>
  - <Response sessionHandle="11a775:ea7a8d9889:-7f9e" work
    - <QueryListResponse name="queryList_ItemRevision_Stat
        requestedDomainClass="com.mapics.epdm.ItemRevisio
      + <MetaData>
      - <DomainEntity domainClass="com.mapics.epdm.ItemRe
        + <Key>
        + <Property path="item">
        + <Property path="description">
        + <Property path="site">
        + <Property path="revision">
        + <Property path="itemType">
        + <Property path="implementationStatus">
        + <Property path="relatedItemRevisionStatus.releaseT
        + <Property path="relatedItemRevisionStatus.releaseT
        + <Property path="relatedItemRelease.releaseDate">
```

The Stylesheet tab

The **Stylesheet** tab displays the XSL stylesheet. The **View** tab uses this stylesheet to format the System-Link response.
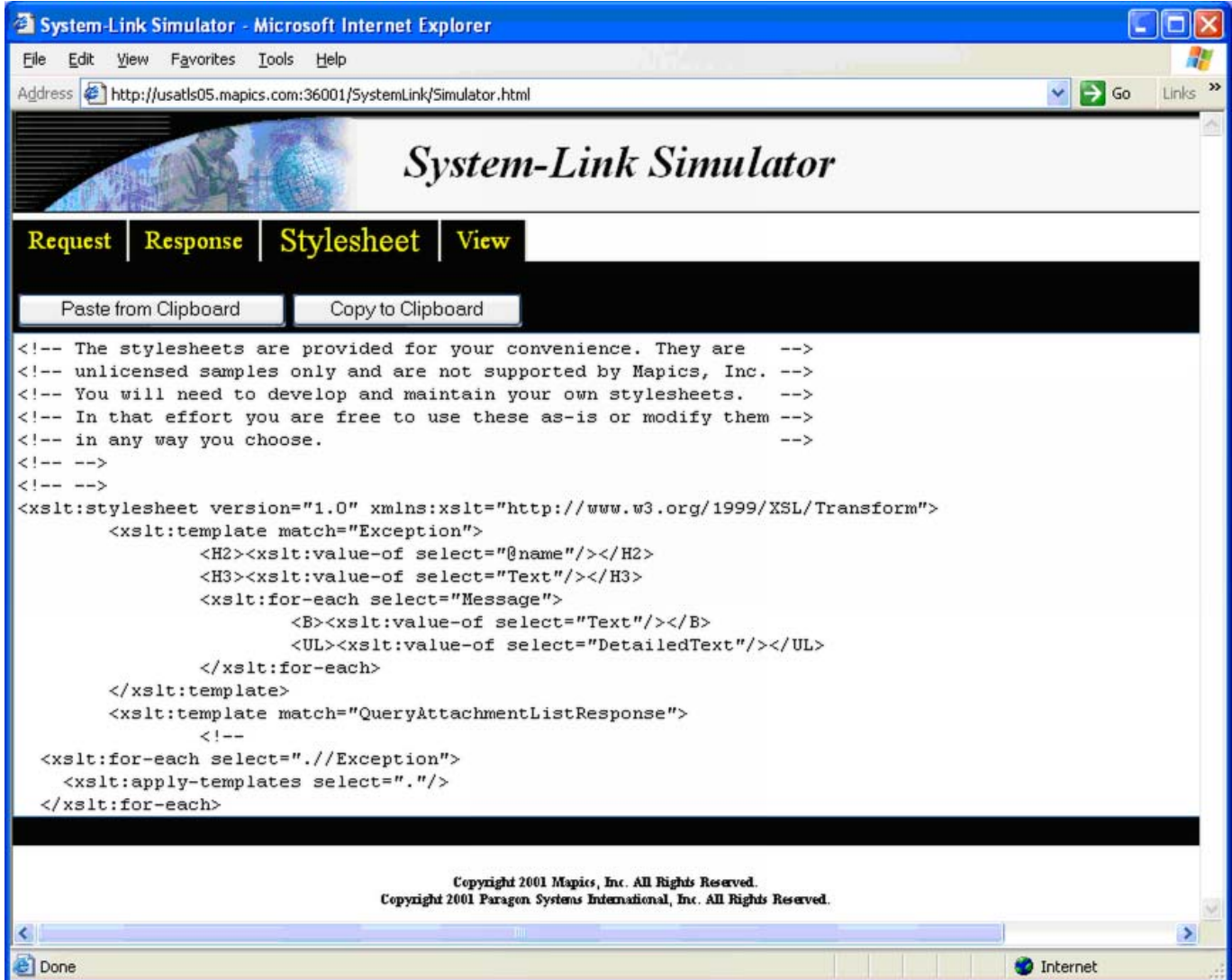
The stylesheet is valuable in two ways:

1. It formats the data in the System-Link response so that you can view it on the **View** tab. It allows you to validate the XML request using the **View** tab, therefore, making it easier than examining the data in the System-Link response. The shipped stylesheets are used for this purpose.

2. It provides a way to test stylesheets used with web applications. The best characteristics of a stylesheet depend on the amount and type of data presented. The Simulator allows a programmer to paste in a stylesheet to see how it looks with the data. You can make changes to the stylesheet and view it with the data. A satisfactory stylesheet can be copied from the Simulator to the clipboard and can be pasted into an XSL document.

Two basic stylesheets are shipped with the Simulator, as a convenience to the programmer. They are provided as-is and are not supported by Infor.

The View tab

The **View** tab shows how the XML from the **Response** tab looks after it is transformed by the XSL stylesheet.

This is the easiest way to see whether your System-Link request needs to be changed. If the data is not what you expected, the System-Link request on the **Request** tab can be adjusted and resubmitted.

When the **View** tab displays problems with the stylesheet, you can change it on the **Stylesheet** tab. The query does not need to be resubmitted; changes to the stylesheet are immediately apparent in the **View** tab.

The stylesheets shipped with the Simulator displays an object request as a list of attributes. The stylesheets display a list request as a table of attributes. Relationship requests display the object data followed by list data. The stylesheets shipped with the Simulator do not display nested relationship requests (i.e., tree structures). The System-Link response, however, will support an unlimited number of nested relationships.

The System-Link Simulator is a convenient tool to test a System-Link request. However, it is not required to process System-Link requests. Production System-Link requests bypass the Simulator and go directly to the servlet.

Page 57

It is the responsibility of the programmer to use a valid System-Link request for production. The Simulator can assist in the following ways:

- Web programmers can copy/paste a tested System-Link request from the Simulator to a web page. The web page needs to provide substitution variables, such as user ID, password, customer number, item number, etc. Web programmers can also use the Simulator to test the XSL (the stylesheet) rather than their System-Link requests.

- Server programmers can create programs that produce requests to be sent to System-Link. The Simulator can test such programs by processing the XML output. When the Simulator displays problems with a System-Link request, the program producing the XML needs to be modified accordingly. Thus, the Simulator provides programmers a way to test server-to-server programs.

Chapter 9.   Examples

This chapter illustrates the System-Link syntax using typical business examples. Each example starts with the XA Browser list window(s) and/or card file(s) to show the business data in context. Once the Browser view or card file has been described, it is followed by the System-Link request that would request the business data seen in the XA Browser. A detailed discussion deconstructs the request in order to clarify each section's role in requesting the business data. Finally, the System-Link response is shown, with an equally detailed deconstruction of its various sections.

The examples are meant to be studied in order. Certain System-Link tags are used in some or all of the examples, but they are described only in the first example in which they appear. Subsequent examples consider only the new features.

To assist the reader in finding various features, the following table lists the first example where a feature is described:

Feature	Example
Login tag	Example 1. Query a list (Items)
Session handle	Example 1. Query a list (Items)
QueryList tag	Example 1. Query a list (Items)
FetchNext tag	Example 1. Query a list (Items)
PQL	Example 1. Query a list (Items)
Using view, subset and sort definitions instead of PQL	Example 1. Query a list (Items)
MetaData tag	Example 1. Query a list (Items)
DomainEntity tag	Example 1. Query a list (Items)
Key property	Example 1. Query a list (Items)
Skipping login	Example 2. Query an object (Customer)
QueryObject tag	Example 2. Query an object (Customer)
Key property (multi-part key)	Example 2. Query an object (Customer)
Properties 　　Dates 　　Decimals 　　Boolean	Example 2. Query an object (Customer)
Using a card file instead of PQL	Example 2. Query an object (Customer)
QueryRelationship tag	Example 3. Query multiple objects (Manufacturing order)
Substitution variables	Example 3. Query multiple objects (Manufacturing order)
Create tag ReasonCode tag ApplyTemplate tag SourceObject tag	Example 4. Create or copy a simple object (Warehouse)
ValueRef tag	Example 5. Create multiple simple objects together (Warehouse with Address)
Update tag	Example 8: Update (PO line)
Delete tag	Example 9: Delete (Warehouse)

Spacing used in Examples

The XML shown in the examples may not always *exactly* match what you see in IE 5.5. The following points should be understood:

a) Generally, vertical space and horizontal space are ignored by the parser. For example, the following request fragment indicates the value of the customer field is 101:

```
<Property path='customer'>
    <Value>101</Value>
<Property>
```

The same XML could look have the horizontal space eliminated…

```
<Property path='customer'>
<Value>101</Value>
<Property>
```

…or the vertical space eliminated:

```
<Property path='customer'><Value>101</Value><Property>
```

All three of these illustrations are identical to the parser. The examples in this chapter show XML with whatever spacing best illustrates the point being made.

b) This rule does not apply to the <Value> tag. The parser will consider everything between the <Value> start tag and the </Value> end tag to be intentional. In the following fragment the value is *Example*:

```
<Value><![CDATA[Example]]></Value>
```

However, in this XML the value is *(new line)(space)(space)(space)Example(new line)*:

```
<Value>
   <![CDATA[Example]]>
</Value>
```

For this reason it is important to have only what you intend between the <Value> start tag and the </Value> end tag. System-Link respects this rule in the response, but IE 5.5 will often add new lines to make it easier (in the mind of Microsoft) to understand. This is purely a display feature; the underlying XML is unchanged.

c) In a similar fashion, IE 5.5 will sometimes add leading and trailing blanks to the significant characters in a CDATA section. The XML may be this:

```
<![CDATA[101]]>
```

but IE 5.5 may display it like this:

```
<![CDATA[  101  ]]>
```

Again, this is a display-only feature. The underlying XML is unchanged.

Example 1. Query a list (Items)

Below we see a list of six items. The custom definitions are as follows:

View:	Item, Description, Item type, Drawing, Stocking unit of measure and Department.
Subset:	All items containing "PC" in the description.
Sort:	Item number in ascending sequence.

XML Request

The System-Link request generated for the list window above looks like this:

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE System-Link SYSTEM 'SystemLinkRequest.dtd'>
<System-Link>

   <Login userId='USERID' password='PASSWORD' maxIdle='900000'
      properties='com.mapics.cas.domain.EnvironmentId=DD,
         com.mapics.cas.domain.SystemName=USATLD06.INFOR.COM,
         com.mapics.cas.user.LanguageId=en'/>

   <Request sessionHandle='*current' workHandle='*new'
      broker='EJB' maxIdle='900000'>

      <QueryList name='queryList_Item_Sample' domainClass='com.mapics.epdm.Item' includeMetaData='true'
maxReturned='10'>
         <Pql>
            <![CDATA[SELECT item,description,itemClass,itemType,drawingNumber,
               stockingUm,department WHERE ( (description like '%PC%'))
                ORDER BY item EXCLUDE UNAUTHORIZED]]>
         </Pql>
      </QueryList>

   </Request>

</System-Link>
```

The request can be broken down into sections, shown below. The XML lines will be shown after the section

heading, followed by a discussion of what the XML means.

Request Header

<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE SystemLink SYSTEM 'SystemLinkRequest.dtd'>

This entire System-Link request will be sent to the XML parser. The parser needs to know what level of XML to expect. Therefore, the first line in this example tells the parser that the following XML will be compliant with XML 1.0 or higher. It also specifies the Latin character set to use.
The second line identifies the DTD where the request syntax is specified. This DTD document was installed with System-Link.

System-Link tag

<System-Link>

This specifies the beginning of the System-Link document to the parser. There is another tag at the end to indicate that there is no more XML to follow for this System-Link document.

Login tag

<Login userId='JSMITH' password='1X2Y3Z' maxIdle='900000'
        properties='com.mapics.cas.domain.EnvironmentId=DD,
                com.mapics.cas.domain.SystemName=USATLD06.INFOR.COM,
                com.mapics.cas.user.LanguageId=en'/>

The User ID is JSMITH with a password of 1X2Y3Z. The maximum time the session can be idle is 900,000 milliseconds, or900 seconds, or 15 minutes. This request will be run against the DD environment on the USATLD06 machine, and the language is en (English).
Notice the forward slash (/) at the end, right before the greater-than sign(>). This indicates that there will not be a separate Login end tag.

Request tag

<Request sessionHandle='*current' workHandle='*new'
        broker='EJB' maxIdle='900000'>

This request is for a new session, indicated by *current*. The user will experience a delay while the system checks security and (if valid) creates a session. This delay can be avoided for future requests by skipping the login section and including a session handle in place of *current*. The session handle is obtained from a previous System-Link response to a successful login.

Each query must have a work area to work in. The work area can expire. Thus, the System-Link request in our example specifies a new work area with a maximum idle time of 900,000 milliseconds, or 15 minutes.

The broker will always be EJB, for now.

Query list tag

        <QueryList name='queryList_Item_Sample' domainClass='com.mapics.epdm.Item'
includeMetaData='true' maxReturned='10'>

Since we are asking for a list System-Link specifies a QueryList. If we were asking for one object (i.e., a card file or "details"), the System-Link tag would be QueryObject.

We need a name for this query. The Browser gave us a name of queryList_Item_Sample. The name is a composite of the query type (list vs. object) plus the object being queried (Item) plus the view name (Sample). The query name is useful for identifying query responses when you have multiple queries in one System-Link request. If you limit the number of records to be returned in one request (a good idea) the query name can also simplify a request for additional records.

Since the list is a list of items, the domain class is Item. In Java, the package must also be specified along with the name. (A package is similar to a path.) The package for Item is com.mapics.epdm. This means that request must ask for com.mapics.epdm.Item in order to query the Item object.

The attribute meta data, such as headings, will be returned in the System-Link response (*includeMetaData='true'*).

The server will return only ten records at the most (*maxReturned='10'*). In our example we only got six because that's all there were. If there had been more than ten we would have gotten only ten. This parameter is useful to cap the amount of data being sent to the requesting program. If the requestor is a user, (as opposed to a server program) it is usually recommended that you send only the amount of data that a user can reasonably comprehend. That may be, say, 50 records at a time. You can write the System-Link request so that the user can keep asking for more, with each request limited by this parameter.

PQL tag

```
<Pql>
    <![CDATA[SELECT item,description,itemClass,itemType,drawingNumber,
        stockingUm,department WHERE ( (description like '%PC%'))
         ORDER BY item]]>

</Pql>
```

This section is the actual request for data. It maps closely to the view, subset and sort of the original browser list window:

Select: This is similar to a view, since it specifies the attributes that will be provided for each record. The attributes are not exactly like the attribute descriptions in the XA Browser, because they must follow different naming rules. For example, they must not have blanks, they must be unique, and they have some quirky rules about upper/lower case. Nevertheless, they are still recognizable as the columns from the original view.

Where: This is similar to a subset. The subset we used in the original list window was all items with a description that contained *PC.* Here you can see that *PC* is preceded and followed by a percent sign (%) indicating a wildcard (like an asterisk in MS-DOS).

Order by: This is similar to a sort. The default order is ascending, so it is not specified. Had it been descending, it would have "desc" at the end, as in "ORDER BY item desc".

Exclude: System-Link honors XA security, including content security. To improve performance the records can be excluded on the host by adding "EXCLUDE UNATHORIZED" to the end of the PQL. Without the EXCLUDE clause a "Not authorized" message will be returned for each record the requestor is not authorized to.

QueryDfn tag (not shown in example)

```
<QueryDfn>
  <HeaderDfnKey clientClass='com.mapics.epdm.Item'><![CDATA[Sample]]></HeaderDfnKey>
  <SubsetDfnKey clientClass='com.mapics.epdm.Item'><![CDATA[Desc contains ...]]></SubsetDfnKey>
    <PromptedSubsetValues>
      <CriteriaSpec path='description'><Value><![CDATA[pc]]></Value></CriteriaSpec>
    </PromptedSubsetValues>
  <SortDfnKey clientClass='com.mapics.epdm.Item'><![CDATA[(default)]]></SortDfnKey>
</QueryDfn>
```

The QueryDfn tag is an alternative to the PQL tag. It yields the same results using custom definitions.

HeaderDfnKey: This replaces the SELECT clause by identifying the view to use.

SubsetDfnKey: This replaces the WHERE clause by identifying the subset to use. A prompted subset will require additional tags to supply the prompt value(s).

SortDfnKey: This replaces the ORDER BY clause by identifying the sort to use.

System-Link Response

The System-Link response is what will be returned to the calling program, or placed in the outbound file specified to the CL program. The Simulator displays the XML with the ability to expand or collapse nodes. The minus sign (-) appears next to a node that can be collapsed, while the plus sign (+) appears next to a node that can be expanded. The XML on the Response tab is first displayed with all nodes expanded. In order to make the XML in which you're interested easier to comprehend, it is often helpful to collapse the other nodes.

The System-Link response for the previous request is shown below. Nodes with a plus sign have been collapsed. (Minus signs have been removed for clarity.)

```
<?xml version="1.0" ?>
<System-Link version="1.0" hostVersion="1.0">
    <LoginResponse>
        <SessionHandle value="61d36b:e54302f9b5:-7fc8" />
    </LoginResponse>
    <Response sessionHandle="61d36b:e54302f9b5:-7fc8" workHandle="61d36b:e54302f9b5:-7fc6">
        <QueryListResponse name="queryList_Item_Sample" requestedDomainClass="com.mapics.epdm.Item"
         moreResults="false">
        +  <MetaData>
        +  <DomainEntity domainClass="com.mapics.epdm.Item">
        +  <DomainEntity domainClass="com.mapics.epdm.Item">
        +  <DomainEntity domainClass="com.mapics.epdm.Item">
        +  <DomainEntity domainClass="com.mapics.epdm.Item">
        +  <DomainEntity domainClass="com.mapics.epdm.Item">
        +  <DomainEntity domainClass="com.mapics.epdm.Item">
        </QueryListResponse>
    </Response>
</System-Link>
```

Response Header

<?xml version="1.0" ?>
-          **<System-Link version="1.0" hostVersion="1.0">**

The System-Link response header performs the same functions as the XML request header. This is because the response may be passed to other systems that do not have access to the request. Thus, the first line says that the following XML will be compliant with XML 1.0 or higher.

Login response tag

- **<LoginResponse>**
       <SessionHandle value="7f1ba3:e5347e3f6f:-7ff1" />
    </LoginResponse>

The login response shows that the UserID and password were valid. The result is a session handle that will be used for this request. Had the UserID and/or password not been valid the response would have been limited to an error message.

The session will remain open for 100 minutes, since that is the maximum idle time we specified at login. We can avoid the login delay by using this session handle for additional requests. We would simply include the session handle value in place of ***current** in our request tag.

Response tag

- **<Response sessionHandle="7f1ba3:e5347e3f6f:-7ff1" workHandle="7f1ba3:e5347e3f6f:-7fec">**

The session handle is passed to the requestor as a result of the login. The requestor can use this for future requests to save the login time.

The work area is like a session within a session. Multiple work areas are useful for maintenance, especially for complex objects where you want to send a header record and multiple detail records all together. Generally, for inquiry the entire response will be for one session and one work area.

Query list response tag

**<QueryListResponse name="queryList_Item_Sample"
requestedDomainClass="com.mapics.epdm.Item" moreResults="false">**

What follows is the response to the **queryList_Item_Sample** request of the Item object. **moreResults = "false"** means that we have all the records that met the selection criteria. The request specified no more than ten records in the response. Ours has six. If **moreResults** were *"true"*, we could use the FetchNext tag, described later, to display any or all of the remaining results.

A System-Link request can have more than one query list request. Each will have a unique name. The System-Link response will use each query list response's name to associate it with its query list request.

Meta data tag

<MetaData>

The Meta data tag indicates that the XML to follow is meta data. You only get meta data if you ask for it in the query list tag. Generally, you ask for meta data when you expect to display the results of your request. If the results were going directly to another database (i.e., server-to-server), the time and load on the system to get the meta data would be wasted.

Our original view had seven columns. Each one is an attribute, so our System-Link request included seven attributes. The XML attribute name is similar to the Browser attribute name, except that there are no spaces and the capitalization is different:

Attribute	Heading	XML attribute name
Item	Item	Item
Description	Description	Description
Item class	Class	ItemClass
Item type	I/T	ItemType
Drawing number	Drawing	DrawingNumber
Stocking UM	Stk UM	StockingUm
Department	Dept	Department

Each attribute has a group of response lines that describe the meta data. The following lines describe the meta data for the Item attribute:

> - **<Descriptor type="heading" path="item">**
>   **<![CDATA[Item]]>**
>   **</Descriptor>**

This fragment says that the type is **heading**. (Another type could be **label**, which we will see in our next example.) The **path** indicates which attribute this heading is for – in this case, the **item** attribute. Finally, the character data inside the CDATA section of the meta data for this attribute is **Item**. Therefore, **Item** is the **heading** for the **item** attribute. This may look redundant since the attribute and the heading are almost identical. Sometimes the difference between the heading and the attribute name is more pronounced, as in the case of **Item type**:

> - **<Descriptor type="heading" path="itemType">**
>   **<![CDATA[I/T]]>**
>   **</Descriptor>**

In this case, **I/T** is the **heading** for the **itemType** attribute.
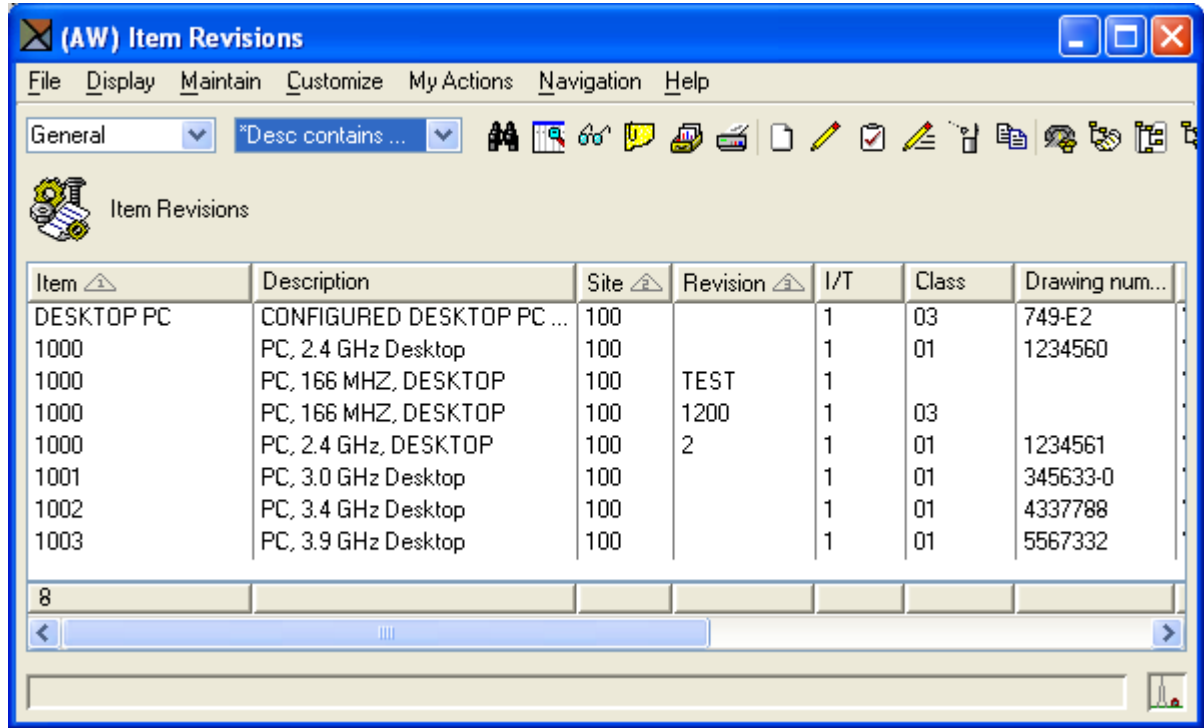
In the previous view of the response, XML the meta data node was collapsed. It is shown expanded below. You will see all seven of the attributes listed. At the end is the meta data end tag: **</MetaData>**

```xml
<?xml version="1.0" ?>
<System-Link version="1.0" hostVersion="1.0">
    <LoginResponse>
        <SessionHandle value="61d36b:e54302f9b5:-7fc8" />
    </LoginResponse>
    <Response sessionHandle="61d36b:e54302f9b5:-7fc8" workHandle="61d36b:e54302f9b5:-7fc6">
        <QueryListResponse name="queryList_Item_Sample" requestedDomainClass="com.mapics.epdm.Item"
         moreResults="false">
            <MetaData>
                <Descriptor type="heading" path="item">
                    <![CDATA[Item]]>
                </Descriptor>
                 <Descriptor type="heading" path="description">
                    <![CDATA[Description]]>
                </Descriptor>
                <Descriptor type="heading" path="itemClass">
                    <![CDATA[Class]]>
                </Descriptor>
                <Descriptor type="heading" path="itemType">
                    <![CDATA[I/T]]>
                </Descriptor>
                 <Descriptor type="heading" path="drawingNumber">
                    <![CDATA[Drawing number]]>
                </Descriptor>
                <Descriptor type="heading" path="stockingUm">
                    <![CDATA[Stk UM]]>
                </Descriptor>
                <Descriptor type="heading" path="department">
                    <![CDATA[Dept]]>
                </Descriptor>
            </MetaData>
             <DomainEntity domainClass="com.mapics.epdm.Item">
************************* NOTE: The rest of the XML is not shown. *************************
```

Domain entity tag

- **<DomainEntity domainClass="com.mapics.epdm.Item">**

A **domain entity** is a business object, like item, customer, order, or vendor. This is the data that your System-Link request is seeking. Our System-Link request asked for item data, and this tag signals that the request that follows will be from the item object. See the following list window showing six items.

This is the data that our System-Link response will contain. Each line of data will have a domain entity tag in the response. Since we are getting six records, we see six domain entity tags. Each one will be identical to the one shown above, but the keys and properties following them will vary based on the data in the records.

The System-Link response shown earlier had the domain entity tags collapsed. The following shows the first domain entity node expanded (and the meta data node is collapsed again):

```xml
<?xml version="1.0" ?>
System-Link version="1.0" hostVersion="1.0">
   <LoginResponse>
      <SessionHandle value="61d36b:e54302f9b5:-7fc8" />
   </LoginResponse>
   <Response sessionHandle="61d36b:e54302f9b5:-7fc8" workHandle="61d36b:e54302f9b5:-7fc6">
      <QueryListResponse name="queryList_Item_Sample" requestedDomainClass="com.mapics.epdm.Item"
       moreResults="false">
      +  <MetaData>
      +  <DomainEntity domainClass="com.mapics.epdm.Item">
         <Key>
            <Property path="item">
               <Value><![CDATA[DESKTOP PC]]></Value>
            </Property>
         </Key>
            <Property path="item">
               <Value><![CDATA[DESKTOP PC]]></Value>
            </Property>
            <Property path="description">
               <Value><![CDATA[CONFIGURED DESKTOP PC  - PJ]]></Value>
            </Property>
            <Property path="itemClass">
               <Value><![CDATA[01]]></Value>
            </Property>
            <Property path="itemType">
               <Value><![CDATA[1]]></Value>
            </Property>
            <Property path="drawingNumber">
               <Value><![CDATA[749-E2]]></Value>
            </Property>
            <Property path="stockingUm">
               <Value><![CDATA[EA]]></Value>
            </Property>
            <Property path="department">
               <Value><![CDATA[2]]></Value>
            </Property>
         </DomainEntity>
```

Each domain entity tag starts with the key of the record. The fragment for this is shown below:

```xml
<Key>
   <Property path="item">
      <Value><![CDATA[DESKTOP PC]]></Value>
   </Property>
</Key>
```

The sequence of statements illustrates the symmetrical structure of XML. Each tag has a beginning and an end. In this case, you see the **Key** tag to begin the list of key properties and the **Key** tag again at the bottom to signal the end of the key properties.

Between the key tags, you see a pair of **property** tags. The property path is the attribute (notice the similarity to path in the meta data). In our example, this is **item** because, as we know, that is the key of the Item object. For a multi-part key, we would have seen multiple pairs of property tags, each showing a different part of the key. For example, if the object had been Item Revision we would have seen three pairs of property tags: one for **Site**, a second for **Item** and a third for **Revision**.

The value of a property is shown between the **value** tags. Since the first record's key is **DESKTOP PC** that is the value.

The key element defines the record that is to follow. The first record is as follows (the headings are shown for convenience):

Item	Description	Class	I/T	Drawing	Stk UM	Dept
DESKTOP PC	Configured Desktop PC – Pj	01	1	749-E2	EA	2

Each attribute will be described using properties. For example, the property for the first attribute is shown below:
- **<Property path="item">**
  - **<Value><![CDATA[DESKTOP PC]]></Value>**
  </Property>

Of course, this is the same as the key field. However, there is no **Key** tag here. The key tag defines the record. The **Property** tag identifies the data. System-Link will not assume that key data is to be displayed. Often it is useful to not display the key…or not all of it. For example, many objects have company as part of the key. Organizations with only one company usually choose not to display it.

All six of the remaining property elements (not shown) are listed: one after the other. Following the last one is the closing **Domain Entity** tag, signaling the end of the record.

</DomainEntity>

The next record then starts with a new Domain Entity tag. This pattern is repeated for the next record, and the next, until the last one. At the end of the System-Link response, you see the final closing Domain Entity tag…

</DomainEntity>

…followed by the closing Query list response tag…

</QueryListResponse>

…followed by the closing XML Response tag…

</Response>

…followed by the very end: the closing System-Link tag:

</System-Link>

Fetch Next Request

A *queryList* request will return a list of records up to the number specified in the *maxReturned* attribute. For example, if there had been fifteen records in our list, instead of six, the response would have returned only ten – the *maxReturned* value in our request. The *moreResults="false"* attribute, which means there were no more records, would have been *"true"*, meaning that more records were available.

Usually we need to provide the user with the ability to get additional records. Allowing the user to repeat the *queryList* request would simply return the same data a second time. Instead, we need the ability to repeat the *queryList* request (same object, same attributes, etc.) with the stipulation that it returns the next set of records. This is the purpose of the *FetchNext* tag.

Below is a request for the first ten items in the item master. The PQL contains no WHERE clause, meaning there is no subset. This is equivalent to the *all records* designation in the XA Browser.

```
<QueryList name='queryList_Item_General' domainClass='com.mapics.epdm.Item' includeMetaData='true'
maxReturned='10'>
        <Pql>
              <![CDATA[SELECT item,description,itemClass,itemType,drawingNumber,
                  stockingUm,department ORDER BY item]]>
        </Pql>
```

The response shows the session handle, the work handle and the moreResults indicator = "true"

```
<Response sessionHandle="-63b0faa5:e53fe694be:-7f6e" workHandle="-63b0faa5:e53fe694be:-7f6c">
<QueryListResponse name="queryList_Item_General" requestedDomainClass="com.mapics.epdm.Item"
moreResults="true">
```

In order to get additional records we must create a new request using the FetchNext tag. The request must specify the same session handle and work handle (both shown in bold) as the response to the earlier request.

```
<?xml version='1.0'?>
<!DOCTYPE System-Link SYSTEM 'SystemLinkRequest.dtd'>
<System-Link>
    <Request sessionHandle='-63b0faa5:e53fe694be:-7f6e' workHandle='-63b0faa5:e53fe694be:-7f6c'
        broker='EJB' maxIdle='7000000'>
            <FetchNext name='second_group' queryName='queryList_Item_General'
              maxReturned='20'/>
    </Request>
</System-Link>
```

The FetchNext tag above has three attributes: a name, a queryName and maxReturned.

- name   *second_group*   The name allows the parser to pair the response with the *FetchNext* request.

- queryName   *queryList_Item_General* This tells the server to return the results of this previous query, starting with the next record.

- maxReturned   The server will not assume that the maximum is the same as the earlier request. In this case, we're assuming that if the user is not satisfied with ten records we'll be a little more aggressive with the next request and return twenty.

The FetchNext response is very close to the queryList response.

```
<Response sessionHandle="-63b0faa5:e53fe694be:-7f6e" workHandle="-63b0faa5:e53fe694be:-7f6c">
<FetchNextResponse name="second_group" queryName="queryList_Item_General" moreResults="true">
```

After the *FetchNextResponse* tag, the domain entity tags begin. The domain entity responses are identical to those in the original query list response. We will get twenty, because that was the maximum we asked for and the *moreResults="true"* tells us that we got the maximum…and there are still more for the asking.

Example 2. Query an object (Customer)

System-Link Request

The previous example discussed a number of features that also apply to this example. These features include the login tag, session handles, PQL or QueryDfn tags, and meta data. These topics will not be discussed in detail again. Please refer to the previous example for illustrations of these features.
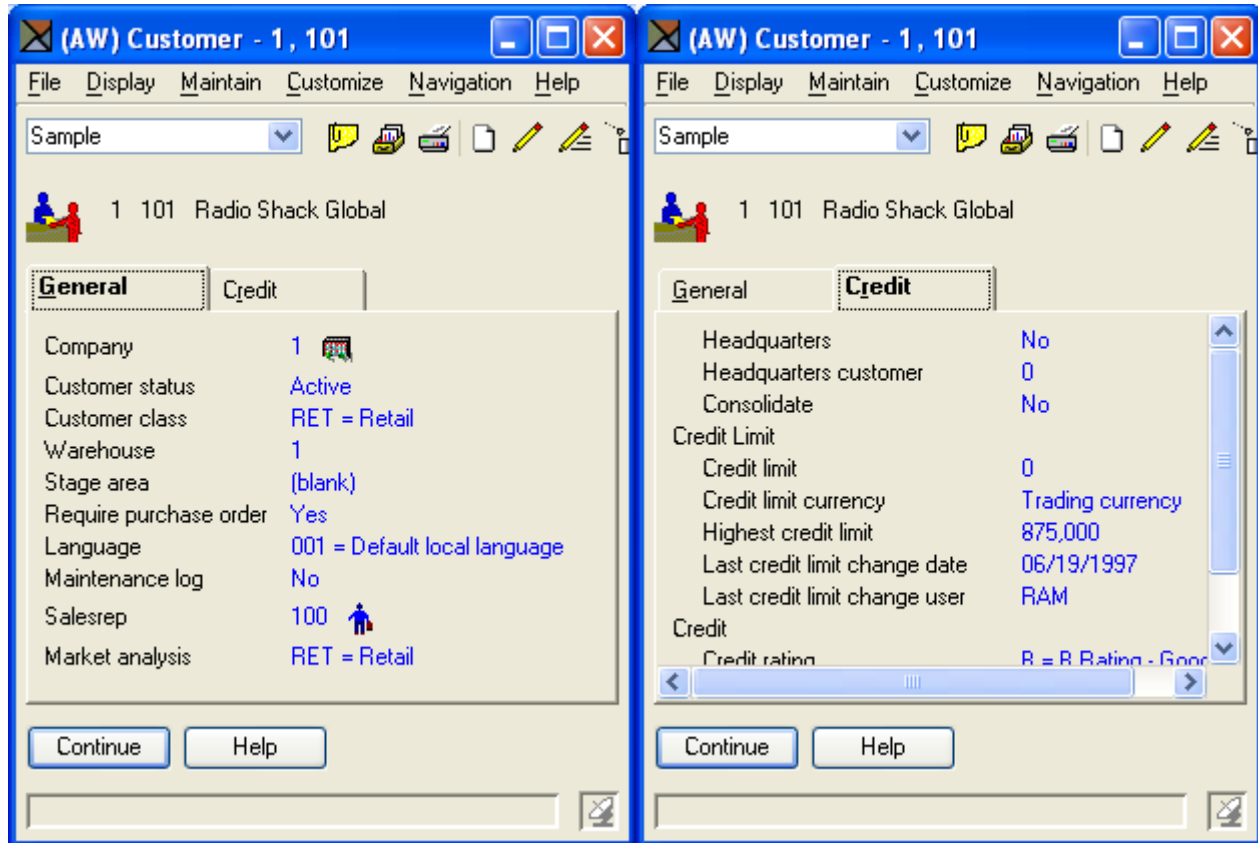
This example will request the record for one customer. Below we see details for one customer. The card file contains only two cards in order to simplify the example. Both cards are shown below:

The object has a two-part key:

Key field	Value
Company number	1
Customer number	101

The first card has six attributes. The second card has seven more, for a total of thirteen.

The System-Link request is shown below:

```
<Request sessionHandle='7f1ba3:e5347e3f6f:-7e13' workHandle='*new'
  broker='EJB' maxIdle='7000000'>
      <QueryObject name='queryObject_Customer_Sample' domainClass='com.mapics.csm.Customer'
       includeMetaData='true' >
         <Pql>
            <![CDATA[SELECT commissionablePercent,language,requireCreditCheck,
               highestCreditLimit,requirePurchaseOrder,customer,customerStatus,
               lastCreditLimitChangeDate,creditLimitCurrency,creditRating,
               creditLimit,customerClass,salesrep,customerName,company WHERE company=1 and
               customer=101]]>
         </Pql>
      </QueryObject>
   </Request>
</System-Link>
```

One difference you will note is that there is no login request tag. This is because we used the session handle provided by the System-Link response to an earlier login.

<Request sessionHandle='7f1ba3:e5347e3f6f:-7e13' workHandle='*new'
**   broker='EJB' maxIdle='7000000'>**

This saves time for the user (if there is one) as well as using system resources more efficiently. Other than the session handle, the rest of the request is similar to that already described in the previous example.

Query Object tag

**   <QueryObject name='queryObject_Customer_Sample' domainClass='com.mapics.csm.Customer'**
includeMetaData='true' >

This QueryObject tag is virtually identical to the QueryList tag. The only difference is that it doesn't have a maximum return value, because it will return only one record.

PQL tag

**   <Pql>**
**   <![CDATA[SELECT commissionablePercent,language,requireCreditCheck,**
**   highestCreditLimit,requirePurchaseOrder,customer,customerStatus,**
**   lastCreditLimitChangeDate,creditLimitCurrency,creditRating,**
**   creditLimit,customerClass,salesrep,customerName,company WHERE company=1 and**
**   customer=101]]>**
**   </Pql>**

Again, the differences between this and the QueryList PQL are minor. The **SELECT** clause lists fifteen attributes: the thirteen attributes on the two cards plus two more from the banner. (The third attribute in the banner, Customer name, is also on one of the cards.)

The **WHERE** clause contains the values for the key field. This will insure that only one record is retrieved.

There is no **ORDER BY** clause, since sort doesn't apply.

The remainder of the XML is composed of end tags.

Using custom definitions instead of PQL

In the PQL above, the SELECT clause lists each of the fields to be retrieved. The WHERE clause identifies the key field values of the record to be retrieved.

You can specify the key field values using the <Key> tag, as shown below. The <Key> tag would be used in place of the <PQL> tag.

```
<Key>
  <Property path='company'>
    <Value>1</Value>
  </Property>
  <Property path='customer'>
    <Value>101</Value>
  </Property>
</Key>
```

Now that System-Link knows which record you want, you need to specify the data to be retrieved. This can be done by referencing a card file, using the <CardFileDfnKey> tag immediately after the </Key> end tag. The card file is called Default in the following example:

```
<CardFileDfnKey clientClass='com.mapics.csm.Customer'><![CDATA[Default]]></CardFileDfnKey>
```

Everything from the <Key> start tag to the </CardFileDfnKey> end tag would replace the PQL (i.e., everything from the <PQL> start tag to the </PQL> end tag, inclusive). The rest of the XML is unchanged.

The advantage to this approach is that the System-Link results can be controlled very simply by customizing a card file. Thus, if an attribute is added to a card in the card file it will automatically be included in the next System-Link retrieval referencing that card file. The same goes for attributes removed from a card. Situations where this flexibility is too dynamic can still use PQL create "hardcoded" requests.

System-Link Response

The discussion below will concentrate on those areas where the response for the QueryObject request differs from that for the QueryList request. Thus, the Header and Response tags are skipped. All but one of the meta data nodes and some of the property path nodes have been collapsed. The key tags under the domain entity node show how multiple keys are handled.

```
    <QueryObjectResponse name="queryObject_Customer_Sample"
      requestedDomainClass="com.mapics.csm.Customer">
      <MetaData>
        +  <Descriptor type="label" path="commissionablePercent">
        +  <Descriptor type="label" path="language">
           <Descriptor type="label" path="requireCreditCheck">
           <![CDATA[Require credit check]]>
           </Descriptor>
        +  <Descriptor type="label" path="highestCreditLimit">
        +  <Descriptor type="label" path="requirePurchaseOrder">
        +  <Descriptor type="label" path="customer">
        +  <Descriptor type="label" path="customerStatus">
        +  <Descriptor type="label" path="lastCreditLimitChangeDate">
        +  <Descriptor type="label" path="creditLimitCurrency">
        +  <Descriptor type="label" path="creditRating">
        +  <Descriptor type="label" path="creditLimit">
        +  <Descriptor type="label" path="customerClass">
        +  <Descriptor type="label" path="salesrep">
        +  <Descriptor type="label" path="customerName">
        +  <Descriptor type="label" path="company">
      </MetaData>
    <DomainEntity domainClass="com.mapics.csm.Customer">
      <Key>
        <Property path="company">
           <Value>1</Value>
        </Property>
        <Property path="customer">
           <Value>101</Value>
        </Property>
      </Key>
      <Property path="commissionablePercent">
         <Value>0.253</Value>
      </Property>
      <Property path="language">
         <Value><![CDATA[001]]></Value>
      </Property>
      <Property path="requireCreditCheck">
         <Value>1</Value>
      </Property>
      <Property path="highestCreditLimit">
         <Value>875000</Value>
      </Property>
    +  <Property path="customerStatus">
      <Property path="lastCreditLimitChangeDate">
         <Value><![CDATA[19970619]]></Value>
      </Property>
    +  <Property path="creditLimitCurrency">
    +  <Property path="creditLimit">
    +  <Property path="customerClass">
    +  <Property path="customerName">
    +  <Property path="company">
    +  </DomainEntity>
    </QueryObjectResponse>
```

Query Object Response tag

<QueryObjectResponse name="queryObject_Customer_Sample"
requestedDomainClass="com.mapics.csm.Customer">

The QueryObjectResponse tag syntax is the same as the QueryListResponse. The only difference in this example is that it is using a different object (Customer instead of Item).

Meta Data

<Descriptor type="label" path="requireCreditCheck">
- <![CDATA[Require credit check]]>
</Descriptor>

A Query Object request is for a card file rather than a list. This is why the meta data returns labels instead of headings. The remaining differences are due to the object and attributes chosen, rather than for any intrinsic difference between QueryList and QueryObject.

Domain Entity tag – multiple keys

<DomainEntity domainClass="com.mapics.csm.Customer">
   <Key>
      <Property path="company">
         <Value>1</Value>
      </Property>
      <Property path="customer">
         <Value>101</Value>
      </Property>
   </Key>

Since this object has two key attributes, we see two property elements between the **Key** tags. Each shows the name of the property (i.e., company, customer) and its value (i.e., 1, 101). The value syntax illustrates a numeric value for both attributes.

Property tags – alphanumeric, numeric and boolean

The syntax varies slightly depending on the type of attribute. Attributes can be alphanumeric, numeric or boolean. A boolean attribute can have only two possibilities, one of which indicates the **true** condition and the other the **false** condition. Whether these values in the database are numeric alphanumeric is not significant. All the XML communicates is whether the value is true or false.
Thus, the value of an attribute can be of three types, as follows:

- Boolean: will always return a value of either 1 (true) or 0 (false). The property below illustrates that the condition of requireCreditCheck is true (i.e., a credit check is required):

    <Property path="requireCreditCheck">
      <Value>1</Value>
    </Property>

- Alphanumeric: can be any alphanumeric value, indicated by CDATA (i.e., character data). The property below illustrates that the value of **language** is **001**

    <Property path="language">
      <Value><![CDATA[001]]></Value>
    </Property>

Dates are passed as YYYYMMDD. The property below illustrates that the **last credit limit change date** was **June 19, 1997**:

```
<Property path="lastCreditLimitChangeDate">
   <Value><![CDATA[19970619]]></Value>
</Property>
```

- Numeric: can be any numeric data. The property below indicates that the *highest credit limit* is *875,000*.

```
<Property path="highestCreditLimit">
   <Value>8750000</Value>
</Property>
```

Decimal values are presented in the same way. The property below indicates that **the commissionable percent** is **0.253**. (It is up to the application to determine whether this is 25.3% or .253%.)

```
<Property path="commissionablePercent">
   <Value>0.253</Value>
</Property>
```

Results

The result of QueryObject is a list of attributes and values. All the attributes on all the cards will be included. The receiving program can format the data using a stylesheet (or other method) if required. The following table shows the System-Link response in our example transformed via a simple stylesheet.

Company	1
Customer	101
Commissionable percent	0.253
Language	001
Require credit check	true
Highest credit limit	875000
Require purchase order	true
Customer	101
Customer status	A
Last credit limit change date	0970619
Credit limit currency	true
Credit rating	B
Credit limit	0
Customer class	RET
Salesrep	100
Customer name	Radio Shack Global
Company	1

Example 3. Query multiple objects (Manufacturing order)

Often we want to query a complex object, such as an order. A complex object is one that has two or more objects related in a header/detail relationship. MO's, CO's, PO's and IFM transactions are all examples of complex objects.

In addition, it is often useful to query a collection of objects that are related, although not in a header/detail relationship. The following examples will illustrate this:

- Customers and CO's
- Vendors and PO's
- Items and MO's or schedules
- MO components and inventory transactions

System-Link allows you to request multiple objects at one time. For this example, we will request a manufacturing order with its components and operations, as well as the inventory transactions for the components. This will illustrate the technique for requesting multiple objects…in this case, four:

- Manufacturing Order
- MO Component
- MO Operation
- Inventory Transaction History

The easiest way to start this is to let XA provide the System-Link request. We have created a card file for an MO that contains four cards: one with MO attributes (only a few to keep things simple) plus two list cards – one for components and one for operations. Then we can ask for the System-Link request.

The following three cards show the card file:
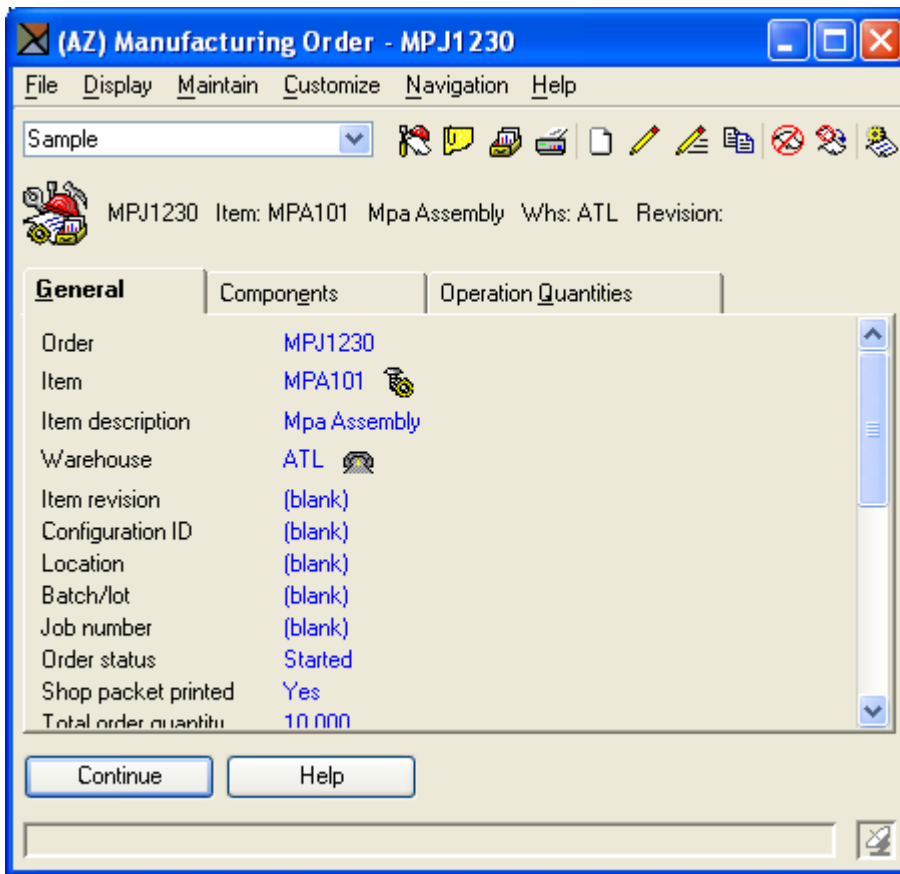
General card: This first card shows eight attributes. (The attributes in the banner are also on the card. We only count an attribute once.)
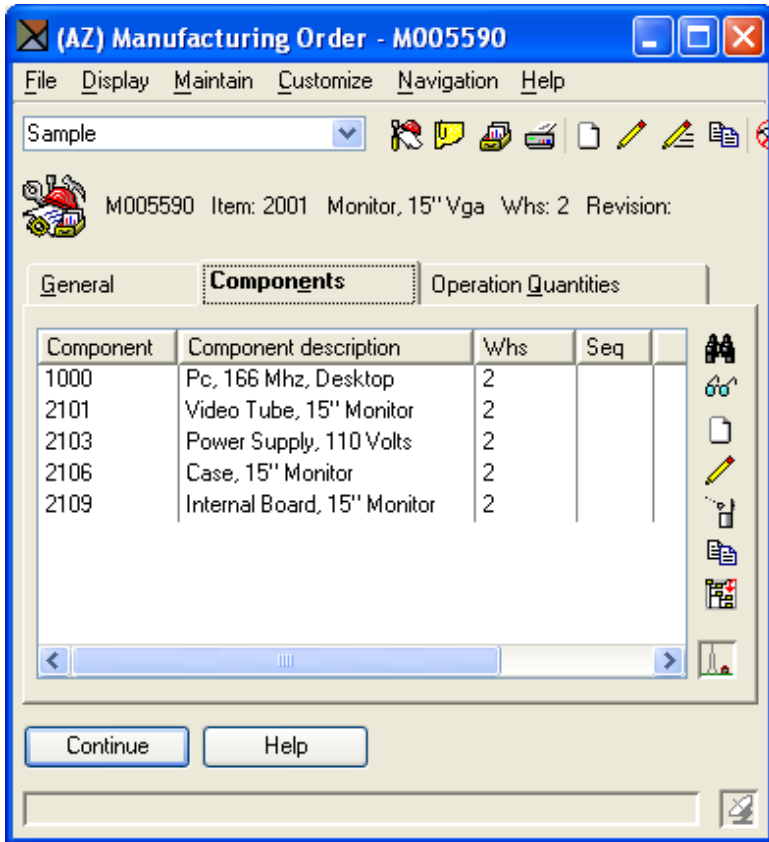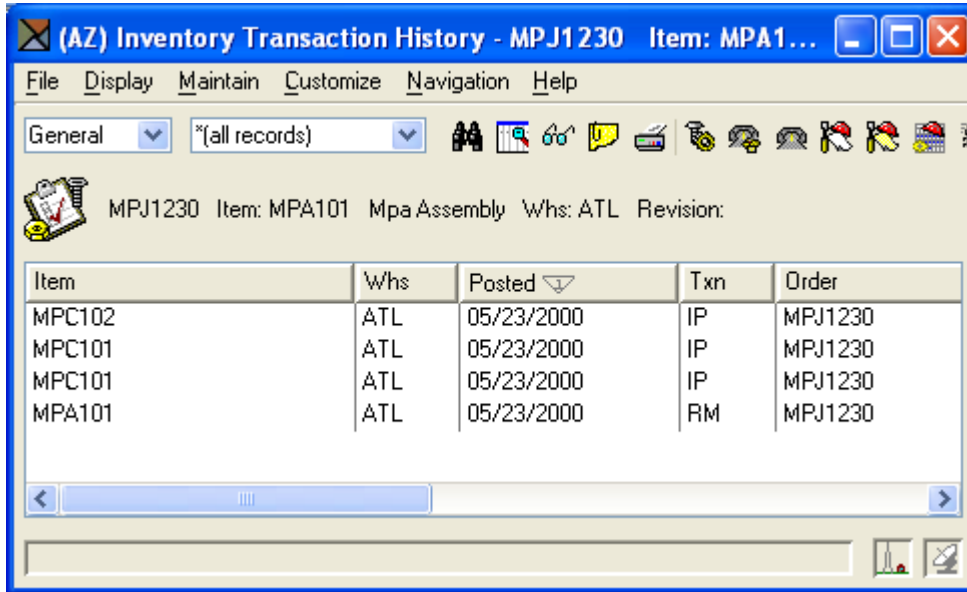
Components card: The second card shows four components, with four attributes (i.e., columns) each.
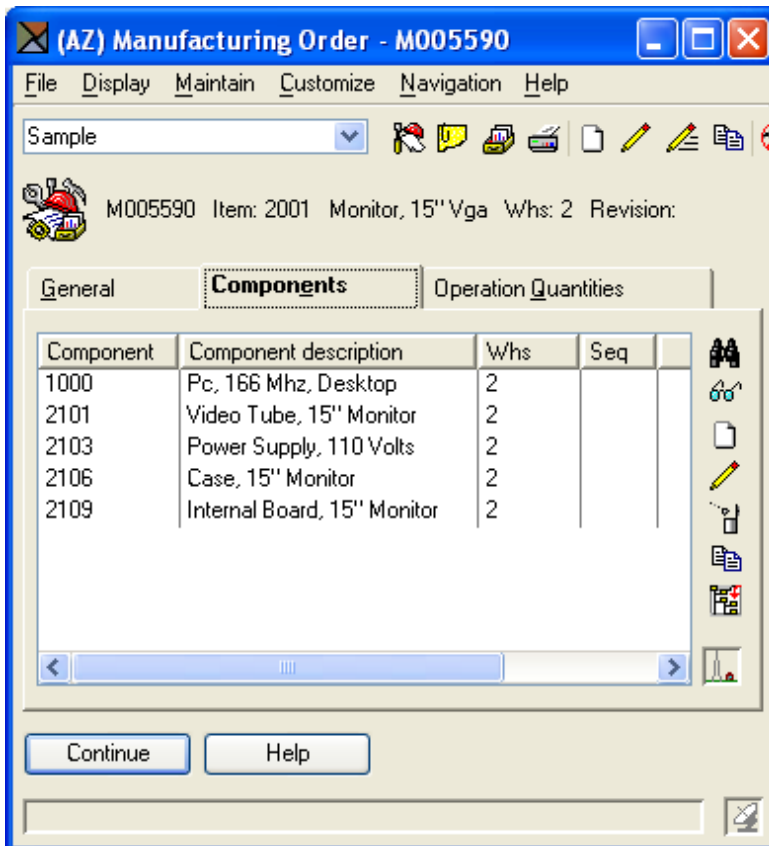
Operation Quantities card: The third card shows three operations, with four attributes each.

Inventory Transaction History: For this view we navigated from the first component on the Components card. The first component has two transactions with three attributes each. Other components may also have transactions.

The objective is to have a single XML request include:

- A QueryObject request for the manufacturing order and its eight attributes
- A QueryRelationship request for the components and their four attributes
- A nested QueryRelationship request for the transactions for each component, with their four attributes
- A QueryRelationship request for the operations and their four attributes

System-Link Request

From this card file, we will use the menu option to get the System-Link request. The System-Link request is shown below. Note that this card file does not include inventory transactions:

```xml
<?xml version='1.0'?>
<!DOCTYPE System-Link SYSTEM 'SystemLinkRequest.dtd'>
<System-Link>
    <Request sessionHandle='7f1ba3:e5347e3f6f:-7767' workHandle='*new' broker='EJB' maxIdle='900000'>
        <QueryObject name='queryObject_ManufacturingOrder_Sample'
          domainClass='com.mapics.obpm.ManufacturingOrder' includeMetaData='true' >
            <Pql>
                <![CDATA[SELECT item,totalOrderQuantityCalculated,relatedItem.stockingUm,
                  dueDate,order,itemWarehouse,orderStatus,itemDescription WHERE order='MPJ1230']]>
            </Pql>
            <QueryRelationship name='queryRelationship_Components' relationshipName='relatedMoMaterials'
             includeMetaData='true' maxReturned='10'>
                <Pql>
                    <![CDATA[SELECT
                      componentItem,componentItemDescription,standardQuantity,relatedItem.stockingUm WHERE
                      (order='MPJ1230')  ORDER BY
                      order,componentItem,componentWarehouse,userSequence,sequenceNumber]]>
                </Pql>
            </QueryRelationship>
            <QueryRelationship name='queryRelationship_OperationQuantities'
             relationshipName='relatedMoOperations' includeMetaData='true' maxReturned='10'>
                <Pql>
                    <![CDATA[SELECT operationSequence,description,facilityActual,remainingQuantityCalculated
                      WHERE (order='MPJ1230') ORDER BY order,operationSequence]]>
                </Pql>
            </QueryRelationship>
        </QueryObject>
    </Request>
</System-Link>
```

Query Relationship tag

The first part of the System-Link request matches what we have seen before. It includes a header section, a session section, the query object section to retrieve the MO and the PQL to get the attributes for the manufacturing order. So far so good. Then it gets interesting.

If all you wanted were the attributes for the MO object then you would see the Query Object end tag (</QueryObject>) after the PQL. Instead, we see a Query Relationship section as follows:

```
        <QueryRelationship name='queryRelationship_Components'
          relationshipName='relatedMoMaterials' includeMetaData='true' maxReturned='10'>
            <Pql>
                <![CDATA[SELECT
                  componentItem,componentItemDescription,standardQuantity,relatedItem.stockingUm
                  WHERE (order='MPJ1230')  ORDER BY
                  order,componentItem,componentWarehouse,userSequence,sequenceNumber]]>
            </Pql>
        </QueryRelationship>
```

This is the System-Link request that will retrieve the list of components for the MO. You can tell this from the name: **queryRelationship_Components**. Essentially, this is the name of the list of components that will be retrieved. The relationship name (vs. query relationship name) is the relationship between the MO object and the MO Component object. This relationship provides the link fields and other information the system needs to retrieve the data.

The maxReturned attribute specifies that a maximum of ten component records will be retrieved. Unlike a QueryList, however, this is a list query inside an object query. Therefore, you will not be able to retrieve additional records using a FetchNext tag.

With only one chance to retrieve the data, the temptation would be to make the maxReturned value high. This could be a problem, because there is no limit to the number of queryRelationships that can be in a single request. In addition, there is no limit to the number of nesting levels. This means that the amount of data retrieved by a single request can be enormous, and the user would probably not be willing to wait for the time it would take to assemble the data.

The maxReturned attribute is the means of controlling this. The maximum number of records should be low enough to provide adequate performance given the entire request (i.e., the total number of queryRelationships and their maxReturned values). At the same time, it should be large enough to provide value to the user.

The PQL defines the MO Component data that will be retrieved:

- SELECT   =The attributes, like columns in a view. These will be MO Component attributes
- WHERE   =The selection criteria, similar to a subset. We will get all the components for MO MPJ1230.
- ORDER BY   =The sort order for the component list. Each attribute is ascending unless it has **DESC** after it.

Note: PQL must be used in a QueryRelationship request. You cannot bypass PQL using <QueryDfn>, as we did in Example 1 (Item list).

The query relationship request closes with the end tag: **</QueryRelationship>**

The System-Link request then shows another query relationship for the operations, followed by all the end tags.

If we submitted this request we would get eight attribute labels and values for the MO, four attribute headings and values for each of the four components, and four attribute headings and values for each of the three operations. However, that is not enough. We still need the inventory transaction information.

To get the System-Link request for the inventory transactions we created a new card file for MO Component. The new card file has two cards: one with attributes (which we don't really need) and one with a list of inventory transactions. This list card is the one that will cause the System-Link request to include a query relationship.
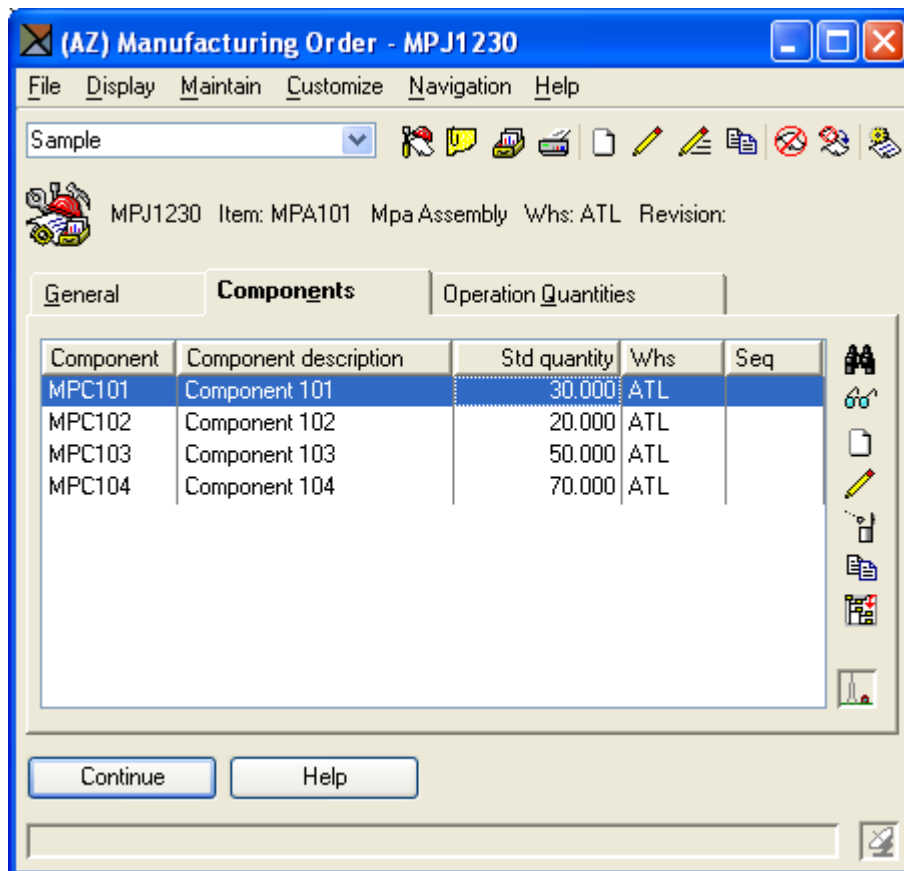
To display the MO Components card file we can display the details for one of the components on the list card in the MO object. We displayed the details for component item MPC101, below. The Transactions card looks like this:
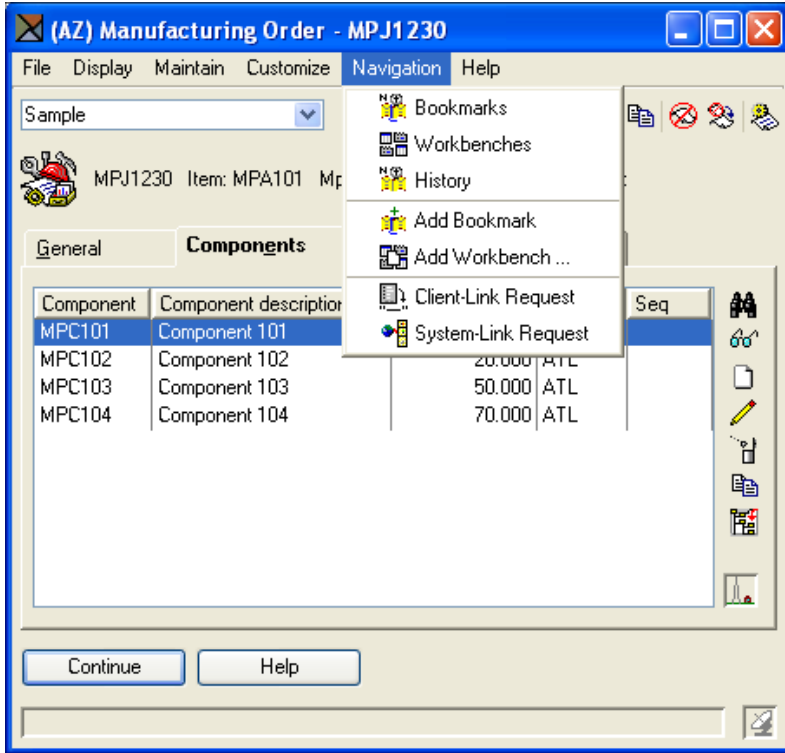
The Transaction card shows three attributes columns in the list. These are the attributes that we want to include in a nested Query Relationship tag in our request.

Using the option from the Navigation menu, we can display the System-Link request for the MO Component details. The part we care about is the query relationship section:

<QueryRelationship name='queryRelationship_Transactions' relationshipName='relatedInvTxnHistory'
includeMetaData='true' maxReturned='10'>
    <Pql>
      <![CDATA[SELECT item,order,transactionCode,transactionQuantity,unitOfMeasure WHERE
        (order='MPJ1230' and item='MPC101' and warehouse='ATL' and MOUserSequenceOrPOLine=''
        and itemSequence=2) ORDER BY postedDate desc,postedTime desc]]>
    </Pql>
</QueryRelationship>

You can see that this fragment matches the list we had displayed in the Browser. Note that the WHERE clause specifies the MO component item (MPC101) whose card file we were on. This will be a problem we need to fix later.

This section starts with a query relationship tag and ends with a query relationship tag. All query relationships work this way, including MO component. The way we specify a nested relationship is by putting one query relationship inside the other, i.e., between the beginning and ending tags. We will copy this entire section – everything between the starting and ending tags plus the tags themselves – and then go to the earlier fragment to see where to paste it.

Listed below is the query relationship System-Link request fragment for the MO Components. Note the beginning and ending tags:

<QueryRelationship name='queryRelationship_Components' relationshipName='relatedMoMaterials'
includeMetaData='true' maxReturned='10'>
    <Pql>
      <![CDATA[SELECT
      componentItem,componentItemDescription,standardQuantity,relatedItem.stockingUm WHERE
      (order='MPJ1230') ORDER BY
      order,componentItem,componentWarehouse,userSequence,sequenceNumber]]>
    </Pql>
</QueryRelationship>

We will paste the query relationship request fragment for inventory transactions into the query relationship request fragment for components immediately *before* the end tag. It should look like this, with the nested fragment in **bold**:

<QueryRelationship name='queryRelationship_Components' relationshipName='relatedMoMaterials'
 includeMetaData='true' maxReturned='10'>
    <Pql>
      <![CDATA[SELECT componentItem,componentItemDescription,standardQuantity,relatedItem.stockingUm
      WHERE (order='MPJ1230') ORDER BY
      order,componentItem,componentWarehouse,userSequence,sequenceNumber]]>
    </Pql>
    <QueryRelationship name='queryRelationship_Transactions' relationshipName='relatedInvTxnHistory'
     includeMetaData='true' maxReturned='10'>
      <Pql>
       <![CDATA[SELECT item,order,transactionCode,transactionQuantity,unitOfMeasure WHERE
       (order='MPJ1230' and item='MPC101' and warehouse='ATL' and MOUserSequenceOrPOLine=''
       and itemSequence=2) ORDER BY postedDate desc,postedTime desc]]>
      </Pql>
    </QueryRelationship>

</QueryRelationship>

At this point, we are almost done with our request. The only thing left to fix is the WHERE clause in the nested fragment. Left unchanged we would only see the inventory transactions for item MPC101. We want to see the transaction records for all the items. To do this we need to understand both the way the nested query relationships are processed as well as the link data.

When the system processes the query relationship for MO Component, it will retrieve the first record in the list. Before it looks for a second one, it processes the nested query relationship. This means that it gets the entire list of inventory transactions for the first component before looking for the second component. After the second component, it runs the transaction query relationship again. It keeps doing this until it has processed the last transaction of the last component. Then it proceeds to the query relationship for the MO Operation object.

The PQL in the nested relationship contains a WHERE clause that identifies the transaction records that will be selected. Right now, these are hard-coded to look for the record we had displayed (the card file) when we asked for the System-Link request from the Navigation menu:

Attribute	Value
order	MPJ1230
item	MPC101
warehouse	ATL
MOUserSequenceOrPOLine	(blank)
itemSequence	2

What we want to do is to have these values change depending on the component record. The way to do this is to use System-Link substitution variables. You can see the names of the attributes in the component query relationship that contain the same values as those in the transaction query relationship. Putting an ampersand (&) around the attribute name makes it a System-Link substitution variable. The following table shows the association:

Attribute	Value	Substitution variable
order	MPJ1230	&order&
item	MPC101	&componentItem&
warehouse	ATL	&componentWarehouse&
MOUserSequenceOrPOLine	(blank)	&userSequence&
itemSequence	2	&sequenceNumber&

The substitution variables come from the component record, but will be used in the transaction query relationship. For example, the attribute item in the transaction record is the attribute **componentItem** in the component record. This is how the link is accomplished. We will substitute the variables containing component values in the WHERE clause of the transaction PQL thus:

> <Pql>
>   <![CDATA[SELECT item,order,transactionCode,transactionQuantity,unitOfMeasure WHERE
>   (order=&order& and item=**&componentItem&** and warehouse=**&componentWarehouse&** and
>   MOUserSequenceOrPOLine=**&userSequence&** and itemSequence=**&sequenceNumber&**) ORDER
>   BY postedDate desc,postedTime desc]]>
> </Pql>

When the first component record is read, each of these five attributes will contain a value. For example, the **componentItem** will be MPC101. The nested query relationship will then execute, using a WHERE value of MPC101 for the item and resulting in the transactions for MPC101. After the second component record is read, the value of **componentItem** will be MPC102. The WHERE clause in the nested query relationship will result in the transactions for item MPC102, etc.

The final fragment for the request, including the nested query relationship with substitutions, is as follows:

```xml
<?xml version='1.0'?>
<!DOCTYPE System-Link SYSTEM 'SystemLinkRequest.dtd'>
<System-Link>
    <Request sessionHandle='7f1ba3:e5347e3f6f:-7767' workHandle='*new' broker='EJB' maxIdle='900000'>
        <QueryObject name='queryObject_ManufacturingOrder_Sample'
          domainClass='com.mapics.obpm.ManufacturingOrder' includeMetaData='true' >
          <Pql>
             <![CDATA[SELECT item,totalOrderQuantityCalculated,relatedItem.stockingUm,
                dueDate,order,itemWarehouse,orderStatus,itemDescription WHERE order='MPJ1230']]>
          </Pql>
          <QueryRelationship name='queryRelationship_Components' relationshipName='relatedMoMaterials'
           includeMetaData='true' maxReturned='10'>
             <Pql>
                <![CDATA[SELECT
                  componentItem,componentItemDescription,standardQuantity,relatedItem.stockingUm WHERE
                  (order='MPJ1230')  ORDER BY
                  order,componentItem,componentWarehouse,userSequence,sequenceNumber]]>
             </Pql>
                <QueryRelationship name='queryRelationship_Transactions'
                 relationshipName='relatedInvTxnHistory' includeMetaData='true'  maxReturned='10'>
                   <Pql>
                      <![CDATA[SELECT item,order,transactionCode,transactionQuantity,unitOfMeasure WHERE
                       (order=&order& and item=&componentItem& and warehouse=&componentWarehouse& and
                       MOUserSequenceOrPOLine=&userSequence& and itemSequence=&sequenceNumber&)
                       ORDER BY postedDate desc,postedTime desc]]>
                   </Pql>
                </QueryRelationship>
          </QueryRelationship>
        <QueryRelationship name='queryRelationship_OperationQuantities'
         relationshipName='relatedMoOperations' includeMetaData='true' maxReturned='10'>
           <Pql>
              <![CDATA[SELECT operationSequence,description,facilityActual,remainingQuantityCalculated
               WHERE (order='MPJ1230') ORDER BY order,operationSequence]]>
           </Pql>
        </QueryRelationship>
        </QueryObject>
    </Request>
</System-Link>
```

System-Link Response

The System-Link response, with nodes collapsed, is shown below. Note the three lines in bold. You can see that the manufacturing order is queried first, followed by the components and then the operations. The inventory transactions are not visible, because they are nested within the components node.

```
<?xml version="1.0" ?>
<System-Link version="1.0" hostVersion="1.0">
    <Response sessionHandle="7f1ba3:e539f9a28b:-7f0b" workHandle="7f1ba3:e539f9a28b:-7ec0">
      <QueryObjectResponse
       name="queryObject_ManufacturingOrder_Sample"requestedDomainClass="com.mapics.obpm.Manufacturi
       ngOrder">
        <MetaData>
        <DomainEntity domainClass="com.mapics.obpm.ManufacturingOrder">
        <QueryRelationshipResponse name="queryRelationship_Components" moreResults='false'>
        <QueryRelationshipResponse name="queryRelationship_OperationQuantities"
         moreResults='false'>
      </QueryObjectResponse>
    </Response>
</System-Link>
```

Below you see the same System-Link response, only the components node has been expanded (shown in bold). Notice that the query relationship for components results in a list of four domain entity tags (i.e., four records meaning four components, as we saw in the card). After each one, you see a query relationship for transactions, indicating a list of transactions for each component record. (Actually, you have to expand the node to see whether there are any transactions. There may only be meta data.)

```
<?xml version="1.0" ?>
<System-Link version="1.0" hostVersion="1.0">
    <Response sessionHandle="7f1ba3:e539f9a28b:-7f0b" workHandle="7f1ba3:e539f9a28b:-7ec0">
      <QueryObjectResponse
       name="queryObject_ManufacturingOrder_Sample"requestedDomainClass="com.mapics.obpm.Manufacturi
       ngOrder">
        <MetaData>
        <DomainEntity domainClass="com.mapics.obpm.ManufacturingOrder">
        <QueryRelationshipResponse name="queryRelationship_Components" moreResults='false'>
          <MetaData>
          <DomainEntity domainClass="com.mapics.obpm.MoComponent">
          <QueryRelationshipResponse name="queryRelationship_Transactions" moreResults='false'>
          <DomainEntity domainClass="com.mapics.obpm.MoComponent">
          <QueryRelationshipResponse name="queryRelationship_Transactions" moreResults='false'>
          <DomainEntity domainClass="com.mapics.obpm.MoComponent">
          <QueryRelationshipResponse name="queryRelationship_Transactions" moreResults='false'>
          <DomainEntity domainClass="com.mapics.obpm.MoComponent">
          <QueryRelationshipResponse name="queryRelationship_Transactions" moreResults='false'>
        </QueryRelationshipResponse>
        <QueryRelationshipResponse name="queryRelationship_OperationQuantities" moreResults='false'>
      </QueryObjectResponse>
    </Response>
</System-Link>
```

Here we have expanded the first attribute of the first component record to show that it is for component item MPC101.

```xml
<?xml version="1.0" ?>
<System-Link version="1.0" hostVersion="1.0">
    <Response sessionHandle="7f1ba3:e539f9a28b:-7f0b" workHandle="7f1ba3:e539f9a28b:-7ec0">
        <QueryObjectResponse
          name="queryObject_ManufacturingOrder_Sample"requestedDomainClass="com.mapics.obpm.Manufacturi
          ngOrder">
            <MetaData>
            <DomainEntity domainClass="com.mapics.obpm.ManufacturingOrder">
            <QueryRelationshipResponse name="queryRelationship_Components" moreResults='false'>
                <MetaData>
                <DomainEntity domainClass="com.mapics.obpm.MoComponent">
                    <Key>
                    <Property path="componentItem">
                        <Value><![CDATA[MPC101]]></Value>
                    </Property>
                    <Property path="componentItemDescription">
                    <Property path="standardQuantity">
                    <Property path="relatedItem.stockingUm">
                    <Property path="order">
                    <Property path="componentWarehouse">
                    <Property path="userSequence">
                    <Property path="sequenceNumber">
                </DomainEntity>
                <QueryRelationshipResponse name="queryRelationship_Transactions" moreResults='false'>
                <DomainEntity domainClass="com.mapics.obpm.MoComponent">
                <QueryRelationshipResponse name="queryRelationship_Transactions" moreResults='false'>
                <DomainEntity domainClass="com.mapics.obpm.MoComponent">
                <QueryRelationshipResponse name="queryRelationship_Transactions" moreResults='false'>
                <DomainEntity domainClass="com.mapics.obpm.MoComponent">
                <QueryRelationshipResponse name="queryRelationship_Transactions" moreResults='false'>
            </QueryRelationshipResponse>
            <QueryRelationshipResponse name="queryRelationship_OperationQuantities" moreResults='false'>
        </QueryObjectResponse>
    </Response>
</System-Link>
```

Continuing to expand the nodes, we can see that the query transaction for the first component, MPC101, has results in two transaction records.

```xml
<?xml version="1.0" ?>
<System-Link version="1.0" hostVersion="1.0">
    <Response sessionHandle="7f1ba3:e539f9a28b:-7f0b" workHandle="7f1ba3:e539f9a28b:-7ec0">
        <QueryObjectResponse
          name="queryObject_ManufacturingOrder_Sample"requestedDomainClass="com.mapics.obpm.Manufacturi
          ngOrder">
            <MetaData>
            <DomainEntity domainClass="com.mapics.obpm.ManufacturingOrder">
            <QueryRelationshipResponse name="queryRelationship_Components" moreResults='false'>
                <MetaData>
                <DomainEntity domainClass="com.mapics.obpm.MoComponent">
                    <Key>
                    <Property path="componentItem">
                        <Value><![CDATA[MPC101]]></Value>
                    </Property>
                    <Property path="componentItemDescription">
                    <Property path="standardQuantity">
                    <Property path="relatedItem.stockingUm">
                    <Property path="order">
                    <Property path="componentWarehouse">
                    <Property path="userSequence">
                    <Property path="sequenceNumber">
                </DomainEntity>
                <QueryRelationshipResponse name="queryRelationship_Transactions" moreResults='false'>
                    <MetaData>
                    <DomainEntity domainClass="com.mapics.mm.InventoryTransactionHistory'>
                    <DomainEntity domainClass="com.mapics.mm.InventoryTransactionHistory">
                </QueryRelationshipResponse>
                <DomainEntity domainClass="com.mapics.obpm.MoComponent">
                <QueryRelationshipResponse name="queryRelationship_Transactions" moreResults='false'>
                <DomainEntity domainClass="com.mapics.obpm.MoComponent">
                <QueryRelationshipResponse name="queryRelationship_Transactions" moreResults='false'>
                <DomainEntity domainClass="com.mapics.obpm.MoComponent">
                <QueryRelationshipResponse name="queryRelationship_Transactions" moreResults='false'>
            </QueryRelationshipResponse>
            <QueryRelationshipResponse name="queryRelationship_OperationQuantities" moreResults='false'>
        </QueryObjectResponse>
    </Response>
</System-Link>
```

The first transaction record for component item MPC101 has a transaction code of IP and a quantity of 5.000 EA.

```xml
<?xml version="1.0" ?>
<System-Link version="1.0" hostVersion="1.0">
    <Response sessionHandle="7f1ba3:e539f9a28b:-7f0b" workHandle="7f1ba3:e539f9a28b:-7ec0">
        <QueryObjectResponse name="queryObject_ManufacturingOrder_Sample"requestedDomainClass=
        "com.mapics.obpm.ManufacturingOrder">
          <MetaData>
          <DomainEntity domainClass="com.mapics.obpm.ManufacturingOrder">
          <QueryRelationshipResponse name="queryRelationship_Components" moreResults='false'>
            <MetaData>
            <DomainEntity domainClass="com.mapics.obpm.MoComponent">
                <Key>
                <Property path="componentItem">
                    <Value><![CDATA[MPC101]]></Value>
                </Property>
                <Property path="componentItemDescription">
                <Property path="standardQuantity">
                <Property path="relatedItem.stockingUm">
                <Property path="order">
                <Property path="componentWarehouse">
                <Property path="userSequence">
                <Property path="sequenceNumber">
            </DomainEntity>
            <QueryRelationshipResponse name="queryRelationship_Transactions" moreResults='false'>
                <MetaData>
                <DomainEntity domainClass="com.mapics.mm.InventoryTransactionHistory">
                  <Key>
                  <Property path="item">
                  <Property path="order">
                  <Property path="transactionCode">
                      <Value><![CDATA[IP]]></Value>
                  <Property>
                  <Property path="transactionQuantity">
                      <Value>5.000</Value>
                  <Property>
                  <Property path="unitOfMeasure">
                      <Value><![CDATA[EA]]></Value>
                  <Property>
                  <Property path="postedDate">
                  <Property path="postedTime">
                </DomainEntity>
                <DomainEntity domainClass="com.mapics.mm.InventoryTransactionHistory">
            </QueryRelationshipResponse>
            <DomainEntity domainClass="com.mapics.obpm.MoComponent">
            <QueryRelationshipResponse name="queryRelationship_Transactions" moreResults='false'>
            <DomainEntity domainClass="com.mapics.obpm.MoComponent">
            <QueryRelationshipResponse name="queryRelationship_Transactions" moreResults='false'>
            <DomainEntity domainClass="com.mapics.obpm.MoComponent">
            <QueryRelationshipResponse name="queryRelationship_Transactions" moreResults='false'>
        </QueryRelationshipResponse>
        <QueryRelationshipResponse name="queryRelationship_OperationQuantities" moreResults='false'>
    </QueryObjectResponse>
  </Response>
</System-Link>
```

If we expanded the entire request we would see all the transactions for all the components, as well as all the operations, plus the attributes for the manufacturing order.

Collapsing and expanding nodes is simply an Internet browser (user interface) convenience. The System-Link response will send all the data (all nodes "expanded") without any special programmatic controls.

Compare the results with what we expected:

- A QueryObject request for the manufacturing order and its eight attributes
- A QueryRelationship request for the components and their four attributes
- A nested QueryRelationship request for the transactions for each component, with their four attributes
- A QueryRelationship request for the operations and their four attributes

Order	MPJ1230
Item	MPA101
Total order quantity	10.000
Stocking UM	EA
Due date	1000506
Warehouse	ATL
Order status	40
Item description	MPA ASSEMBLY

Component	Component description	Std quantity	Stk UM
MPC101	COMPONENT 101	30.000	EA

Item	Order	Txn	Quantity	UM
MPC101	MPJ1230	IP	5.000	EA
MPC101	MPJ1230	IP	10.000	EA

Component	Component description	Std quantity	Stk UM
MPC102	COMPONENT 102	20.000	EA

Item	Order	Txn	Quantity	UM
MPC102	MPJ1230	IP	17.000	EA

Component	Component description	Std quantity	Stk UM
MPC103	COMPONENT 103	50.000	EA

Component	Component description	Std quantity	Stk UM
MPC104	COMPONENT 104	70.000	EA

Oper	Description	Facility	Remaining
0010	OPERATION 10	1000	-
0020	OPERATION 20	2000	-
0030	OPERATION 30	3000	-

Example 4. Create or copy a simple object (Warehouse)

Up to now, we have looked at ways to get data out of your system. Now we will look at ways to put data into it. The first maintenance example is for a warehouse. We will create a warehouse called WH1 with the following properties:

Warehouse number	WH1
Description	Southeast Shipping Center
Type	True (Controlled)
Site	RON
Shipping calendar	P1

The create action is shown below, with reason code *ABC*. Login and request tags have been omitted for clarity:

```
<Create name='newWarehouse' domainClass='com.mapics.mm.Warehouse' retainResult='false'>
    <ReasonCode><![CDATA[ABC]]></ReasonCode>
    <DomainEntity>
        <Key>
            <Property path="warehouse"><Value><![CDATA[WH1]]></Value></Property>
        </Key>
        <Property path="description"><Value><![CDATA[Southeast Shipping Center]]></Value>
        </Property>
        <Property path="warehouseType"><Value>true</Value></Property>
        <Property path="site"><Value><![CDATA[RON]]></Value></Property>
        <Property path="shippingCalendar"><Value><![CDATA[P1]]></Value></Property>
    </DomainEntity>
</Create>
```

<Create> and <DomainEntity> tags

As you would expect, the create action begins with the <Create> tag. The <Create> tag must be followed immediately by the <DomainEntity> tag. (Exception: see ReasonCode tag, below.) Similarly, the </DomainEntity> end tag must be followed immediately by the </Create> end tag. These two tags bracket each record that is to be created. You can create multiple records in one document, but they must each have their own <Create> and <DomainEntity> tags.

```
<Create name='newWarehouse' domainClass='com.mapics.mm.Warehouse' retainResult='false'>
    <DomainEntity>
```

The name must be unique in this document. If you create multiple records at once, each one must have its own name. We will see how this can be used when we create warehouse addresses.
The domainClass tells System-Link that we're creating a warehouse record.
We will discuss retainResult when we create warehouse addresses.

<ReasonCode> tag

When reason tracking is turned on for an object a reason code must be entered when creating, updating or deleting the object. The <ReasonCode> tag is placed in-between the <Create> (or <Update> or <Delete>) and <DomainEntity> tags. The <ReasonCode> tag is followed by the reason code value and the </ReasonCode> end tag.

Other tags

The rest of the request in the create action should be familiar. The key, property and value tags work in exactly the same way in maintenance as they do in inquiry.

Using a template

You can automatically populate fields by using the <ApplyTemplate> tag to reference a template. The following example creates a warehouse record using the <ApplyTemplate> tag, shown in bold. If you compare this to the previous illustration, you can see that we have eliminated the System-Link property tags for warehouse type, shipping calendar and site because these values will be supplied by the template:

```
<Create name='newWarehouse' domainClass='com.mapics.mm.Warehouse' retainResult='false'>
    <ApplyTemplate clientClass='com.mapics.mm.Warehouse'>
        <![CDATA[Sample]]>
    </ApplyTemplate>
    <DomainEntity>
        <Key>
            <Property path="warehouse"><Value><![CDATA[WH1]]></Value></Property>
        </Key>
        <Property path="description"><Value><![CDATA[Southeast Shipping Center]]></Value>
        </Property>
    </DomainEntity>
</Create>
```

In the above illustration, the template we are using is called Sample. We supply the name and number of the warehouse and let the template fill in the other values. The system reads the template values before the System-Link values. Thus, if the create action includes information for fields that are also in the template, the System-Link request values will prevail. This allows you to override template values if you wish.

Copy

Copying an object is like creating with a template, except that the maintainable field values come from a source object. To indicate this we use the <SourceObject> tag. Most <SourceObject> tags will not specify the object class, because it will be the same class as that defined in the <Create> tag.[4] All the <SourceObject> tag has to do is supply the key values of the object to be copied. In the following example, we are creating warehouse WH1 as a copy of WH2:

```
<Create name='newWarehouse' domainClass='com.mapics.mm.Warehouse' retainResult='false'>
    <SourceObject>
        <Key>
         <Property path='warehouse'><Value><![CDATA[WH2]]></Value></Property>
        </Key>
    </SourceObject>
    <DomainEntity>
        <Key>
            <Property path="warehouse"><Value><![CDATA[WH2]]></Value></Property>
        </Key>
        <Property path="description"><Value><![CDATA[Southeast Shipping Center]]></Value>
        </Property>
    </DomainEntity>
</Create>
```

Note: It is possible to use both <SourceObject> and <ApplyTemplate>. Copy templates are for this very purpose.

---

[4] There are a very few exceptions to this rule, e.g. CustomerOrders being copied to StandingOrders. In these instances, the <SourceObject> tag will specify the class for the source object.

Response

Regardless of the use of templates, source objects and overrides, in the end we get a new WH1 warehouse record. When the record is created the response includes the create action name and the new key. In the example below the create action name is *newWarehouse* and the key is *WH1*:

```xml
<?xml version="1.0" encoding="ISO-8859-1" ?>
<System-Link version="1.0" hostVersion="1.0">
<LoginResponse>
    <SessionHandle value="77158a:eca6484be4:-7ede" />
</LoginResponse>
<Response sessionHandle="77158a:eca6484be4:-7ede"        workHandle="77158a:eca6484be4:-7edc">
    <CreateResponse name="newWarehouse">
      <Key>
        <Property path="warehouse"><Value><![CDATA[WH1]]></Value></Property>
      </Key>
    </CreateResponse>
</Response>
</System-Link>
```

Example 5. Create multiple simple objects together (Warehouse with Address)

Often you need to create multiple objects at the same time. For example, we want to create a warehouse with an address:

Warehouse:

Number	WH1
Description	Southeast Shipping Center
Type	True (Controlled)
Site	RON
Shipping calendar	P1

Address:

Name	North Georgia
Address line 1	1013 Hillcrest Rd.
City	Duluth
State	GA

The XML to create the warehouse was shown in Example 4. The XML to create an address looks like this:

```
<Create name='newWhsAddr1' domainClass='com.mapics.pm.WarehouseAddress'>
   <DomainEntity>
     <Key>
        <Property path="warehouse"><Value><![CDATA[WH1]]></Value></Property>
        <Property path="addressId"><Value>100</Value></Property>
     </Key>
        <Property path="name"><Value><![CDATA[North Georgia]]></Value></Property>
        <Property path="address1"><Value><![CDATA[1300 Hillcrest Road]]></Value>
        </Property>
        <Property path="address4City"><Value><![CDATA[Duluth]]></Value></Property>
        <Property path="state"><Value><![CDATA[GA]]></Value></Property>
   </DomainEntity>
   </Create>
```

As you can see the request for the address looks just like the request for the warehouse. Both start and end with <Create> and <DomainEntity> tags, with properties (both key and regular) in the middle.
One single System-Link document can be used to create both objects. The warehouse would need to come first, of course, since one of the address creation edits verifies the existence of the warehouse.

The <ValueRef> tag

Note that the value of the warehouse ID in the address record is equal to the warehouse ID of the warehouse (i.e., *WH1*).

<Value><![CDATA[WH1]]></Value>

These two values must be the same, so that the new address will be associated with the new warehouse. Instead of typing in the value, *WH1*, we can use the <ValueRef> tag to tell the address object's property to use the same value as that in the warehouse object's property.

<ValueRef><![CDATA[(newWarehouse).warehouse]]></ValueRef>

In this example, <ValueRef> tags replace <Value> tags, and *(newWarehouse).warehouse* replaces *WH1*:

> *(newWarehouse)*   The name of the earlier create action, surrounded by parentheses
> *warehouse*   The property of that create action containing the value to be used

The <ValueRef> tag insures that the value of the warehouse ID is in the warehouse create action will automatically be used in the address create action. It can be used in this same way for any number of subsequent actions.

This technique is especially useful when the value to be reused is unknown by the XML programmer. For example, an order header may have a system-assigned order number that must be reused for the creation of line item objects. The <ValueRef> tag provides a way to tell the line item create actions to use the value from the order number property.

The following illustration shows the warehouse create action followed by the address create action. Note the <ValueRef> tag that gets the value WH1 from the *warehouse* property of the *newWarehouse* action. You can see now why it is important that all the action names within a work session be unique.

```
<Create name='newWarehouse' domainClass='com.mapics.mm.Warehouse' retainResult='true'>
    <ApplyTemplate clientClass='com.mapics.mm.Warehouse'><![CDATA[Sample]]></ApplyTemplate>
    <DomainEntity>
        <Key>
            <Property path="warehouse"><Value><![CDATA[WH1]]></Value></Property>
        </Key>
        <Property path="description"><Value><![CDATA[Southeast Shipping Center]]></Value>
        </Property>
    </DomainEntity>
</Create>
<Create name='newWhsAddr1' domainClass='com.mapics.pm.WarehouseAddress'>
<DomainEntity>
    <Key>
        <Property path="warehouse">
            <ValueRef><![CDATA[(newWarehouse).warehouse]]></ValueRef>
        </Property>
        <Property path="addressId"><Value>100</Value></Property>
    </Key>
        <Property path="name"><Value><![CDATA[North Georgia]]></Value></Property>
        <Property path="address1"><Value><![CDATA[1300 Hillcrest Rd.]]></Value></Property>
        <Property path="address4City"><Value><![CDATA[Duluth]]></Value></Property>
        <Property path="state"><Value><![CDATA[GA]]></Value></Property>
    </DomainEntity>
</Create>
```

Example 6. Create a complex object (Purchase Order)

A complex object contains multiple objects in a header/detail relationship. CO's, PO's, MO's, IFM transactions and engineering bills of material are all examples of complex objects.

As of this writing, a complex object is handled just like a collection of simple objects. The following example shows the creation of a purchase order with one line, and one release as follows:

Header

Number	System-assigned
Buyer	100
Warehouse	MPA
Vendor	123

Line item

PO number	Same as header
Line number	System-assigned
Warehouse	MPA
Item	12233
Quantity	100.0
Blanket	Yes

Release

PO number	Same as header
PO line	Same as line item
PO release	System-assigned
Due date	March 22, 2002
Quantity	5

Notes:

1. The create action for the header includes the <Key> tag with no properties. This tells System-Link that the missing properties are intentional. System-Link will pass the request on to the EJB Server, where the key (i.e., the PO number) will be assigned.

2. The line item has a two-part key: PO number and line number. We use a <ValueRef> tag to tell the system to insert the value of the PO number from the first create action. (The first create action must have *retainResult='true'* for this to work.) The second part of the key (the line number) will be assigned, just like the PO number was in the first create action.

3. The technique described above also applies to the release (three-part key).

4. The XML date is YYYYMMDD format.

5. Each create action stands on its own. Thus, if the header passes all the edits but the line item does not, you will create a PO header with no lines and no releases.

As usual, the login and request tags have been omitted for clarity.

```
<Create name='newPO' domainClass='com.mapics.pm.PurchaseOrder' retainResult='true'>
  <DomainEntity>
    <Key>
    </Key>
    <Property path='buyer'><Value><![CDATA[100]]></Value></Property>
    <Property path='warehouse'><Value><![CDATA[MPA]]></Value></Property>
    <Property path='vendor'><Value><![CDATA[123]]></Value></Property>
  </DomainEntity>
</Create>
<Create name='newPoItem' domainClass='com.mapics.pm.PoItem' retainResult='true'>
  <DomainEntity>
    <Key>
        <Property path='order'><ValueRef><![CDATA[(newPO).order]]></ValueRef></Property>
    </Key>
    <Property path='warehouse'><Value><![CDATA[MPA]]></Value></Property>
    <Property path='item'><Value><![CDATA[12233]]></Value></Property>
    <Property path='orderQuantityRequested'><Value>100.0</Value></Property>
    <Property path='dueToStockDate'><Value><![CDATA[20020314]]></Value></Property>
    <Property path='blanketItem'><Value>true</Value></Property>
  </DomainEntity>
</Create>
<Create name='newPoItemRel' domainClass='com.mapics.pm.PoItemRelease'>
  <DomainEntity>
    <Key>
      <Property path='order'><ValueRef><![CDATA[(newPO).order]]></ValueRef></Property>
      <Property path='line'><ValueRef><![CDATA[(newPoItem).line]]></ValueRef></Property>
    </Key>
    <Property path='dueToStockDate'><Value><![CDATA[20020322]]></Value></Property>
    <Property path='releaseQuantity'><Value>5</Value></Property>
  </DomainEntity>
</Create>
```

Result

The result shows each create action name and the key values for the record that was created. Login and session tags have been omitted for clarity:

```
<CreateResponse name="newPO">
    <Key>
        <Property path="order"><Value><![CDATA[P000047]]></Value></Property>
    </Key>
</CreateResponse>
<CreateResponse name="newPoItem">
    <Key>
        <Property path="order"><Value><![CDATA[P000047]]></Value></Property>
        <Property path="line"><Value>1</Value></Property>
    </Key>
</CreateResponse>
<CreateResponse name="newPoItemRel">
    <Key>
        <Property path="order"><Value><![CDATA[P000047]]></Value></Property>
        <Property path="line"><Value>1</Value></Property>
        <Property path="release"><Value>1</Value></Property>
    </Key>
</CreateResponse>
```

Example 7: 'Create' an action (PO receipt)

Sometimes the create operation is not for an object but for an action to an object. In this case, we want to receive items against a PO:

PO Number	P000798
PO Line	1
PO Release	0 (i.e., not a blanket item)
Quantity received to stock	2.0 EA
Location	AA01
FIFO date	3/12/02
Batch/lot	ABCDE

The System-Link action is a "create" of the receipt.

```
<Create name='newReceiveItem' domainClass='com.mapics.mm.ReceiveItem'>
   <ApplyTemplate clientClass='com.mapics.mm.ReceiveItem'>Stock receipts only</ApplyTemplate>
   <DomainEntity>
     <Key>
        <Property path='orderNumber'><Value><![CDATA[P000798]]></Value></Property>
      <Property path='line'><Value>1</Value></Property>
      <Property path='release'><Value>0</Value></Property>
     </Key>
     <Property path='quantityReceivedForStock'><Value>2.0</Value></Property>
     <Property path='stockLocation'><Value><![CDATA[AA01]]></Value></Property>
     <Property path='fifoDate'><Value><![CDATA[20020312]]></Value></Property>
     <Property path='batchLot'><Value><![CDATA[ABCDE]]></Value></Property>
     <Property path='receiptUm'><Value><![CDATA[EA]]></Value></Property>
   </DomainEntity>
   </Create>
```

The response includes the name of the create action and the key values:

Name	newReceiveItem
Key	PO P001381
	Line 1
	Rel 0

The <Login> and <Response> tags have been omitted for clarity:

```
<CreateResponse name="newReceiveItem">
   <Key>
      <Property path="orderNumber"><Value><![CDATA[P001381]]></Value></Property>
      <Property path="line"><Value>1</Value></Property>
      <Property path="release"><Value>0</Value></Property>
   </Key>
</CreateResponse>
```

Example 8: Update (PO line)

The <Update> tag is very similar to the <Create> tag:

- Both require a name for the action
- Both require the domainClass to be identified
- Both can have the same nested tags: <ApplyTemplate>, <DomainEntity>, etc.

In this example, we are making the following changes:

PO number	P000047	(key field)
Line	1	(key field)
Quantity requested	100	
Unit price requested	.51	There is no currency shown because it is not being changed

The following illustration shows the update action. Login and request tags have been omitted for clarity:

```
<Update name='updatePoItem' domainClass='com.mapics.pm.PoItem'>
   <DomainEntity>
     <Key>
         <Property path='order'><Value><![CDATA[P000047]]></Value></Property>
         <Property path='line'><Value><![CDATA[1]]></Value></Property>
     </Key>
     <Property path='orderQuantityRequested'><Value>100</Value></Property>
     <Property path='unitPriceRequested'><Value>.51</Value></Property>
   </DomainEntity>
</Update>
```

A successful update will result in a confirmation of the session.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<System-Link version="1.0" hostVersion="1.0">
   <LoginResponse>
       <SessionHandle value="1121f6:ecaa687cb1:-7476" />
   </LoginResponse>
<Response sessionHandle="1121f6:ecaa687cb1:-74b4" workHandle="1121f6:ecaa687cb1:-7474" />
</System-Link>
```

An unsuccessful update action will return an <UpdateResponse> tag identifying the name of the update action that failed (shown in bold):

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<System-Link version="1.0" hostVersion="1.0">
   <LoginResponse>
       <SessionHandle value="1121f6:ecaa687cb1:-743f" />
   </LoginResponse>
<Response sessionHandle="1121f6:ecaa687cb1:-743f" workHandle="1121f6:ecaa687cb1:-743d">
   <UpdateResponse name="updatePoItem">
     <Exception name="com.pjx.eScript.RequestDataException">
         <Text><![CDATA[ Error occurred during update]]></Text>
```

(The rest of the response has been omitted.)

Example 9: Delete (Warehouse)

The delete action is very straightforward – all you have to do is tell System-Link the name of the object and its key. In this example, the object is *Warehouse* and the key is *WH1*. Login and request tags have been omitted for clarity:

```
<Delete name='oldWarehouse' domainClass='com.mapics.mm.Warehouse'>
   <DomainEntity>
      <Key>
         <Property path='warehouse'>
            <Value><![CDATA[WH1]]></Value>
         </Property>
      </Key>
   </DomainEntity>
</Delete>
```

As with the update action, a successful delete action will have an empty response:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<System-Link version="1.0" hostVersion="1.0">
   <LoginResponse>
      <SessionHandle value="371786b:eca49ab7d6:-7dba" />
   </LoginResponse>
   <Response sessionHandle="371786b:eca49ab7d6:-7dba" workHandle="371786b:eca49ab7d6:-7db8" />
</System-Link>
```

As with the update action, an unsuccessful delete action will have a <DeleteResponse> tag showing the name of the delete action that failed:

<DeleteResponse name="oldWarehouse">

Chapter 10.   XML Syntax for requests and responses

The following is a description of the XA System-Link request and response tag hierarchy. See Examples for examples of System-Link requests and responses. The System-Link requests and response DTDs are contained in **Document Type Definitions (DTD)**

Note: business object attachments have special requirements, and thus are handled with a dedicated set of System-Link actions. For more information, see **Attachment Commands**.

Root Document

<u>System-Link</u>

System-Link tag

The System-Link tag is always the root entity for a System-Link request or response document. Aside from the XML prolog (containing the "<?xml" and "<!DOCTYPE" declarations), all entities in a System-Link document must be nested within a System-Link begin and end tag pair.

In addition to the reserved attributes, the System-Link tag has three additional attributes available. Two of these, "formatForDisplay" and "includeNewLines", are for use during development and debugging. These attributes control the generation of visual formatting whitespace (indentation and newlines) in the response output. (Please note: this whitespace is not required by any proper XML parser when parsing the System-Link response, and is not related to formatting the data for display – e.g. using an XSL stylesheet to generate an HTML document. Relying upon an XML element's absolute "position" in an XML document is very fragile, and is heavily discouraged.) Since formatting whitespace increases the size of the response document, this whitespace should be omitted when the document is used in production.

The third attribute, "useTimeZone", sets the timezone for all time values returned in the response. Currently, the two values supported are "default", which defaults to the timezone in which the System-Link server is running, and "GMT", which will convert all returned time values to GMT.

Attributes (only present in the request document - may appear in any order)

version	reserved for future use
hostVersion	reserved for future use
formatForDisplay	(Optional) if "true", formatting whitespace (indentation and newlines) will be included in the response document, otherwise indentation will be omitted. Default is "false".
includeNewlines	(Optional) if "true", formatting newlines will be included in output. Default is the same value as formatForDisplay. <u>includeNewlines is ignored if formatForDisplay is "true".</u>
useTimeZone	(Optional) sets the timezone for the time values returned in the response. Must be one of the following: "default" or "GMT". Defaults to "default".

Note:

- The default behavior for the generation of formatting whitespace has changed between Release 6 eZ-Link and Release 7 System-Link. In Release 6, the default behavior was for all formatting whitespace (indentation and newlines) to be included. In Release 7, the default behavior is for the formatting whitespace to be excluded.

Session Commands

<u>Login</u>

Login tag

Before a request can be processed, the user must login to the server. The login tag, when specified, must be the first action. A login action can be sent by itself or it may be immediately followed by a request action.

Attributes (may appear in any order)

userid	identifies the user that is making the request.
password	used to authenticate the userid
maxIdle	is the time, in milliseconds, that specifies how long the session will remain idle before being reclaimed. When this time expires, the server will automatically logout the session and reclaim its resources. A maxIdle value of zero will cause a logout immediately after the associated System-Link script is processed **Note:** The value specified for maxIdle must not exceed 10800000 (3 hours).
properties	a list of name/value pairs, each separated by a comma. Properties are used to select and configure a particular runtime implementation (e.g. currency, national language).

Notes:

- The userid and password attributes were previously named "principal" and "credentials". These older namings are still supported, however the use of the above attributes is recommended.
- The session maxIdle value can be further limited both globally and on a per-user basis. The former is accessed through Cross Application Support (under Application Settings in Power-Link), and the latter through User Profiles (under Integrator in Power-Link).

Example

This example script is for user "john doe". The password attribute contains his password "12345", and the maxIdle attribute sets his maximum idle time at 5 minutes. The properties attribute contains the XA environment, system name, and language selection.

```
<SystemLink version='1.0' hostVersion='1.0'>
  <Login userId='john doe' password='12345' maxIdle='300000'
     properties='com.mapics.cas.domain.EnvironmentId=25,
              com.mapics.cas.domain.SystemName=USATLD06.INFOR.COM,
              com.mapics.cas.user.LanguageId=en' />
</SystemLink>
```

LoginResponse tag

A login is either successful or it fails. The LoginResponse contains information that depends upon the outcome. A successful login returns the session handle, whereas a failed login returns an exception.

Attributes

actionSucceeded	set to "true" if the login action succeeded, otherwise set to "false"

Nested Tags

SessionHandle	if the login succeeds.
Exception	if the login fails.

Example 1

This example shows a possible response if the above login example were to succeed:

```
<SystemLink version='1.0' hostVersion='1.0'>
  <LoginResponse actionSucceeded='true'>
    <SessionHandle value='f3456a3fg:f34d'/>
  </LoginResponse>
</SystemLink>
```

Example 2

This example shows a possible response if the above login example were to fail:

```
<SystemLink version='1.0' hostVersion='1.0'>
  < LoginResponse actionSucceeded='false'>
   <Exception name='com.pjx.eScript.SecurityException'>
    <Text><![CDATA[Login Exception]]></Text>
    <Message type='error'>
     <Text><![CDATA[Unable to login user 'john doe': com.pjx.eScript.SecurityException:
com.ibm.as400.access.AS400SecurityException: Password is incorrect.]]></Text>
    </Message>
    <Message type='information'>
     <Text><![CDATA[Contact system administration]]></Text>
     <DetailedText><![CDATA[Contact your system administrator.]]></DetailedText>
    </Message>
    <Message type='error'>
     <Text><![CDATA[com.ibm.as400.access.AS400SecurityException: Password is incorrect.]]></Text>
     <DetailedText><![CDATA[]]></DetailedText>
    </Message>
   </Exception>
  </ LoginResponse >
</SystemLink>
```

Logout

Logout tag

The logout tag is used to force a session to logout, rather than having it automatically log out after remaining idle past its specified maximum idle time. This helps to reduce the use of server resources, as well as minimizes the security exposure caused by leaving the session handle valid longer than necessary.

Attributes

sessionHandle	identifies the target session for this action. This value must be a valid session handle.

Nested Tags

Exception	if the logout fails.

Example

This example is used to explicitly logout the user session known by 'f345'.

```
<SystemLink version='1.0' hostVersion='1.0'>
  <Logout sessionHandle='f345'/>
</SystemLink>
```

LogoutResponse tag

If the logout failed, the LogoutResponse will contain an exception describing the reason for failure. Otherwise, the LogoutResponse will have no sub elements, and actionSucceeded will simply be set to "true".

Attributes

actionSucceeded	set to "true" if the logout action succeeded, otherwise set to "false"

Example

This example shows a logout that failed because the session handle is invalid (possibly because the session has already timed out).

```
<SystemLink version='1.0' hostVersion='1.0'>
  <LogoutResponse actionSucceeded='false'>
   <Exception name='com.pjx.eScript.SecurityException'>
    <Text><![CDATA[sessionHandle not found]]></Text>
    <Message type='error'>
     <Text><![CDATA[Named session does not exist]]></Text>
     <DetailedText><![CDATA[There is no active session matching the given sessionHandle]]></DetailedText>
    </Message>
    <Message type='information'>
     <Text><![CDATA[Check for maxIdle expiration]]></Text>
     <DetailedText><![CDATA[Make sure that your maxIdle timeout is sufficiently long]]></DetailedText>
    </Message>
   </Exception>
  </LogoutResponse>
</SystemLink>
```

Request

Request tag

The request tag is used to scope an entire request action. A request tag must specify a session handle, work handle, and a maximum idle time for the work area.

Attributes (may appear in any order)

sessionHandle	identifies the login session and user for this request. A session handle value of "*current" must be specified if the request was preceded by a login action in the same document. The request will be executed only if the login is successful.
workHandle	identifies the work area for this request. Specifying a work handle tells the server to process the request in the work area identified by the work handle. A new work area can be requested by specifying a work handle value of "*new". The new work handle will be returned in the response message.
maxIdle	is the time, in milliseconds, that specifies how long the session will remain idle before being reclaimed. Each request will reset the work area's max idle time to the value specified. When this time expires, the server will reclaim the work area's resources. A maxIdle value of zero will cause the work area to be reclaimed immediately after the request action is processed. **Note:** The value specified for maxIdle must not exceed 10800000 (3 hours).

Nested Tags

Zero or more of the following tags:
- QueryObject, QueryList
- QueryAttachmentList, QueryAttachmentObject
- FetchNext, FetchNextAttachments, FetchNextMaintenanceHistory, FetchNextText

- Create, Update, Delete
- CreateAttachment, UpdateAttachment, DeleteAttachment
- IfCondition, EndIfCondition
- ListCustomDfns
- ListDomainClassInfo
- RefreshCustomizationData

Example 1

This example shows a login followed by a request. Notice that the work handle must be given a value of "*new" since the session did not exist before this script was processed. This request sets the workHandle's maximum idle value to 2 minutes.

```
<SystemLink version='1.0' hostVersion='1.0'>
  <Login userId='john doe' password='12345' maxIdle='300000'
      properties='com.mapics.cas.domain.EnvironmentId=25,
                com.mapics.cas.domain.SystemName=USATLD06.INFOR.COM,
                com.mapics.cas.user.LanguageId=en' />
  <Request sessionHandle='*current' workHandle='*new' maxIdle='120000'>
  </Request>
</SystemLink>
```

Example 2

This example shows a request intended for a work area created during a previous request. This request resets the workHandle's maximum idle time to 1 minute.

```
<SystemLink version='1.0' hostVersion='1.0'>
  <Request sessionHandle='f345' workHandle ='6c97d4' maxIdle ='60000'>
  </ Request >
</SystemLink>
```

Response

Response tag

A response tag is used to scope the entire response message. A response has attributes to link it back to its request.

Attributes (may appear in any order)

sessionHandle	identifies the login session and user for this response.
workHandle	identifies the login session and user for this response.

Nested Tags

Zero or more response tags	depending upon what request tags were present.
Exception	if the request action failed.

Example

This example shows the response for the Request Example 2 given above:

```
<SystemLink version='1.0' hostVersion='1.0'>
  <Response sessionHandle='f345' workHandle=' 6c97d4'>
  </Response>
</SystemLink>
```

Inquiry Commands

Note: business object attachments have special requirements, and thus are handled with a dedicated set of System-Link actions. For more information, see **Attachment Commands**.

QueryList

QueryList tag

A QueryList tag is used to query instances of a domain class. The value of the QueryList tag is an expression that selects a list of domain objects.

Attributes (may appear in any order)

name	an identifier for this query. **Must be unique within a given System-Link request document.** Queries are named in the context of a given work area. If a subsequent query is requested with the same name for the same work area, any outstanding results for the old query will be reclaimed.
domainClass	(Optional) the fully-qualified domain class over which the query applies. Either a domainClass or a classMnemonic must be provided.
classMnemonic	(Optional) the class identifier corresponding to the domain class over which the query applies. Either a domainClass or a classMnemonic must be provided.
includeMetaData	(Optional) If set to "true", meta information on the selected properties (e.g. from the PQL "SELECT" clause) will be provided. (At the present time, the meta information for QueryList is limited to the property "heading" value.) Defaults to "false".
maxReturned	the maximum number of instances returned in the associated QueryListResponse. For QueryList, the value of maxReturned must be greater than or equal to 0.
retainResult	(Optional) If set to "true", more results can later be obtained with the FetchNext action. Defaults to "true".

Nested Tags (in the order that they must be specified - either a Pql or a QueryDfn tag must be specified.)

Pql	contains a free-format query request. If a Pql tag is given, then no QueryDfn tag should be specified.
QueryDfn	contains references to custom definitions (header, subset, sort) used to create the list. If a QueryDfn tag is given, then no Pql tag should be specified.
Constraint	(Optional) contains the from object Key and relationship on which to constrain the QueryList action.
QueryText	(Optional) specifies the associated text object data to return as part of the QueryList response. One or more QueryText requests can be specified per QueryList request.
RunHostJob	(Optional) runs the specified host job against the list of objects specified by the Pql or QueryDfn tags.
CreateHostReport	(Optional) runs the specified host report against the list of objects specified by the Pql or QueryDfn tags.

Note:
- Not all domain classes support headers ("views"), subsets, or sorts. For these, the PQL tag must be used. Use the Power-Link customization panes, or see the ListCustomDfns tag to determine which definitions are present for a given domain class.
- If the sole purpose of a query is to run one or more host processes (i.e. the actual business object data is

not desired), its overhead (communication and resource) can be reduced by setting maxReturned to 0 and retainResult to "false".

- Unlike that with QueryObject, any QueryText subactions specified in a QueryList will never be available for subsequent FetchNextText actions (i.e. the retainResult for the QueryText will be ignored).

Example 1

This example queries a range of purchase orders and sorts it by buyer number. The purchase order domainClass "com.mapics.pm.PurchaseOrder" is specified. The response will only contain purchase orders selected by the PQL WHERE clause. MetaData for properties "order", "relatedBuyer.buyerName", and "relatedVendor.vendorName1" will be included in the response.

```
<SystemLink version='1.0' hostVersion='1.0'>
 <Login userId='john doe' password='12345' maxIdle='300'
     properties='com.mapics.cas.domain.EnvironmentId=25,
                 com.mapics.cas.domain.SystemName=USATLD06.INFOR.COM,
                 com.mapics.cas.user.LanguageId=en' />
 <Request sessionHandle='*current' workHandle='*new' maxIdle='30'>
   <QueryList name='queryPoList' domainClass='com.mapics.pm.PurchaseOrder'
       includeMetaData='true' maxReturned='20' retainResult='true'>
   <Pql>
    <![CDATA[SELECT order, relatedBuyer.buyerName, relatedVendor.vendorName1
             WHERE buyer > 'ABC' and buyer < 'XYZ'
             ORDER BY buyer]]>
       </Pql>
     </QueryList>
   </Request>
</SystemLink>
```

Example 2

This example queries a list of vendor items for a given Vendor, using the header, subset, and sort definitions created via Power-Link. The vendor item classMnemonic "VNDITM" is specified. Note that the Constraint keyword is used to limit this list to those items related to a specific Vendor. In this example, the sort is referenced by its object Id (obtainable via the ListCustomDfns action), whereas the header and subset are referenced by their names. The optional PromptedSubsetValues tag is also shown (with the CriteriaSpec showing the optional Operator tag). Since the optional retainResult attribute is not specified on the QueryList action, any items not displayed in the resulting QueryListResponse will not be available for a subsequent FetchNext action.

```
<SystemLink version='1.0' hostVersion='1.0'>
 <Login userId='john doe' password='12345' maxIdle='300'
     properties='com.mapics.cas.domain.EnvironmentId=25,
                 com.mapics.cas.domain.SystemName=USATLD06.INFOR.COM,
                 com.mapics.cas.user.LanguageId=en' />
  <Request sessionHandle='*current' workHandle='*new' maxIdle='30'>
   <QueryList name='query_VendorItem' classMnemonic='VNDITM'
               includeMetaData='true' maxReturned='10'>
    <QueryDfn>
     <HeaderDfnKey objectId='SVNDITM-HD-GENERAL'
               clientClass='com.mapics.pm.VendorItem'><![CDATA[General]]></HeaderDfnKey>

     <SubsetDfnKey objectId='Z#100CCBM0936321971777222123AAAZ6'
               clientClass='com.mapics.pm.VendorItem'><![CDATA[Commodity ...]]></SubsetDfnKey>
     <PromptedSubsetValues>
      <CriteriaSpec path='purchasedItemCommodityClass'>
        <Operator><![CDATA[Contains]]></Operator>
        <Value><![CDATA[207]]></Value>
      </CriteriaSpec>
     </PromptedSubsetValues>

     <SortDfnKey objectId='SVNDITM-SO-DEFAULT'
```

```
                    clientClass='com.mapics.pm.VendorItem'><![CDATA[(default)]]></SortDfnKey>
        </QueryDfn>

        <Constraint domainClass='com.mapics.pm.Vendor' relationshipName='relatedVendorItems'>
          <Key>
            <Property path='vendor'>
              <Value><![CDATA[PARA1]]></Value>
            </Property>
          </Key>
        </Constraint>
      </QueryList>
    </Request>
</SystemLink>
```

This example queries a list of Warehouses, using the header, subset, and sort definitions created via Power-Link. In addition, the host report "Order Shortages" is also run against the selected warehouses. Note that both a confirmation email message and an output email message (along with an attached report in PDF format) are sent. No attachments are made to any warehouse object (as indicated by the "attachTo" attribute), but the report is saved in HTML format.

```
<SystemLink version='1.0' hostVersion='1.0'>
  <Login userId='john doe' password='12345' maxIdle='300'
        properties='com.mapics.cas.domain.EnvironmentId=25,
                    com.mapics.cas.domain.SystemName=USATLD06.INFOR.COM,
                    com.mapics.cas.user.LanguageId=en' />
  <Request sessionHandle='*current' workHandle='*new' maxIdle='30'>
      <QueryList name='queryListWarehouse_General' domainClass='com.mapics.mm.Warehouse'
              includeMetaData='true' maxReturned='10'>
        <Pql>
          <![CDATA[SELECT warehouse,description,warehouseType,
                            site,planningWarehouse,sellingWarehouse,
                            primaryPlanningWarehouse
                    ORDER BY warehouse]]>
        </Pql>
        <CreateHostReport name='createHostReport_OrderShortages'>
          <HostReportType mnemonic='ORDSHTPRT'><![CDATA[Order Shortages]]></HostReportType>
          <ProcessContent>
            <Property path='description'>
              <Value><![CDATA[My Content Description]]></Value>
            </Property>
          </ProcessContent>
          <ProcessConfirmation>
            <EmailHeader>
              <AddressTo><![CDATA[Bob.Johnson@mycorp.com]]></AddressTo>
              <Subject><![CDATA[Confirmation: &description&]]></Subject>
            </EmailHeader>
            <EmailBody><![CDATA[The Order Shortages report &description& has been run.]]></EmailBody>
          </ProcessConfirmation>
          <ProcessOutput saveFormatText='false' saveFormatHtml='true' saveFormatPdf='false' attachTo='none'
                      emailAttachmentFormat='pdf'>
            <EmailHeader>
              <AddressTo><![CDATA[Frank.Smith@mycorp.com]]></AddressTo>
              <AddressCc><![CDATA[Bob.Johnson@mycorp.com]]></AddressCc>
              <Subject><![CDATA[Order Shortages report]]></Subject>
            </EmailHeader>
            <EmailBody><![CDATA[Frank, please review the attached report: &description&]]></EmailBody>
          </ProcessOutput>
        </CreateHostReport>
      </QueryList>
```

```
    </Request>
</SystemLink>
```

QueryListResponse tag

A QueryListResponse tag encompasses the result of a QueryList request. The query result is linked back to the requesting query through the name attribute.

Attributes (may appear in any order)

name	the same name submitted to the QueryList action tag.
requestedDomainClass	the domain class requested in the QueryList request.
actionSucceeded	set to "true" if the query action succeeded, otherwise set to "false"
moreResults	if more results exist than have been displayed, this has a value of "true". Note: these additional results can only be retrieved via FetchNext if "retainResult" was set to "true" on the QueryList action.

Nested Tags

MetaData	if the request specified that metadata was to be included.
ColumnStatistics	if the request contains column statistics (e.g. COUNT, MIN, MAX, AVG)
DomainEntity	contains a single Domain Object instance. Any associated QueryTextResponses will appear immediately after the DomainEntity.
QueryTextResponse	if one or more QueryText requests were present in the QueryList action. Will appear immediately after the associated DomainEntity.
RunHostJobResponse	if one or more RunHostJob requests were present in the QueryList action
CreateHostReportResponse	if one or more CreateHostReport requests were present in the QueryList action
Exception	if the QueryList has failed.

Example

This example is a possible response to the QueryList Example 1 previously mentioned. (For brevity, the LoginResponse has been omitted.)

```
<SystemLink version='1.0' hostVersion='1.0'>
  <Response sessionHandle='f345' workHandle='6c97d4'>
    <QueryListResponse name='queryPoList'
            requestedDomainClass='com.mapics.pm.PurchaseOrder'
            actionSucceeded='true' moreResults='false'>
      <MetaData>
        <Descriptor type='heading' path='order'>
          <![CDATA[Order]]>
        </Descriptor>
        <Descriptor type='heading' path='relatedBuyer.buyerName'>
          <![CDATA[Name - buyer]]>
        </Descriptor>
        <Descriptor type='heading' path='relatedVendor.vendorName1'>
          <![CDATA[Name - vendor]]>
        </Descriptor>
      </MetaData>
      <DomainEntity domainClass='com.mapics.pm.PurchaseOrder'>
        <Key>
          <Property path='order'>
            <Value><![CDATA[00000015]]></Value>
          </Property>
```

```
        </Key>
      <Property path='relatedBuyer.buyerName'>
        <Value><![CDATA[George Washington]]></Value>
      </Property>
      <Property path='relatedVendor.vendorName1'>
        <Value><![CDATA[ACME Supplies]]></Value>
      </Property>
    </DomainEntity>
    <DomainEntity domainClass='com.mapics.pm.PurchaseOrder'>
      <Key>
        <Property path='order'>
          <Value><![CDATA[00000018]]></Value>
        </Property>
      </Key>
      <Property path='relatedBuyer.buyerName'>
        <Value><![CDATA[John Adams]]></Value>
      </Property>
      <Property path='relatedVendor.vendorName1'>
        <Value><![CDATA[Continental Distributors]]></Value>
      </Property>
    </DomainEntity>
  </QueryListResponse>
 </Response>
</SystemLink>
```

QueryObject

QueryObject tag

The QueryObject tag is used to query a single object. The value of the QueryObject tag is an expression that selects exactly one object.

Attributes (may appear in any order)

name	an identifier for this query. **Must be unique within a given System-Link request document.** Actions are named in the context of a given work area. If a subsequent action is requested with the same name for the same work area, any outstanding results for the old action will be reclaimed.
domainClass	(Optional) the domain class over which the query applies. Either a domainClass or a classMnemonic must be provided.
classMnemonic	(Optional) the class identifier corresponding to the domain class over which the query applies. Either a domainClass or a classMnemonic must be provided.
includeMetaData	(Optional) if set to "true", meta information on the selected properties (e.g. from the PQL "SELECT" clause) will be provided. (At the present time, the meta information for QueryObject is limited to the property "label" value.) Default is "false".
showResult	(Optional) if set to "false", a successful object retrieval will result in no response being generated. This is useful in combination with retainResult set to "true" when only actions based upon the "named object" are of interest. Default is "true".
retainResult	(Optional) If set to "true", the resulting "named object" (with name set to the name of this QueryObject name identifier) will be retained for use in subsequent "value references". Default is "false".
existenceCheck	(Optional) If set to "true", an action failure due to ObjectNotFoundException will still show "actionSuceeded = 'true'" in the QueryObjectResponse. (This setting can be helpful when used in conjuction with conditional blocks.) Default is "false".

alwaysReturnKey	(Optional) If set to "true", the domain key properties will always be returned, even if the domain object is not found. Default is "false".

Nested Tags (in the order that they must be specified - either a Pql tag or a Key-CardFileDfnKey tag pair must be specified)

Pql	contains a free-format query request. This statement must return zero or one object. (If more than one object would be returned, an exception will be returned.) If a Pql tag is specified, then no Key and CardFileDfnKey tags should be specified.
Key	contains a list of key properties for the given domain class, as well as the corresponding values needed to uniquely reference the desired object. If a Key and a CardFileDfnKey are specified, then no Pql tag should be given.
CardFileDfnKey	(Optional) identifies the card file to be used to determine which properties to display for the selected object. If a Key is specified and no CardFileDfnKey is given, then the default card file is used from the user's preferences. If a Key is specified (with or without a CardFileDfnKey tag), then no Pql tag should be given.
QueryText	(Optional) specifies the associated text object data to return as part of the QueryObject response. One or more QueryText requests can be specified per QueryObject request.
QueryMaintenanceHistory	(Optional) specifies that associated maintenance history data should be returned as part of the QueryObject response.
RunHostJob	(Optional) runs the specified host job against the object specified by the Pql tag or Key-CardFileDfn tag pair.
CreateHostReport	(Optional) runs the specified host report against the object specified by the Pql tag or Key-CardFileDfn tag pair.
QueryRelationship	(Optional) specifies that a list of related objects should be returned as part of the QueryObject response. One or more QueryRelationship requests can be specified per QueryObject request.

Notes:
- Not all domain classes support card files. For these, the PQL tag must be used. Use the Power-Link customization panes, or see the ListCustomDfns tag to determine which definitions are present for a given domain class.
- The existenceCheck attribute can be helpful when the receiving program parses the response for instances of "actionSucceeded = 'false'". Using existenceCheck set to 'true', one can indicate that the ObjectNotFoundException was anticipated, and thus does not reflect an "unexpected failure" case. Internally, however, a QueryObject with an ObjectNotFoundException will always have its status marked as failure, for use by any subsequent IfCondition actions.
- For examples of QueryText, QueryMaintenanceHistory, RunHostJob, CreateHostReport, or QueryRelationship usage, please see their respective tag sections.

Example 1

This example uses PQL to query a single purchase order that is known by the key "00000015". The purchase order domainClass "com.mapics.pm.PurchaseOrder" is specified. MetaData will be returned in the response.

```
<SystemLink version='1.0' hostVersion='1.0'>
  <Login userId='john doe' password='12345' maxIdle='300'
      properties='com.mapics.cas.domain.EnvironmentId=25,
               com.mapics.cas.domain.SystemName=USATLD06.INFOR.COM,
               com.mapics.cas.user.LanguageId=en' />
  <Request sessionHandle='*current' workHandle='*new' maxIdle='30'>
    <QueryObject name='querySinglePo' domainClass='com.mapics.pm.PurchaseOrder'
```

```
                        includeMetaData='true' retainResult='false'>
        <Pql>
          <![CDATA[SELECT order, relatedBuyer.buyerName, relatedVendor.vendorName1
                WHERE order='00000015']]>
        </Pql>
      </QueryObject>
    </Request>
</SystemLink>
```

This example uses the Key and CardFileDfnKey to query the same single purchase order as in Example 1. The purchase order classMnemonic "POR" is specified. The properties returned are those defined in the specified card file.

```
<SystemLink version='1.0' hostVersion='1.0'>
  <Login userId='john doe' password='12345' maxIdle='300'
      properties='com.mapics.cas.domain.EnvironmentId=25,
                  com.mapics.cas.domain.SystemName=USATLD06.INFOR.COM,
                  com.mapics.cas.user.LanguageId=en' />
  <Request sessionHandle='*current' workHandle='*new' maxIdle='30'>
    <QueryObject name='querySinglePo' classMnemonic='POR'
            includeMetaData='true' retainResult='false'>
      <Key>
          <Property path='order'>
            <Value><![CDATA[00000015]]></Value>
          </Property>
      </Key>
      <CardFileDfnKey
clientClass='com.mapics.pm.PurchaseOrder'><![CDATA[MyPOCardFile]]></CardFileDfnKey>
      </QueryObject>
    </Request>
</SystemLink>
```

QueryObjectResponse tag

A QueryObjectResponse tag encompasses the result of a QueryObject request. The query result is linked back to the requesting query through the name attribute.

Attributes (may appear in any order)

name	the same name submitted to the QueryObject tag. If "retainResult" was set to "true", this name can be used to reference the retrieved object for later value references.
requestedDomainClass	the domain class requested by the QueryObject tag.
actionSucceeded	set to "true" if the query action succeeded, otherwise set to "false"

Nested Tags

MetaData	if the request specified that metadata was to be included.
DomainEntity	contains a Domain Object instance.
QueryTextResponse	if one or more QueryText requests were present in the QueryObject action
QueryMaintenanceHistoryResponse	If a QueryMaintenanceHistory request was present in the QueryObject action

RunHostJobResponse	if one or more RunHostJob requests were present in the QueryObject action
CreateHostReportResponse	if one or more CreateHostReport requests were present in the QueryObject action
QueryRelationshipResponse	if one or more QueryRelationship requests were present in the QueryObject action
Exception	if the QueryObject request failed.

Example 1

This example is a possible response to the first QueryObject example previously mentioned.

```
<SystemLink version='1.0' hostVersion='1.0'>
  <Response sessionHandle='f345' workHandle='6c97d4'>
   <QueryObjectResponse name='querySinglePo'
          requestedDomainClass='com.mapics.pm.PurchaseOrder'
          actionSucceeded='true'>
    <MetaData>
     <Descriptor type='label' path='order'>
      <![CDATA[Order]]>
     </Descriptor>
     <Descriptor type='label' path='relatedBuyer.buyerName'>
      <![CDATA[Buyer Name (related)]]>
     </Descriptor>
     <Descriptor type='label' path='relatedVendor.vendorName1'>
      <![CDATA[Vendor Name (related)]]>
     </Descriptor>
    </MetaData>
    <DomainEntity domainClass='com.mapics.pm.PurchaseOrder'>
     <Key>
      <Property path='order'>
       <Value><![CDATA[00000015]]></Value>
      </Property>
     </Key>
     <Property path='relatedBuyer.buyerName'>
      <Value><![CDATA[George Washington]]></Value>
     </Property>
     <Property path='relatedVendor.vendorName1'>
      <Value><![CDATA[ACME Supplies]]></Value>
     </Property>
    </DomainEntity>
   </QueryObjectResponse>
  </Response>
</SystemLink>
```

Example 2

This example shows the exception thrown when the PQL provided to QueryObject would return more than one domain object. (For brevity, the LoginResponse has been omitted.)

```
<SystemLink version='1.0' hostVersion='1.0'>
  <Response sessionHandle='f345' workHandle='6c97d4'>
    <QueryObjectResponse name='bogusQuery'
                 requestedDomainClass='com.mapics.pm.Buyer'
                 actionSucceeded='false'>
      <Exception name='com.pjx.eScript.WrongNumberObjectsException'>
        <Text><![CDATA[Multiple objects returned for QueryObject request]]></Text>
        <Message type='error'>
          <Text><![CDATA[Multiple objects returned for a QueryObject request: SELECT
buyer,buyerName,buyerTelephone,buyerUserid,EMailAddress WHERE buyer <> 'franklin' ORDER
BY buyer,buyerName]]></Text>
        </Message>
        <Message type='information'>
          <Text><![CDATA[Check for XML errors]]></Text>
          <DetailedText><![CDATA[Check your XML for errors.]]></DetailedText>
        </Message>
      </Exception>
    </QueryObjectResponse>
  </Response>
</SystemLink>
```

QueryText

QueryText tag

A QueryText action is used to request that text object data (e.g. comments, text object extensions created via Integrator, etc.) be returned as part of a QueryObject request.

Attributes (may appear in any order)

name	an identifier for this query. **Must be unique within a given System-Link request document.** Actions are named in the context of a given work area. If a subsequent action is requested with the same name for the same work area, any outstanding results for the old action will be reclaimed.
relationshipName	name of the relationship that will be traversed to produce the text data returned in the QueryText response.
includeMetaData	(Optional) If set to "true", meta information on the text object will be displayed. This will include the relationship description, as well as descriptions for each comment type (if the text object is "advanced" (typed)). Defaults to "false".
maxCharsReturned	maximum number of characters returned in the associated QueryTextResponse. If the text object is typed, each type requested will return up to this number of characters.
retainResult	(Optional) If set to "true", more results can later be obtained with the FetchNextText action. (Ignored unless the QueryText appears inside a QueryObject.) Defaults to "true".

Nested Tags (in the order that they must be specified)

TextFilter	used only with an advanced ("typed") text object to specify which comment types should be returned. For a simple (non-typed) text object, should not be used.

Notes:
- In Power-Link, an advanced text object is shown on a tabbed card, with one tab per comment type.
- For QueryText actions inside of QueryList and QueryRelationship actions, any value specified for retainResult will be ignored.

Example 1

This example shows a request fragment for an object that has an associated simple (non-typed) text object. Meta data will be returned in the response, which will show the first 2048 characters. Since retainResults is set to "false", any additional characters past the initial 2048 displayed will not be retained for later retrieval by FetchNextText.

```
<Request sessionHandle='f345' workHandle='*new' maxIdle='1000'>

  <QueryObject name='queryObject_RoutingOperation_Default'
              domainClass='com.mapics.epdm.RoutingOperation' includeMetaData='true'>
    <Key>
     <Property path='site'>
       <Value><![CDATA[AAB]]></Value>
     </Property>
      …
    </Key>
    <CardFileDfnKey clientClass='com.mapics.epdm.RoutingOperation'><![CDATA[Default]]></CardFileDfnKey>
    < QueryText name='queryText_AdditionalDescription' relationshipName='relatedOperationDesc'
              includeMetaData='true' maxCharsReturned='2048' retainResults='false'>
    </ QueryText >
  </QueryObject>

</Request>
```

Example 2

This example shows a request fragment for an object that has an associated advanced (typed) text object with three types AA, BB, and CC. Meta data will be returned in the response, which will show the first 100 characters. Additional characters beyond the 100 will be available for later retrieval by FetchNextText.

```
<Request sessionHandle='f345' workHandle='*new' maxIdle='1000'>

  <QueryObject name='queryObject_ItemRevision_TestCF'
              domainClass='com.mapics.epdm.ItemRevision' includeMetaData='true'>
    <Key>
     <Property path='site'>
       <Value><![CDATA[AAB]]></Value>
     </Property>
      …
    </Key>
    <CardFileDfnKey clientClass='com.mapics.epdm.ItemRevision'><![CDATA[IRYTestCF]]></CardFileDfnKey>
    < QueryText name='queryText_AdvancedComments' relationshipName='relatedItemRevisionAdvancedComments'
              includeMetaData='true' maxCharsReturned='100'>
      <TextFilter>
         <Type><![CDATA[AA]]></Type>
         <Type><![CDATA[BB]]></Type>
         <Type><![CDATA[CC]]></Type>
      </TextFilter>
    </ QueryText >
  </QueryObject>

</Request>
```

QueryTextResponse tag

A QueryTextResponse tag encompasses the result of a QueryText request. The query result is linked back to the requesting query through the name attribute.

Attributes (may appear in any order)

name	the same name submitted to the QueryText tag
actionSucceeded	set to "true" if the query action succeeded, otherwise set to "false"
moreChars	if more characters exist than have been displayed, this has a value of "true". Note: these additional characters can only be retrieved via FetchNextText if "retainResult" was set to "true" on the QueryText action.

Nested Tags (either a TypedTextValue or a Value will appear, but never both)

MetaData	if the request specified that metadata was to be included. Note that for typed values, metadata will only
TypedTextValue	for an advanced (typed) text object, contains the text type and value for each type requested
Value	for a simple (non-typed) text object, contains the text value
Exception	if the QueryText request failed.

Note:
- If the "moreChars" attribute is "true", there will be a corresponding "moreChars" attribute appearing on one or more Value tags in the response. Value tags without a "moreChars" attribute reflect the default value (specified in the response DTD) of "false".

Example 1

This example fragment is a possible response to the first QueryText example previously mentioned.

```
<Response sessionHandle='f345' workHandle='6c97d4'>
  <QueryObjectResponse name="queryObject_RoutingOperation_Default"
                       requestedDomainClass="com.mapics.epdm.RoutingOperation">
    <MetaData>
        …
    </MetaData>
    <DomainEntity domainClass="com.mapics.epdm.RoutingOperation">
        …
    </DomainEntity>
    < QueryTextResponse name="queryText_AdditionalDescription" actionSucceeded='true'
                        moreChars="false">
      <MetaData>
        <TextDescriptor type="label">
          <Value><![CDATA[Additional Description]]></Value>
        </TextDescriptor>
      </MetaData>
      <Value><![CDATA[
              First comment line
              Second comment line
              Third comment line
            ]]></Value>
    </ QueryTextResponse >
  </QueryObjectResponse>
```

```
    </Response>
```

This example fragment is a possible response to the second QueryText example previously mentioned. Note that
there are more characters available for type AA, as noted on both QueryTextResponse and the applicable Value
tag.

```
  <Response sessionHandle='f345' workHandle='6c97d4'>
    <QueryObjectResponse name="queryObject_ItemRevision_IRYTestCF" requestedDomainClass="com.mapics.epdm.ItemRe
     <MetaData>
       ...
     </MetaData>
     <DomainEntity domainClass="com.mapics.epdm.ItemRevision">
       ...
     </DomainEntity>
     < QueryTextResponse name="queryText_AdvancedComments"
                            actionSucceeded='true' moreChars="true">
      <MetaData>
       <TextDescriptor type="label">
         <Value><![CDATA[Advanced Comments]]></Value>
       </TextDescriptor>
       <TextDescriptor type="label">
         <Type><![CDATA[AA]]></Type>
         <Value><![CDATA[Engineering]]></Value>
       </TextDescriptor>
       <TextDescriptor type="label">
         <Type><![CDATA[BB]]></Type>
         <Value><![CDATA[Purchasing]]></Value>
       </TextDescriptor>
       <TextDescriptor type="label">
         <Type><![CDATA[CC]]></Type>
         <Value><![CDATA[Miscellaneous]]></Value>
       </TextDescriptor>
      </MetaData>

      <TypedTextValue>
        <Type><![CDATA[AA]]></Type>
        <Value moreChars='true'><![CDATA[
                 First Engineering comment line
                 Second En]></Value>
      </TypedTextValue>
      <TypedTextValue>
        <Type><![CDATA[BB]]></Type>
        <Value></Value>
      </TypedTextValue>
      <TypedTextValue>
        <Type><![CDATA[CC]]></Type>
        <Value><![CDATA[
                 Only Miscellaneous comment line
                 ]]></Value>
      </TypedTextValue>
     </ QueryTextResponse >
    </QueryObjectResponse>
  </Response>
```

QueryMaintenanceHistory tag

A QueryMaintenanceHistory indicates that a list of maintenance history objects (corresponding to individual maintenance actions) should be returned along with the QueryObject response. Each maintenance history object will contain both the details of the maintenance action itself (e.g. maintenance type, date, reason code, etc..)

Attributes (may appear in any order)

name	an identifier for this query. **Must be unique within a given System-Link request document.** Actions are named in the context of a given work area. If a subsequent action is requested with the same name for the same work area, any outstanding results for the old action will be reclaimed.
includeMetaData	(Optional) if set to "true", meta information on the selected properties (e.g. from the PQL "SELECT" clause) will be provided. (At the present time, the meta information for QueryMaintenaceHistory is limited to the property "heading" value.) Default is "false".
maxReturned	the maximum number of instances returned in the associated QueryMaintenanceHistoryResponse.
retainResult	(Optional) If set to "true", more results can later be obtained with the FetchNextMaintenanceHistory action. Defaults to "true".

Nested Tags (in the order that they must be specified - either a Pql or a QueryDfn tag must be specified)

Pql	contains a free-format query request. If a Pql tag is given, then no QueryDfn tag should be specified.
QueryDfn	contains references to custom definitions (header, subset, sort) used to create the list. If a QueryDfn tag is given, then no Pql tag should be specified.

Notes:
- The where clause of the PQL statement is optional and is only used to add additional criteria to the relationship.
- The display of MetaData for a QueryMaintenanceHistory request is independent of the display of MetaData for its QueryObject request.
- A single QueryObject request can have only one QueryMaintenanceHistory request.

Example

The following example fragment shows a query request for a single Purchase Order, with the first ten maintenance history records included. Due to the values specified for the "includeMetaData" attributes, MetaData will only be included for the maintenance history list, not for the Purchase Order.

```
<QueryObject name='querySinglePo' domainClass='com.mapics.pm.PurchaseOrder'
            includeMetaData='false'>
 <Pql>
  <![CDATA[SELECT order, relatedBuyer.buyerName, relatedVendor.vendorName1
    WHERE order='00000013']]>
 </Pql>
 <QueryMaintenanceHistory name='queryMaintHist_PurchaseOrder_Default'
                            includeMetaData='true' maxReturned='10'>
   <Pql>
     <![CDATA[SELECT transactionDate,transactionTime,
                relatedTransactionMaster.transactionDescriptionDerived,
                objectType,objectId,user,relatedMaintenanceReason.description
             REQUIRED relatedTransactionMaster
```

```
                ORDER BY transactionDate DESC, transactionTime DESC]]>
        </Pql>
    </QueryMaintenanceHistory>
</QueryObject>
```

QueryMaintenanceHistoryResponse tag

A QueryMaintenanceHistoryResponse tag encompasses the result of a QueryMaintenanceHistory request. The query result is linked back to the requesting query through the name attribute.

Attributes (may appear in any order)

name	the same name submitted to the QueryMaintenanceHistory tag.
actionSucceeded	set to "true" if the query action succeeded, otherwise set to "false"
moreResults	if more results exist than have been displayed, this has a value of "true".

Nested Tags

MetaData	if the request specified that metadata was to be included.
HistoryEntry	contains a single maintenance history action instance.
Exception	if the QueryMaintenanceHistory failed.

Example

This example fragment is a possible response to the QueryMaintenanceHistory example previously mentioned.

```
<QueryObjectResponse name='querySinglePo' requestedDomainClass='com.mapics.pm.PurchaseOrder'>
  <DomainEntity domainClass='com.mapics.pm.PurchaseOrder'>
    <Key>
      <Property path='order'>
        <Value><![CDATA[P001013]]></Value>
      </Property>
    </Key>
    <Property path='relatedBuyer.buyerName'>
      <Value><![CDATA[Jack Scott]]></Value>
    </Property>
    <Property path='relatedVendor.vendorName1'>
      <Value><![CDATA[Jadot Brothers]]></Value>
    </Property>
  </DomainEntity>
  <QueryMaintenanceHistoryResponse name='queryMaintHist_PurchaseOrder_Default'
                                   actionSucceeded='true' moreResults='false'>
    <MetaData>
      <Descriptor type='heading' path='transactionDate'><![CDATA[Date]]></Descriptor>
      <Descriptor type='heading' path='transactionTime'><![CDATA[Time]]></Descriptor>
      <Descriptor type='heading'
                  path='relatedTransactionMaster.transactionDescriptionDerived'><![CDATA[Transaction
description (derived)]]></Descriptor>
      <Descriptor type='heading' path='objectType'><![CDATA[Object type]]></Descriptor>
      <Descriptor type='heading' path='objectId'><![CDATA[Object ID]]></Descriptor>
      <Descriptor type='heading' path='user'><![CDATA[User]]></Descriptor>
      <Descriptor type='heading'
path='relatedMaintenanceReason.description'><![CDATA[Description]]></Descriptor>
    </MetaData>
    <HistoryEntry>
```

```xml
<Property path='transactionDate'>
  <Value>20040108</Value>
</Property>
<Property path='transactionTime'>
  <Value>122538</Value>
</Property>
<Property path='relatedTransactionMaster.transactionDescriptionDerived'>
  <Value><![CDATA[Create P.O. Item]]></Value>
</Property>
<Property path='objectType'>
  <Value><![CDATA[Purchase Order Item]]></Value>
</Property>
<Property path='objectId'>
  <Value><![CDATA[P001013,1]]></Value>
</Property>
<Property path='user'>
  <Value><![CDATA[PATESHI]]></Value>
</Property>
<Property path='relatedMaintenanceReason.description'>
  <Value><![CDATA[Reason B]]></Value>
</Property>
<ChangedDomainValues domainClass='com.mapics.pm.PoItem'>
  <ChangedProperty path='order'>
    <Description><![CDATA[Order]]></Description>
    <ValueAfter><![CDATA[P001013]]></ValueAfter>
  </ChangedProperty>
  <ChangedProperty path='vendor'>
    <Description><![CDATA[Vendor]]></Description>
    <ValueAfter><![CDATA[101]]></ValueAfter>
  </ChangedProperty>
    ...
  <ChangedProperty path='extendedDescriptionFlag'>
    <Description><![CDATA[Extended description flag]]></Description>
    <ValueAfter><![CDATA[0]]></ValueAfter>
  </ChangedProperty>
</ChangedDomainValues>
</HistoryEntry>
<HistoryEntry>
  <Property path='transactionDate'>
    <Value>20040108</Value>
  </Property>
  <Property path='transactionTime'>
    <Value>103137</Value>
  </Property>
  <Property path='relatedTransactionMaster.transactionDescriptionDerived'>
    <Value><![CDATA[Change Purchase Order]]></Value>
  </Property>
  <Property path='objectType'>
    <Value><![CDATA[Purchase Order]]></Value>
  </Property>
  <Property path='objectId'>
    <Value><![CDATA[P001013]]></Value>
  </Property>
  <Property path='user'>
    <Value><![CDATA[PATESHI]]></Value>
  </Property>
  <Property path='relatedMaintenanceReason.description'>
    <Value><![CDATA[Reason A]]></Value>
  </Property>
  <ChangedDomainValues domainClass='com.mapics.pm.PurchaseOrder'>
```

```
          <ChangedProperty path='estimatedSpecialCharges'>
            <Description><![CDATA[Estimated special charges]]></Description>
            <ValueBefore><![CDATA[0.00]]></ValueBefore>
            <ValueAfter><![CDATA[11.00]]></ValueAfter>
          </ChangedProperty>
          <ChangedProperty path='revised'>
            <Description><![CDATA[Revised]]></Description>
            <ValueBefore></ValueBefore>
            <ValueAfter><![CDATA[R]]></ValueAfter>
          </ChangedProperty>
        </ChangedDomainValues>
      </HistoryEntry>
    </QueryMaintenanceHistoryResponse>
 </QueryObjectResponse>
```

HistoryEntry

HistoryEntry tag

The HistoryEntry tag is used to contain a single maintenance history action instance. A HistoryEntry will first list the requested properties for the maintenance history instance itself, followed by a list of changed properties for the associated domain object.

Attributes
(none)

Nested Tags

Property	each property tag holds a value associated with the maintenance history instance
ChangedDomainValues	contains changed properties for the associated domain object
Exception	if the QueryMaintenanceHistory failed.

See the QueryMaintenanceHistory tag section for examples

ChangedDomainValues

ChangedDomainValues tag

The ChangedDomainValues tag contains the list of changed properties for the associated domain object instance.

Attributes

domainClass	the class of the domain object instance

Nested Tags

ChangedProperty	contains the before and after values (as applicable) for a single changed property on the domain object instance
Exception	if the QueryMaintenanceHistory failed.

See the QueryMaintenanceHistory tag section for examples

ChangedProperty

ChangedProperty tag

The ChangedProperty tag contains the change information for a single changed property on the domain object instance.

The presence of before and after values will depend upon the type of maintenance action. For example, create actions will only result in an after value, whereas change actions will show both.

Attributes

path	the property path for the property

Nested Tags

Description	description of the property
ValueBefore	the value of the property before the maintenance action was applied. Will not appear for a create action.
ValueAfter	the value of the property after the maintenance action was applied. Will not appear for the delete action.

See the QueryMaintenanceHistory tag section for examples

ValueBefore and ValueAfter

ValueBefore and ValueAfter tags

The ValueBefore and ValueAfter tags contain the before and after values, respectively, for a single changed property on the domain object instance.

In some circumstances, a property's type may change (e.g. changing a domain class with Integrator). This will likely cause any previous maintenance history entries containing values for this property to no longer match (e.g. changing an integer type to a date type). If this occurs, the isValid attribute will be set to "false", and the value will be given as it appears in the database.

Attributes

isValid	if present and set to "false", the corresponding value in the database could not be converted to the current property type. Otherwise, the value given is valid for the current property type.
isNull	if present and set to "true", the empty value given represents a null property value. Otherwise, the value given is non-null. isNull is used to distinguish a null property value from an empty String property value.

Nested Tags
a String value

See the QueryMaintenanceHistory tag section for examples

RunHostJob tag

A RunHostJob action is used to run a given host job against the selected dataset specified by the outer query action. (For QueryObject, the host job will be run against the single business object. For QueryList, the host job will be run against all business objects that match the list selection criteria.)

Attributes (may appear in any order)

name	an identifier for this host process creation. **Must be unique within a given System-Link request document.** Actions are named in the context of a given work area. If a subsequent action is requested with the same name for the same work area, any outstanding results for the old action will be reclaimed.

Nested Tags (in the order that they must be specified)

HostJobType	contains a reference to the host job to be run
ProcessContent	contains one or more properties specific to the specified host job
ProcessConfirmation	(Optional) contains the properties used to produce a confirmation email message

Example

The following example fragment shows host job "Purge" being run against a list of all purchase orders that have "TONYSA" as the vendor. Please note the following:

- the selection criteria for the host job are specified in the outer QueryList, not within the RunHostJob action.
- maxReturned for the outer QueryList has been set to 0, retainResult set to 'false', and includeMetaData is omitted (defaults to 'false'). This is recommended if the QueryList is only used to set the selection criteria for the nested host process create actions.
- There is a substitution variable in the confirmation email Subject. The confirmation email sent will have the value of the ProcessContent description field ("Content Description") in this place.

```
<QueryList name='queryListPurchaseOrder_General' domainClass='com.mapics.pm.PurchaseOrder'
        maxReturned='0' retainResult='false'>
  <Pql>
    <![CDATA[SELECT order WHERE vendor = 'TONYSA' ORDER BY order]]>
  </Pql>
  <RunHostJob name='runHostJob_Purge'>
    <HostJobType mnemonic='PORPRG'><![CDATA[Purge]]></HostJobType>
    <ProcessContent>
       <Property path='copyMaintenanceHistoryToHistory'>
          <Value><![CDATA[1]]></Value>
       </Property>
       <Property path='description'>
          <Value><![CDATA[Content Description]]></Value>
       </Property>
       <Property path='purgeMiscellaneousInvoices'>
          <Value><![CDATA[1]]></Value>
       </Property>
       <Property path='copyAttachmentsToHistory'>
          <Value><![CDATA[0]]></Value>
       </Property>
    </ProcessContent>
    <ProcessConfirmation>
       <EmailHeader>
          <AddressTo><![CDATA[Fred.Smith@mycorp.com]]></AddressTo>
          <AddressCc><![CDATA[Jake.Jones@mycorp.com]]></AddressCc>
          <Subject><![CDATA[Purge &description& has completed]]></Subject>
```

```
            </EmailHeader>
            <EmailBody><![CDATA[The PO Purge job has completed.]]></EmailBody>
        </ProcessConfirmation>
      </RunHostJob>
    </QueryList>
```

RunHostJobResponse tag

A RunHostJobResponse tag reports the status of a create host job action. Note that this status reflects whether or not the job was successfully submitted: although some errors (e.g. host job type not found) will be caught at the time of submission, errors and warnings returned by the host process itself will only be reflected in the corresponding transaction status record(s).

Attributes (may appear in any order)

name	the same name submitted to the RunHostJob tag.
actionSucceeded	set to "true" if the host job submission succeeded, otherwise set to "false"

Nested Tags

Exception	if the RunHostJob submission failed.

Example

This example is a possible response fragment to the RunHostJob example previously mentioned.

```
<QueryListResponse name='queryListPurchaseOrder_General'
                    requestedDomainClass='com.mapics.pm.PurchaseOrder' moreResults='true'>
 <RunHostJobResponse name='runHostJob_Purge' actionSucceeded='true'/>
</QueryListResponse>
```

CreateHostReport

CreateHostReport tag

A CreateHostReport action is used to run a given host report against the selected dataset specified by the outer query action. (For QueryObject, the host report will be run against the single business object. For QueryList, the host report will be run against all business objects that match the list selection criteria.)

Attributes (may appear in any order)

name	an identifier for this host process creation. **Must be unique within a given System-Link request document.** Actions are named in the context of a given work area. If a subsequent action is requested with the same name for the same work area, any outstanding results for the old action will be reclaimed.

Nested Tags (in the order that they must be specified)

HostReportType	contains a reference to the host report to be run
ProcessContent	contains one or more properties specific to the specified host report
ProcessConfirmation	(Optional) contains the properties used to produce a confirmation email message
ProcessOutput	(Optional) contains the properties used to produce an output email message, as well as an Attachment object

The following example fragment shows host report "Item Shortage" being run against a list of all manufacturing orders. Please note the following:

- the selection criteria for the host report are specified in the outer QueryList, not within the CreateHostReport action.
- maxReturned for the outer QueryList has been set to 0, retainResult set to 'false', and includeMetaData is omitted (defaults to 'false'). This is recommended if the QueryList is only used to set the selection criteria for the nested host process create actions.
- There is a substitution variable in the confirmation and output email Subject. The email messages sent will have the value of the ProcessContent description field ("Content Description") in this place.
- There is a substitution variable in the output email body. The output email message will have the value of the AddressTo tag in the ProcessCofirmation ("Frank.Smith@mycorp.com") in this place.
- One global attachment common to all MO will be created (text format), and the output email message will have a matching attachment (PDF format).

```
<QueryList name='queryListManufacturingOrder_General' domainClass='com.mapics.obpm.ManufacturingOrder'
            maxReturned='0' retainResult='false'>
  <Pql>
      <![CDATA[SELECT order ORDER BY order DESC]]>
  </Pql>
  <CreateHostReport name='createHostReport_ItemShortage'>
      <HostReportType mnemonic='ITMSHTPRT'><![CDATA[Item Shortage]]></HostReportType>
      <ProcessContent>
          <Property path='description'>
              <Value><![CDATA[Content Description]]></Value>
          </Property>
      </ProcessContent>
      <ProcessConfirmation>
          <EmailHeader>
              <AddressTo><![CDATA[Frank.Smith@mycorp.com]]></AddressTo>
              <AddressBcc><![CDATA[Jim.Johnson@mycorp.com]]></AddressBcc>
              <Subject><![CDATA[Item Shortage &description& complete]]></Subject>
          </EmailHeader>
          <EmailBody><![CDATA[Item shortage report has been completed.]]></EmailBody>
      </ProcessConfirmation>
      <ProcessOutput saveFormatText='true' saveFormatHtml='false' saveFormatPdf='false'
                    attachTo='global' emailAttachmentFormat='pdf'>
          <EmailHeader>
              <AddressTo><![CDATA[Fred.Murphy@mycorp.com]]></AddressTo>
              <Subject><![CDATA[Item Shortage report: &description&]]></Subject>
          </EmailHeader>
          <EmailBody><![CDATA[The Item Shortage report was generated by &confirmationMessageSendTo&.]]></Emai
      </ProcessOutput>
  </CreateHostReport>
</QueryList>
```

CreateHostReportResponse tag

A CreateHostReportResponse tag reports the status of a create host report action. Note that this status reflects whether or not the report was successfully submitted: although some errors (e.g. host report type not found) will be caught at the time of submission, errors and warnings returned by the host process itself will only be reflected in the corresponding transaction status record(s).

Attributes (may appear in any order)

name	the same name submitted to the CreateHostReport tag.

actionSucceeded	set to "true" if the host report submission succeeded, otherwise set to "false"

Nested Tag

Exception	if the RunHostJob submission failed.

Example
This example is a possible response fragment to the CreateHostReport example previously mentioned.

```
<QueryListResponse name='queryListManufacturingOrder_General'
                   requestedDomainClass='com.mapics.obpm.ManufacturingOrder' moreResults='true'>
  <CreateHostReportResponse name='createHostReport_ItemShortage' actionSucceeded='true'/>
</QueryListResponse>
```

ProcessContent

ProcessContent tag

The ProcessContent tag is used to specify values for host-process-specific properties. These properties will consist of a description, followed by zero or more additional options as dictated by the individual host process.

Attributes: none

Nested Tag

Property	contains a value for a single content option

See the RunHostJob and CreateHostReport sections for examples.

ProcessConfirmation

ProcessConfirmation tag

The ProcessConfirmation tag is used to specify that an acknowledgement email message should be sent upon successful completion of the host process. (ProcessConfirmation is different from the ProcessOutput tag, which can used to actually send the generated output of a host report as an attachment on an email message.)

Attributes: none

Nested Tags (in the order that they must be specified)

EmailHeader	contains the header values (e.g. addresses, subject) for the confirmation email message
EmailBody	(Optional) contains the actual body text for the confirmation email message

See the RunHostJob and CreateHostReport sections for examples.

ProcessOutput

ProcessOutput tag

The ProcessOutput tag is used to specify that the output of the corresponding host report should be sent via an email message and / or attached (via an XA Attachment object) to the associated business objects. (ProcessOutput is different from the ProcessConfirmation tag, which can used to send an acknowledgement email upon successfully completion of a host process.)

Attributes (may appear in any order)

saveFormatText	(Optional) If set to "true", the report output will be stored in text format. Default is "false".
saveFormatHtml	(Optional) If set to "true", the report output will be stored in HTML format. Default is "false".
saveFormatPdf	(Optional) If set to "true", the report output will be stored in PDF format. Default is "false".
attachTo	(Optional) If set to "global", a global XA Attachment object will be created against all business object instances of the associated class (e.g. Warehouse). If set to "none", no XA Attachment will be created. Default is "none".
emailAttachmentFormat	(Optional) creates an output email message with an email attachment in the named format: "text", "html", or "pdf" (for text, HTML, or PDF format respectively). **If set to "none", no output email message will be generated.** Default is "none".

Nested Tag (in the order that they must be specified)

EmailHeader	contains the header values (e.g. addresses, subject) for the confirmation email message
EmailBody	(Optional) contains the actual body text for the confirmation email message

Note:
- Both the EmailHeader and EmailBody tag should be omitted if no output email message is to be generated (i.e. if the emailAttachmentFormat attribute is set to "none").

See the RunHostJob and CreateHostReport sections for examples.

AddressTo, AddressCc, AddressBcc

AddressTo, AddressCc, AddressBcc tag

These tags are used to specify the To, CC, and BCC addresses for host process confirmation and output email messages. They are grouped here due to common functionality and format.

Attributes: none

Nested Tags

A String representation of one email address.

See the RunHostJob and CreateHostReport sections for examples.

Subject

Subject tag

The Subject tag is used to specify the subject for host process confirmation and output email messages.

Attributes: none

Nested Tags

A String giving the subject line for an email message.

See the RunHostJob and CreateHostReport sections for examples.

EmailBody

EmailBody tag

The EmailBody tag is used to specify the body text for host process confirmation and output email messages.

Attributes: none

Nested Tags

A String giving the body text for an email message.

See the RunHostJob and CreateHostReport sections for examples.

QueryRelationship

QueryRelationship tag

A QueryRelationship action is similar to a QueryList action, except that its PQL is partially specified. The result of the enclosing QueryObject is used to provide the missing data.

Attributes (may appear in any order)

name	an identifier for this query. **Must be unique within a given System-Link request document.** Actions are named in the context of a given work area. If a subsequent action is requested with the same name for the same work area, any outstanding results for the old action will be reclaimed.
relationshipName	name of the relationship that will be traversed to produce the QueryRelationship result set.
includeMetaData	(Optional) if set to "true", meta information on the selected properties (e.g. from the PQL "SELECT" clause) will be provided. (At the present time, the meta information for QueryResponse is limited to the property "heading" value.) Default is "false".
maxReturned	the maximum number of instances returned in the associated QueryRelationshipResponse. For QueryRelationship, the value of maxReturned must be greater than or equal to 0.

Nested Tags (in the order that they must be specified - either a Pql or a QueryDfn tag must be specified)

Pql	contains a free-format query request. For a QueryRelationship: the PQL statement will sometimes contain "substitution variables" of the form "&*varName&*", where *varname* is the name of a property in the object returned by the QueryObject or outer QueryRelationship request. If a Pql tag is given, then no QueryDfn tag should be specified.
QueryDfn	contains references to custom definitions (header, subset, sort) used to create the list. If a QueryDfn tag is given, then no Pql tag should be specified.
QueryText	(Optional) specifies the associated text object data to return as part of the QueryRelationship response. One or more QueryText requests can be specified per QueryRelationship request.
QueryRelationship	zero or more QueryRelationship actions may be nested inside a given QueryRelationship action. If present, the nested QueryRelationship actions will be applied to each object returned by the outer QueryRelationship action.

Notes:

- The where clause of the relationship PQL statement is optional and is only used to add additional criteria to the relationship.
- The display of MetaData for a QueryRelationship request is independent of the display of MetaData for its QueryObject request.
- A single QueryObject request can have one or more QueryRelationship requests.
- Even though QueryRelationship has a "maxReturned" attribute like QueryList, FetchNext actions cannot be performed against outstanding QueryRelationship result sets. The "maxReturned" attribute is present solely to limit the number of results returned. Due to the potential result set size of nested QueryRelationship actions, as well as the increased time that processing them requires, one should make especially sure to provide "maxReturned" values.
- not all domain classes support headers, subsets, or sorts. For these, the PQL tag must be used. Use the Power-Link customization panes, or see the ListCustomDfns tag to determine which definitions are present for a given domain class.
- Unlike that with QueryObject, any QueryText subactions specified in a QueryList will never be available for subsequent FetchNextText actions (i.e. the retainResult for the QueryText will be ignored).

Example

The following example shows a query request for a single Purchase Order, followed by a query request for all of its related Items. Due to the values specified for the "includeMetaData" attributes, MetaData will only be included for the Item list, not for the Purchase Order.

```
<SystemLink version='1.0' hostVersion='1.0'>
  <Login userId='john doe' password='12345' maxIdle='300'
       properties='com.mapics.cas.domain.EnvironmentId=25,
                   com.mapics.cas.domain.SystemName=USATLD06.INFOR.COM,
                   com.mapics.cas.user.LanguageId=en'/>
  <Request sessionHandle='*current' workHandle='*new' maxIdle='30'>
    <QueryObject name='querySinglePo' domainClass='com.mapics.pm.PurchaseOrder'
          includeMetaData='false'>
      <Pql>
      <![CDATA[SELECT order, relatedBuyer.buyerName, relatedVendor.vendorName1
        WHERE order='00000015']]>
      </Pql>
      <QueryRelationship name='queryPoItems' relationshipName='relatedPoItems'
                     includeMetaData='true' maxReturned='10'>
        <Pql>
        <![CDATA[SELECT order, line, buyer, currency ORDER BY line, buyer]]>
        </Pql>
      </QueryRelationship>
    </QueryObject>
```

```
    </Request>
</SystemLink>
```

QueryRelationshipResponse tag

A QueryRelationshipResponse tag encompasses the result of a QueryRelationship request. The query result is linked back to the requesting query through the name attribute.

Attributes (may appear in any order)

name	the same name submitted to the QueryRelationship tag.
actionSucceeded	set to "true" if the query action succeeded, otherwise set to "false"
moreResults	if more results exist than have been displayed, this has a value of "true".

Nested Tags

MetaData	if the request specified that metadata was to be included.
ColumnStatistics	if the request contains column statistics (e.g. COUNT, MIN, MAX, AVG)
DomainEntity	contains a Domain Object instance.
Exception	if the QueryRelationship failed.

Note:
- Even though QueryRelationshipResponse has a "moreResults" attribute like QueryList, FetchNext actions cannot be performed against outstanding QueryRelationship result sets.

Example

This example is a possible response to the QueryRelationship example previously mentioned. (For brevity, the LoginResponse has been omitted.)

```
<SystemLink version='1.0' hostVersion='1.0'>
  <Response sessionHandle='f345' workHandle='6c97d4'>
   <QueryObjectResponse name='querySinglePo' requestedDomainClass='com.mapics.pm.PurchaseOrder'>
    <DomainEntity domainClass='com.mapics.pm.PurchaseOrder'>
     <Key>
      <Property path='order'>
       <Value><![CDATA[00000015]]></Value>
      </Property>
     </Key>
     <Property path='relatedBuyer.buyerName'>
      <Value><![CDATA[George Washington]]></Value>
     </Property>
     <Property path='relatedVendor.vendorName1'>
      <Value><![CDATA[ACME Supplies]]></Value>
     </Property>
    </DomainEntity>
    <QueryRelationshipResponse name='queryPoItems' actionSucceeded='true' moreResults='false'>
     <MetaData>
       <Descriptor type='heading' path='order'>
        <![CDATA[Order]]>
       </Descriptor>
       <Descriptor type='heading' path='line'>
        <![CDATA[Line]]>
       </Descriptor>
       <Descriptor type='heading' path='buyer'>
        <![CDATA[Buyer]]>
```

```
        </Descriptor>
        <Descriptor type='heading' path='currency'>
          <![CDATA[Currency]]>
        </Descriptor>
      </MetaData>
      <DomainEntity domainClass='com.mapics.pm.PoItem'>
       <Key>
        <Property path='order'>
          <Value><![CDATA[000152]]></Value>
        </Property>
        <Property path='line'>
          <Value><![CDATA[003455]]></Value>
        </Property>
       </Key>
       <Property path='buyer' >
          <Value><![CDATA[BigCo]]></Value>
       </Property>
       <Property path='currency'>
          <Value><![CDATA[USD]]></Value>
       </Property>
      </DomainEntity>
      <DomainEntity domainClass='com.mapics.pm.PoItem'>
       <Key>
        <Property path='order'>
          <Value><![CDATA[000155]]></Value>
        </Property>
        <Property path='line'>
          <Value><![CDATA[003477]]></Value>
        </Property>
       </Key>
       <Property path='buyer' >
          <Value><![CDATA[BiggerCo]]></Value>
       </Property>
       <Property path='currency'>
          <Value><![CDATA[YEN]]></Value>
       </Property>
      </DomainEntity>
     </QueryRelationshipResponse>
   </QueryObjectResponse>
 </Response>
</SystemLink>
```

FetchNext

FetchNext tag

A FetchNext tag is used in conjunction with QueryListResponse to display any additional items that these response tags did not display (because their "maxReturned" values were less than the total number of items in their result sets).

   Attributes (may appear in any order)

name	an identifier for this FetchNext action. **Must be unique within a given System-Link request document.** Actions are named in the context of a given work area. If a subsequent action is requested with the same name for the same work area, any outstanding results for the old action will be reclaimed.
queryName	the query for which the additional results are to be displayed. **The named query will not be found if the QueryList action had its retainResults attribute set to "false".**

includeMetaData	(Optional) If set to "true", meta information on the selected properties (e.g. from the PQL "SELECT" clause) will be provided. (At the present time, the meta information for FetchNext is limited to the property "heading" value.) Defaults to "false".
maxReturned	the maximum number of instances returned. <u>For FetchNext, the value of maxReturned must be greater than 0.</u>

Notes:

- QueryList names are only valid in the scope of their associated SessionHandle and WorkHandle. Because of this, any FetchNext action must be performed inside of a Request using the same SessionHandle and WorkHandle with which the named query was originally run. If the Request uses a SessionHandle or WorkHandle different from that originally specified, the FetchNext would not find the remaining result set.
- The metainformation returned by FetchNext will be identical to that returned by the associated QueryList action. Inclusion of metadata on the FetchNext action by definition will increase the amount of data returned in the FetchNextResponse, and thus the associated network overhead. It is up to the user to determine whether this additional load is outweighed by the cost of storing the original metadata values from the QueryListResponse.

Example

This example shows the use of an existing session and work area (resetting the work area maxIdle time to 60 msec), to retrieve up to twenty additional domain objects from previous query "queryPoList".

```
<SystemLink version='1.0' hostVersion='1.0'>
  <Request sessionHandle='f345' workHandle='6c97d4' maxIdle='60'>
    <FetchNext name='fetchMoreFromPoList' queryName='queryPoList' maxReturned='20'/>
  </Request>
</SystemLink>
```

FetchNextResponse tag

A FetchNextResponse tag encompasses the result of a FetchNext request. The fetch result is linked back to the requesting fetch through the name attribute.

Attributes (may appear in any order)

name	the same name submitted to the FetchNext action tag.
queryName	name of the query specified in the FetchNext request.
actionSucceeded	set to "true" if the fetch action succeeded, otherwise set to "false"
moreResults	if there are more results in the query that have not yet been displayed, this is set to true.

Nested Tags

MetaData	if the request specified that metadata was to be included.
DomainEntity	contains a Domain Object instance.
Exception	if the FetchNext operation could not complete.

Example 1

This example is a possible response to the FetchNext example previously mentioned.

```
<SystemLink version='1.0' hostVersion='1.0'>
  <Response sessionHandle='f345' workHandle='6c97d4'>
    <FetchNextResponse name='fetchMoreFromPoList' queryName='queryPoList'
```

```
                              actionSucceeded='true' moreResults='false'>
       <DomainEntity domainClass='com.mapics.pm.PurchaseOrder'>
        <Key>
         <Property path='order'>
           <Value><![CDATA[00000020]]></Value>
         </Property>
        </Key>
        <Property path='relatedBuyer.buyerName'>
         <Value><![CDATA[Chipper Jones]]></Value>
        </Property>
        <Property path='relatedVendor.vendorName1'>
         <Value><![CDATA[PR-DamageControl.com]]></value>
        </Property>
       </DomainEntity>
       <DomainEntity domainClass='com.mapics.pm.PurchaseOrder'>
        <Key>
         <Property path='order'>
           <Value><![CDATA[00000021]]></value>
         </Property>
        </Key>
        <Property path='relatedBuyer.buyerName'>
         <Value><![CDATA[Keith Lockhart]]></Value>
        </Property>
        <Property path='relatedVendor.vendorName1'>
         <Value><![CDATA[Short Stops Unlimited]]></Value>
        </Property>
       </DomainEntity>
      </FetchNextResponse>
   </Response>
</SystemLink>
```

Example 2

This is an example response when a FetchNext action is used to request additional domain objects from a query that has no more elements to be displayed in its result set.

```
<SystemLink version='1.0' hostVersion='1.0'>
  <Response sessionHandle='f345' workHandle='6c97d4'>
   <FetchNextResponse name='fetchRel1' queryName='queryBuyers'
                        actionSucceeded='false' moreResults='false'>
    <Exception name='com.pjx.eScript.RequestDataException'>
     <Text><![CDATA[Named query not found]]></Text>
     <Message type='error'>
      <Text><![CDATA[Named query not found.]]></Text>
      <DetailedText><![CDATA[The named query is not found or has no further elements to
enumerate.]]></DetailedText>
     </Message>
     <Message type='information'>
      <Text><![CDATA[Check your XML]]></Text>
      <DetailedText><![CDATA[Please check your XML to ensure that the named query exists and has further
elements.]]></DetailedText>
     </Message>
    </Exception>
   </FetchNextResponse>
  </Response>
</SystemLink>
```

FetchNextText

FetchNextText tag

A FetchNextText tag is used in conjunction with QueryTextResponse to display any additional character data that these response tags did not display (because their "maxCharsReturned" values were less than the total number of characters in their respective text objects).

For an advanced ("typed") text object, all of the types specified in the original QueryText action that have more characters to display will be returned..

Attributes (may appear in any order)

name	an identifier for this FetchNextText action. **Must be unique within a given System-Link request document.** Actions are named in the context of a given work area. If a subsequent action is requested with the same name for the same work area, any outstanding results for the old action will be reclaimed.
queryTextName	the query text for which the additional results are to be displayed. **The named query text will not be found if the QueryText action had its retainResults attribute set to "false".**
includeMetaData	(Optional) If set to "true", meta information on the text object will be displayed. This will include the relationship description, as well as descriptions for each comment type (if the text object is "advanced" (typed)). Defaults to "false".
maxCharsReturned	the maximum number of characters returned. For FetchNextText, the value of maxReturned must be greater than 0.

Notes:

- QueryText names are only valid in the scope of their associated SessionHandle and WorkHandle. Because of this, any FetchNextText action must be performed inside of a Request using the same SessionHandle and WorkHandle with which the named query text was originally run. If the Request uses a SessionHandle or WorkHandle different from that originally specified, the FetchNextText would not find the remaining result set.
- The metainformation returned by FetchNextText will not necessarily be the same as that returned by QueryText. Specifically, for advanced ("typed") text objects, only those types with remaining characters to display will have corresponding entries in the MetaData section.

Example

This example shows the use of an existing session and work area (resetting the work area maxIdle time to 60 msec), to display up to 200 additional characters from previous query text "queryText_AdvancedComments" (shown in Example 2 under QueryTextResponse)

```
<SystemLink version='1.0' hostVersion='1.0'>
  <Request sessionHandle='f345' workHandle=' 6c97d4' maxIdle='60'>
    <FetchNextText name='fetchMoreAdvancedChars' queryTextName=' queryText_AdvancedComments'
              includeMetaData='true' maxCharsReturned='200'/>
  </Request>
</SystemLink>
```

FetchNextTextResponse tag

A FetchNextTextResponse tag encompasses the result of a FetchNextText request. The fetch text result is linked back to the requesting fetch text through the name attribute.

Attributes (may appear in any order)

name	the same name submitted to the FetchNext action tag.
queryTextName	name of the query specified in the FetchNext request.

relationshipName	name of the relationship associated with the initial QueryText operation.
actionSucceeded	set to "true" if the fetch action succeeded, otherwise set to "false"
moreChars	if more characters exist than have been displayed, this has a value of "true"

Nested Tags (either a TypedTextValue or a Value will appear, but never both)

MetaData	if the request specified that metadata was to be included. Note that for typed values, metadata will only
TypedTextValue	for an advanced (typed) text object, contains the text type and value for each type requested
Value	for a simple (non-typed) text object, contains the text value
Exception	if the FetchNextText request failed.

Note:
- If the "moreChars" attribute is "true", there will be a corresponding "moreChars" attribute appearing on one or more Value tags in the response. Value tags without a "moreChars" attribute reflect the default value (specified in the response DTD) of "false".

Example

This example is a fragment showing a possible response to the FetchNextText example previously given. Note that types "BB" and "CC", having no additional characters to display (as shown in the second QueryTextResponse example), do not appear either in the MetaData section or as TypedTextValues.

```
< FetchNextTextResponse name="fetchMoreAdvancedChars" queryTextName=' queryText_AdvancedComments'
                        relationshipName='relatedComments'
                        actionSucceeded='true'  moreChars="false">
    <MetaData>
      <TextDescriptor type="label">
        <Value><![CDATA[Advanced Comments]]></Value>
      </TextDescriptor>
      <TextDescriptor type="label">
        <Type><![CDATA[AA]]></Type>
        <Value><![CDATA[Engineering]]></Value>
      </TextDescriptor>
    </MetaData>

    <TypedTextValue>
      <Type><![CDATA[AA]]></Type>
      <Value><![CDATA[
              Engineering comment line
              ]]></Value>
    </TypedTextValue>
  </FetchNextTextResponse >
```

Example 2

This is an example response when a FetchNextText action is used to request additional text data from a query that has no more yet-to-be-displayed characters in its result set.

```
<SystemLink version='1.0' hostVersion='1.0'>
  <Response sessionHandle='f345' workHandle='6c97d4'>
    <FetchNextTextResponse name='fetchMoreText1' queryTextName='queryText_NoDataLeft'
                           actionSucceeded='false' moreChars='false'>
```

```
     <Exception name='com.pjx.eScript.RequestDataException'>
       <Text><![CDATA[Named text query not found]]></Text>
       <Message type='error'>
         <Text><![CDATA[Named query not found.]]></Text>
         <DetailedText><![CDATA[The named text query was not found or has no further data to
return.]]></DetailedText>
       </Message>
       <Message type='information'>
         <Text><![CDATA[Check your XML]]></Text>
         <DetailedText><![CDATA[Please check your XML to ensure that the named query exists and has further
elements.]]></DetailedText>
       </Message>
     </Exception>
   </FetchNextResponse>
 </Response>
</SystemLink>
```

TextFilter

TextFilter tag

The TextFilter tag is used in a QueryText action for advanced ("typed") text objects to indicate which types should be displayed in the response.

Attributes
(none)

Nested Tags:

Type	(one or more) used to indicate the text object type or types to display in the response

For an example of TextFilter, see Example 2 under the QueryText tag.

Environment Information Commands

These are inquiry actions used to obtain state information about the environment or session used in a given request.

ListCurrentEnvironment

ListCurrentEnvironment tag

A ListCurrentEnvironment action is used to obtain information about the environment associated with the given request.

Attribute

name	an identifier for this ListCurrentEnvironment action. **Must be unique within a given System-Link request document.** Actions are named in the context of a given work area. If a subsequent action is requested with the same name for the same work area, any outstanding results for the old action will be reclaimed.

Nested Tags: none

Example

This simple example shows the ListCurrentEnvironment action being used.

```
<System-Link>
    <Login userId='john doe' password='12345' maxIdle='900000'
        properties='com.mapics.cas.domain.EnvironmentId=7N,
                    com.mapics.cas.domain.SystemName=USATLD06.INFOR.COM,
                    com.mapics.cas.user.LanguageId=en'/>
    <Request sessionHandle='*current' workHandle='*new'
        broker='EJB' maxIdle='900000'>

      <ListCurrentEnvironment name='myEnvironment'/>

    </Request>
</System-Link>
```

ListCurrentEnvironmentResponse tag

A ListCurrentEnvironmentResponse tag encompasses the result of a ListCurrentEnvironment request. The ListCurrentEnvironmentResponse is linked back to its corresponding ListCurrentEnvironment action via the name attribute. A ListCurrentEnvironmentResponse will either contain an EnvironmentInfo structure or an exception if problems occurred.

Attributes (may appear in any order)

name	the same name submitted to the ListCurrentEnvironment action tag.
actionSucceeded	set to "true" if the ListCurrentEnvironment action succeeded, otherwise set to "false"

Nested Tags

EnvironmentInfo	contains the information about the current environment
Exception	(one or more) if the ListCurrentEnvironment operation could not complete.

This example corresponds to a successful completion of the ListCurrentEnvironment example:

```
<System-Link version='1.0' hostVersion='1.0'>
  <Request sessionHandle='f345' workHandle='6c97d4'>

    <ListCurrentEnvironmentResponse name='myEnvironment' actionSucceeded='true'>
      <EnvironmentInfo environmentId='7N' hostName='USATLD06.INFOR.COM'>
        <![CDATA[7N]]>
      </EnvironmentInfo>
    </ListCurrentEnvironmentResponse>

  </Request>
</System-Link>
```

EnvironmentInfo

EnvironmentInfo tag

An EnvironmentInfo tag is used inside of a ListCurrentEnvironmentResponse to return the information relevant to the current environment.

Attributes (may appear in any order)

environmentId	set to the current two-character XA environment Id
hostName	set to the host name associated with the XA environment

Nested Tags
a String value representing the environment description

See the ListCurrentEnvironment section for examples.

ListSessionInfo

ListSessionInfo tag

A ListSessionInfo action is used to obtain information about the session associated with the current request.

Attribute

name	an identifier for this ListSessionInfo action. **Must be unique within a given System-Link request document.** Actions are named in the context of a given work area. If a subsequent action is requested with the same name for the same work area, any outstanding results for the old action will be reclaimed.
showLocale	(Optional) if set to "true", locale (language and territory (if defined)) information will be returned. Default is "false".
showLocalCurrency	(Optional) if set to "true", the local currency (as defined in the Currency code file) will be returned. Default is "false".

Nested Tags: none

Note:
- Currently, only the default local currency is returned.
- If showLocalCurrency is set to true, the Currency code file must have an entry in which the "company ID" is 0, but the "currency ID" is non-blank. If no such entry exists, the ListSessionInfo action will fail.

Example

This simple example shows the ListSessionInfo action being used.

```
<System-Link>
    <Login userId='john doe' password='12345' maxIdle='900000'
        properties='com.mapics.cas.domain.EnvironmentId=7N,
                    com.mapics.cas.domain.SystemName=USATLD06.INFOR.COM,
                    com.mapics.cas.user.LanguageId=en'/>
    <Request sessionHandle='*current' workHandle='*new'
        broker='EJB' maxIdle='900000'>

    <ListSessionInfo name='mySession' showLocale='true' showLocalCurrency='true'/>

    </Request>
</System-Link>
```

ListSessionInfoResponse tag

A ListSessionInfoResponse tag encompasses the result of a ListSessionInfo request. The ListSessionInfoResponse is linked back to its corresponding ListSessionInfo action via the name attribute.

Attributes (may appear in any order)

name	the same name submitted to the ListSessionInfo action tag.
actionSucceeded	set to "true" if the ListSessionInfo action succeeded, otherwise set to "false"

Nested Tags

LocaleInfo	contains the locale information (language and territory). Only given if "showLocale" on ListSessionInfo was "true"
LocalCurrencyInfo	contains the local currency information. Only given if "showLocalCurrency" on ListSessionInfo was "true"
Exception	(one or more) if the ListSessionInfo operation could not complete.

Example

This example corresponds to a successful completion of the ListSessionInfo example. Note that no territory is defined for the current environment.

```
<System-Link version='1.0' hostVersion='1.0'>
    <Request sessionHandle='f345' workHandle='6c97d4'>

    <ListSessionInfoResponse name='myEnvironment' actionSucceeded='true'>
        <LocaleInfo language='en' territory='' />
        <LocalCurrencyInfo currencyId='USD' default='true'>
            <![CDATA[US DOLLARS]]>
        </LocalCurrencyInfo>
    </ListSessionInfoResponse>

    </Request>
</System-Link>
```

LocaleInfo

LocaleInfo tag

A LocaleInfo tag is used inside of a ListSessionInfoResponse to return the language and territory information associated with the current session. The locale information follows the international standard format of "xx-YY", where "xx" is the two-character language ID (e.g. "en", "fr"), and "YY" is the two character territory ID (e.g. "US", "GB".)

Attributes (may appear in any order)

language	set to the two-character language ID
territory	set to the two-character territory ID. Note that not all sessions will have a territory defined.

Nested Tags: none

See the ListSessionInfo section for examples.

LocalCurrencyInfo

LocalCurrencyInfo tag

A LocalCurrencyInfo tag is used inside of a ListSessionInfoResponse to return the local currency information associated with the current session. Currently, only the default local currency will be returned.

Attributes (may appear in any order)

currencyId	set to the currency ID
default	set to "true", since this will be the default local currency

Nested Tags
a String value representing the currency description

See the ListSessionInfo section for examples.

Maintenance Commands

Note: business object attachments have special requirements, and thus are handled with a dedicated set of System-Link actions. For more information, see **Attachment Commands**.

Create and Copy

Create tag

A Create tag is used to create a new instance of a business object (e.g. Purchase Order). If the optional SourceObject nested tag is specified, then this action becomes a Copy.

Attributes (may appear in any order)

name	an identifier for this Create action. **Must be unique within a given System-Link request document.** Actions are named in the context of a given work area. If a subsequent action is requested with the same name for the same work area, any outstanding results for the old action will be reclaimed.
domainClass	(Optional) the domain class over which the create action applies. Either a domainClass or a classMnemonic must be provided.
classMnemonic	(Optional) the class identifier corresponding to the domain class over which the create action applies. Either a domainClass or a classMnemonic must be provided.
retainResult	(Optional) If set to "true", the resulting "named object" (with name set to the name of this Create name identifier) will be retained for use in subsequent "value references". Default is "false".

Nested Tags (in the order that they must be specified)

LogicalTransaction	(Optional) If specified, the maintenance action is performed in the scope of the logical transaction given.
ReasonCode	(Optional) If specified, the maintenance action is performed with the reason code given.
ApplyTemplate	(Optional) If specified, the named Create or Copy (as applicable) template will be used. **Note: unlike the custom definition key tags in query operations (e.g. HeaderDfnKey), no default Template will be used if ApplyTemplate is not specified.**
SourceObject	(Optional) If specified, this action becomes a Copy action (i.e. the values in maintainable fields in the named object will be copied to the new object).
MaintenanceOptions	(Optional) If specified, lists the options (create or copy) to apply when creating or copying the business object.
DomainEntity	specifies the properties (key and non-key) for the new object. Key properties appearing in the non-key section will be ignored.
MaintainText	(Optional) If specified, more or more text actions are performed on the new text object(s) associated with the new business object.

Notes:

- (for Create and Copy only – does not apply to SourceObject or other maintenance tags) if a key property is system generated (e.g. "itemLocationToken" on Item Location), then this property should not be specified in the Key section. If all key properties for a given object are system generated, then the Key section can be omitted. Optionally, the "empty" Key section can have the following format:
  <Key>
  </Key>
- Most maintenance actions (Create / Copy, Update, and Delete) are "auto-commit": each action runs and

commits in its own separate transaction. The exceptions to this are those actions that are implicitly grouped (i.e. the associated business objects have the related operations performed on the UJOB, e.g. creation of a ManufacturingOrder) or explicitly grouped (i.e. appear after a StartTransactionGroup action in the request document).

Example 1

This example shows a Buyer being created, using the reason code "NEWB" and the named buyer create template "MyBuyerTemplate". Since no SourceObject appears, this is not a Copy action.

```
<System-Link>
   <Login userId='john doe' password='12345' maxIdle='900000'
      properties='com.mapics.cas.domain.EnvironmentId=7N,
                  com.mapics.cas.domain.SystemName=USATLD06.INFOR.COM,
                  com.mapics.cas.user.LanguageId=en'/>
   <Request sessionHandle='current' workHandle='*new'
      broker='EJB' maxIdle='900000'>


    <Create name='newBuyer' domainClass='com.mapics.pm.Buyer'>
     <ReasonCode><![CDATA[NEWB]]></ReasonCode>
     <ApplyTemplate clientClass='com.mapics.pm.Buyer'><![CDATA[MyBuyerTemplate]]></ApplyTemplate>
     <DomainEntity>
      <Key>
       <Property path='buyer'>
         <Value><![CDATA[BIGB]]></Value>
       </Property>
      </Key>
      <Property path='buyerName'>
        <Value><![CDATA[Unico Incorporated]]></Value>
      </Property>
      <Property path='buyerTelephone'>
        <Value><![CDATA[800-555-1212]]></Value>
      </Property>
      <Property path='fax'>
        <Value><![CDATA[919-123-4567]]></Value>
      </Property>
      <Property path='buyerUserid'>
        <Value><![CDATA[BobUser]]></Value>
      </Property>
      <Property path='EMailAddress'>
        <Value><![CDATA[Bob.User@unico.com]]></Value>
      </Property>
     </DomainEntity>
    </Create>
   </Request>
</System-Link>
```

Example 2

This example shows a Buyer being created, using an existing Buyer "OLDB" as a copy, with the operation running under logical transaction "CPYBYR". No copy template is specified. The result of this create is retained for later use as a named object.

```
<System-Link>
   <Login userId='john doe' password='12345' maxIdle='900000'
      properties='com.mapics.cas.domain.EnvironmentId=7N,
                  com.mapics.cas.domain.SystemName=USATLD06.INFOR.COM,
                  com.mapics.cas.user.LanguageId=en'/>
```

```
        <Request sessionHandle='*current' workHandle='*new'
          broker='EJB' maxIdle='900000'>
        <Create name='newBuyer' domainClass='com.mapics.pm.Buyer' retainResult='true'>
          <LogicalTransaction><![CDATA[CPYBYR]]></LogicalTransaction>
          <SourceObject>
            <Key>
              <Property path='buyer'>
                <Value><![CDATA[OLDB]]></Value>
              </Property>
            </Key>
          </SourceObject>
          <DomainEntity>
            <Key>
              <Property path='buyer'>
                <Value><![CDATA[BIGB]]></Value>
              </Property>
            </Key>
            <Property path='buyerName'>
              <Value><![CDATA[Unico Incorporated]]></Value>
            </Property>
            <Property path='buyerTelephone'>
              <Value><![CDATA[800-555-1212]]></Value>
            </Property>
            <Property path='fax'>
              <Value><![CDATA[919-123-4567]]></Value>
            </Property>
          </DomainEntity>
        </Create>
      </Request>
</System-Link>
```

CreateResponse tag

A CreateResponse tag encompasses the result of a Create request. The CreateResponse is linked back to its corresponding Create action via the name attribute. A CreateResponse will contain either the Key of the new object, or an exception if problems occurred during Create / Copy. The response will also contain MaintainTextResponses for all MaintainText actions nested in the Create request.

Attributes (may appear in any order)

name	the same name submitted to the Create action tag.
actionSucceeded	set to "true" if the create action succeeded, otherwise set to "false"

Nested Tags

Key	contains a list of key properties for the given domain class, as well as the corresponding values needed to uniquely reference the desired object.
MaintainTextResponse	contains the results of a corresponding MaintainText action.
Exception	(one or more) if the Create or Copy operation could not complete.

Example

This example corresponds to a successful completion of Create Example 1:
```
<System-Link>
  <Request sessionHandle='f345' workHandle='6c97d4'>
    <CreateResponse name='newBuyer' actionSucceeded='true'>
      <Key>
        <Property path='buyer'>
```

```
        <Value><![CDATA[BIGB]]></Value>
      </Property>
    </Key>
  </CreateResponse>
 </Request>
</System-Link>
```

Update

Update tag

An Update tag is used to perform an Update operation on an existing object.

Attributes (may appear in any order)

name	an identifier for this Update action. **Must be unique within a given System-Link request document.** Actions are named in the context of a given work area. If a subsequent action is requested with the same name for the same work area, any outstanding results for the old action will be reclaimed.
domainClass	(Optional) the domain class over which the update action applies. Either a domainClass or a classMnemonic must be provided.
classMnemonic	(Optional) the class identifier corresponding to the domain class over which the update action applies. Either a domainClass or a classMnemonic must be provided.
retainResult	(Optional) If set to "true", the resulting "named object" (with name set to the name of this Update name identifier) will be retained for use in subsequent "value references". Default is "false".

Nested Tags (in the order that they must be specified)

LogicalTransaction	(Optional) If specified, the maintenance action is performed in the scope of the logical transaction given.
ReasonCode	(Optional) If specified, the maintenance action is performed with the reason code given.
ApplyTemplate	(Optional) If specified, the named Update template will be used. **Note: unlike the custom definition key tags in query operations (e.g. HeaderDfnKey), no default Template will be used if ApplyTemplate is not specified.**
DomainEntity	specifies the key properties, and the non-key properties to be updated for the object. Key properties appearing in the non-key section will be ignored.
MaintainText	(Optional) If specified, more or more text actions are performed on the text object(s) associated with the named business object.

Note:

- Most maintenance actions (Create / Copy, Update, and Delete) are "auto-commit": each action runs and commits in its own separate transaction. The exceptions to this are those actions that are implicitly grouped (i.e. the associated business objects have the related operations performed on the UJOB, e.g. creation of a ManufacturingOrder) or explicitly grouped (i.e. appear after a StartTransactionGroup action in the request document).

Example

This example shows the telephone and fax numbers for Buyer 'B132' being updated, using the reason code "CHGB" and the buyer update template referenced by object Id.

```
<System-Link>
    <Login userId='john doe' password='12345' maxIdle='900000'
        properties='com.mapics.cas.domain.EnvironmentId=7N,
                    com.mapics.cas.domain.SystemName=USATLD06.INFOR.COM,
                    com.mapics.cas.user.LanguageId=en'/>
    <Request sessionHandle='current' workHandle='*new'
        broker='EJB' maxIdle='900000'>
     <Update name='updateBuyer' domainClass='com.mapics.pm.Buyer'>
      <ReasonCode><![CDATA[CHGB]]></ReasonCode>
      <ApplyTemplate objectId='f346' clientClass='com.mapics.pm.Buyer'/>
      <DomainEntity>
       <Key>
         <Property path='buyer'>
           <Value><![CDATA[B132]]></Value>
         </Property>
        </Key>
        <Property path='buyerTelephone'>
           <Value><![CDATA[877-111-2222]]></Value>
        </Property>
        <Property path='fax'>
           <Value><![CDATA[281-534-1232]]></Value>
        </Property>
       </DomainEntity>
      </Update>
    </Request>
</System-Link>
```

UpdateResponse tag

An UpdateResponse tag encompasses the result of an Update request. The UpdateResponse is linked back to its corresponding Update action via the name attribute. If the Update action fails, the UpdateResponse will contain an exception detailing the reason for failure. In addition, the UpdateResponse will also contain MaintainTextResponses for all MaintainText actions nested in the Update request.

Attributes (may appear in any order)

name	the same name submitted to the Update action tag.
actionSucceeded	set to "true" if the update action succeeded, otherwise set to "false"

Nested Tags

MaintainTextResponse	appears if an error occurred while processing a MaintainText action.
Exception	(one or more) if the Update operation could not complete.

Example

This example shows a possible response if the Update example fails:
```
<System-Link>
  <Request sessionHandle='f345' workHandle='6c97d4'>
   <UpdateResponse name='newBuyer' actionSucceeded='false'>
    <Exception name='com.pjx.eScript.ObjectNotFoundException'>
     <Text><![CDATA[Specified object not found]]></Text>
     <Message type='error'>
      <Text><![CDATA[Buyer 'B132' could not be found]]></Text>
     </Message>
    </Exception>
   </UpdateResponse>
```

```
     </Request>
  </System-Link>
```

Delete

Delete tag

A Delete tag is used to perform a deletion of an existing object.

Attributes (may appear in any order)

name	an identifier for this Delete action. **Must be unique within a given System-Link request document.** Actions are named in the context of a given work area. If a subsequent action is requested with the same name for the same work area, any outstanding results for the old action will be reclaimed.
domainClass	(Optional) the domain class over which the delete action applies. Either a domainClass or a classMnemonic must be provided.
classMnemonic	(Optional) the class identifier corresponding to the domain class over which the delete action applies. Either a domainClass or a classMnemonic must be provided.

Nested Tags (in the order that they must be specified)

LogicalTransaction	(Optional) If specified, the maintenance action is performed in the scope of the logical transaction given.
ReasonCode	(Optional) If specified, the maintenance action is performed with the reason code given.
DomainEntity	specifies the key properties for the object. Properties not appearing in the Key section will be ignored.

Notes:

- Most maintenance actions (Create / Copy, Update, and Delete) are "auto-commit": each action runs and commits in its own separate transaction. The exceptions to this are those actions that are implicitly grouped (i.e. the associated business objects have the related operations performed on the UJOB, e.g. creation of a ManufacturingOrder) or explicitly grouped (i.e. appear after a StartTransactionGroup action in the request document).
- If the Delete action is performed against an object that is currently stored as a "named object", that named object will no longer be valid.

Example

This example shows the deletion of Buyer 'OLDB' with reason code "DELO".

```
<System-Link>
    <Login userId='john doe' password='12345' maxIdle='900000'
        properties='com.mapics.cas.domain.EnvironmentId=7N,
                com.mapics.cas.domain.SystemName=USATLD06.INFOR.COM,
                com.mapics.cas.user.LanguageId=en'/>
    <Request sessionHandle='current' workHandle='*new'
        broker='EJB' maxIdle='900000'>
    <Delete name='updateBuyer' domainClass='com.mapics.pm.Buyer'>
     <ReasonCode><![CDATA[DELO]]></ReasonCode>
     <DomainEntity>
      <Key>
        <Property path='buyer'>
          <Value><![CDATA[OLDB]]></Value>
        </Property>
      </Key>
```

```
        </DomainEntity>
      </Delete>
    </Request>
</System-Link>
```

DeleteResponse tag

A DeleteResponse tag encompasses the result of a Delete request. The DeleteResponse is linked back to its corresponding Delete action via the name attribute. If the Delete action fails, the DeleteResponse will contain an exception detailing the reason for failure.

Attributes (may appear in any order)

name	the same name submitted to the Delete action tag.
actionSucceeded	set to "true" if the delete action succeeded, otherwise set to "false"

Nested Tags

Exception	(one or more) if the Delete operation could not complete.

Example

This example shows a possible response if the Delete example fails:
```
<System-Link>
  <Request sessionHandle='f345' workHandle='6c97d4'>
    <DeleteResponse name='newBuyer' actionSucceeded='false'>
      <Exception name='com.pjx.eScript.SecurityException'>
        <Text><![CDATA[Security violation during delete]]></Text>
        <Message type='error'>
          <Text><![CDATA[User 'john doe' not authorized to delete 'OLDB']]></Text>
        </Message>
      </Exception>
    </DeleteResponse>
  </Request>
</System-Link>
```

SourceObject

SourceObject tag

A SourceObject tag is used within a Create action tag to perform a Copy. For those business objects that permit a copy from a business object of a different class (e.g. CustomerOrder to StandingOrder), the domainClass or classMnemonic attributes are used to specify the domain class of that SourceObject. If neither a domainClass nor classMnemonic are specified, the SourceObject is assumed to have the same domain class as the new destination object instance.

Attributes (may appear in any order)

domainClass	(Optional) the domain class of the source object. Either a domainClass or a classMnemonic may be provided.
classMnemonic	(Optional) the class identifier corresponding to the domain class of the source object. Either a domainClass or a classMnemonic may be provided.

Nested Tags

Key	specifies the key properties for the object.

MaintenanceOptions tag

A MaintenanceOptions tag is used within a Create action tag to specify options to be performed along with the creation of the new instance. If a SourceObject tag is present, then the MaintenanceOptions must be Copy options. Otherwise, they must be Create options.

Attributes: none

Nested Tags

RelationshipOption	(Copy action only) specifies that the related objects associated with the given relationship be copied as well.
Option	specifies the "simple" options to be performed with the create or copy action.

Example 1

This example is a request fragment showing a Create action with associated create options:

```
<Create name='createObject_PurchaseOrderItem' domainClass='com.mapics.pm.PoItem'>
  <ApplyTemplate clientClass='com.mapics.pm.PoItem'><![CDATA[(none)]]></ApplyTemplate>
  <MaintenanceOptions>
    <Option optionName='includeVendorItemComments'>
      <Value><![CDATA[1]]></Value>
    </Option>
    <Option optionName='includeItemReceivingOperations'>
      <Value><![CDATA[0]]></Value>
    </Option>
  </MaintenanceOptions>
  <DomainEntity>
      …
  </DomainEntity>
</Create>
```

Example 2

This example is a request fragment showing a Copy action with associated copy options:

```
<Create name='createObject_PurchaseOrder' domainClass='com.mapics.pm.PurchaseOrder'>
  <SourceObject>
    <Key>
      <Property path='order'>
        <Value><![CDATA[PASMU]]></Value>
      </Property>
    </Key>
  </SourceObject>

  <MaintenanceOptions>
    <RelationshipOption relationshipName='relatedPoComments'/>
    <RelationshipOption relationshipName='relatedPoItems'>
      <ApplyTemplate objectId='Z1105RA9M145780204271AAAAAAAAADA'
                     clientClass='com.mapics.pm.PoItem'><![CDATA[MyTemplate]]></ApplyTemplate>
      <RelationshipOption relationshipName='relatedPoItemComments'/>
      <RelationshipOption relationshipName='relatedPoItemReleases'>
        <ApplyTemplate clientClass='com.mapics.pm.PoItemRelease'><![CDATA[(none)]]></ApplyTemplate>
      <RelationshipOption relationshipName='relatedPoItemReleaseComments'/>
    </RelationshipOption>
    <RelationshipOption relationshipName='relatedPoItemOperations'>
```

```
        <ApplyTemplate clientClass='com.mapics.pm.PoItemReceivingOperation'>
                      <![CDATA[(none)]]></ApplyTemplate>
    </RelationshipOption>
  </MaintenanceOptions>
  <DomainEntity>
     …
  </DomainEntity>
</Create>
```

RelationshipOption

RelationshipOption tag

The RelationshipOption tag is used in a copy operation to indicate that the related objects associated with the given relationship that should be copied as well.

Attributes

relationshipName	the name of the associated relationship

Nested Tags

ApplyTemplate	(Optional) if specified, the named copy template will be applied during the copy of the related objects.
RelationshipOption	(Optional) if specified, the related objects associated with the given relationship are copied as well.
Option	(Optional) if specified, the "simple" copy option is applied during the copy operation.

See the MaintenanceOptions section for examples.

Option

Option tag

The Option tag is used in a create or copy operation to specify the "simple" (non-relationship and non-template) options to be used during the operation.

Attributes

optionName	the name of the "simple" option

Nested Tags (either Value, ValueRef, NullValue may be specified)

Value	value of the option as a String.
ValueRef	use the same value for this option as that found on the given property for the "named object" (returned by QueryObject, Create / Copy, or Update).
NullValue	specifies that the value of the option is null. (Note: only valid for options that support null

See the MaintenanceOptions section for examples.

MaintainText

MaintainText tag

A MaintainText action is used within a Create / Copy or Update action to perform maintenance on the text objects associated with the specified business object.

Attributes (may appear in any order)

name	an identifier for this text maintenance action. **Must be unique within a given System-Link request document.** Actions are named in the context of a given work area. If a subsequent action is requested with the same name for the same work area, any outstanding results for the old action will be reclaimed.
relationshipName	name of the relationship that associates the text object with the given business object.

Nested Tags

TextAction	specifies the text data and action attributes required for the text action

Notes:

- All MaintainText actions for a given relationship will be processed in the order in which they appear. Inside of a single MaintainText action, all TextActions will be processed in the order in which they appear.
- A given text object may have more than one MaintainText action applied to it in a single request. Inside of a given MaintainText action, more than one TextAction may also be applied to a given text object (or text object type, in the case of advanced ("typed") text objects).
- For CSM business objects, comment insertions always occur at the beginning (as displayed in Power-Link). As a result, using the "addToEnd" (default) TextAction actionType is synonymous with "addToBeginning". The "replace" actionType behavior is the same as for other objects.

Example 1

This example is a request fragment showing two maintenance actions on two different simple ("non-typed") text objects. The first action will append the given text data to its text object, whereas the second will replace any existing text data in its text object with the given text data:

```
<Update name='updateObject_ItemRevision' domainClass='com.mapics.epdm.ItemRevision'>
  <DomainEntity>
    …
  </DomainEntity>
  <MaintainText name='maintainText_Comments' relationshipName='relatedItemRevisionComments'>
    <TextAction valueType='manual'>
      <Value><![CDATA[One new line of comment text
                      Another new line of comment text]]></Value>
    </TextAction>
  </ MaintainText>
  < MaintainText name='maintainText_OtherComments' relationshipName='relatedItemRevisionCommentsAlt'>
    <TextAction valueType='manual' actionType='replace'>
      <Value><![CDATA[Replacement comment text]]></Value>
    </TextAction>
  </MaintainText >
</Update>
```

Example 2

This example is a request fragment showing two maintenance actions on two different advanced ("typed" text objects. The first action will append the given text data to the two text object types specified, whereas the second will insert the given text data in front of any existing text data present in the three text object types specified:

```
<Update name='updateObject_ItemRevision' domainClass='com.mapics.epdm.ItemRevision'>
  <DomainEntity>
     …
  </DomainEntity>
  < MaintainText name='maintainText_AdvancedComments' relationshipName='relatedIRAdvComments'>
    <TextAction valueType='manual'>
     <Type><![CDATA[AA]]></Type>
     <Value><![CDATA[One new comment line for type AA]]></Value>
    </TextAction>
    <TextAction valueType='manual'>
     <Type><![CDATA[BB]]></Type>
     <Value><![CDATA[One new comment line for type BB]]></Value>
    </TextAction>
  </ MaintainText >
  < MaintainText name='maintainText_AdvancedComments'2 relationshipName='relatedIRAdvComments2'>
    <TextAction valueType='manual' actionType='addToBeginning'>
     <Type><![CDATA[CC]]></Type>
     <Value><![CDATA[One new comment line for type CC]]></Value>
    </TextAction>
    <TextAction valueType='manual' actionType='addToBeginning'>
     <Type><![CDATA[DD]]></Type>
     <Value><![CDATA[One new comment line for type DD]]></Value>
    </TextAction>
    <TextAction valueType='manual' actionType='addToBeginning'>
     <Type><![CDATA[EE]]></Type>
     <Value><![CDATA[One new comment line for type EE]]></Value>
    </TextAction>
  </ MaintainText >
</Update>
```

MaintainTextResponse tag

A MaintainTextResponse tag encompasses the result of a MaintainText request. The MaintainTextResponse is linked back to its corresponding MaintainText action via the name attribute. If the MaintainText action fails, the MaintainTextResponse will contain an exception detailing the reason for failure.

Attributes (may appear in any order)

name	the same name submitted to the MaintainText action tag.
relationshipName	the same relationship name submitted to the MaintainText action tag.
actionSucceeded	set to "true" if the MaintainText action succeeded, otherwise set to "false"

Nested Tags

Exception	(one or more) if the MaintainText operation could not complete.

TextAction

TextAction tag

A TextAction tag is used inside of a MaintainText action to group comment text with the specific way that it needs to be applied to the text object.

Attributes (may appear in any order)

valueType	indicates how the character data in the Value subtag should be interpreted. Currently, valueType must be given a value of "manual".
actionType	(Optional) indicates where in the current comment text the given character data should be applied. Must be one of the following: "addToBeginning", "addToEnd", or "replace". Default is "addToEnd".

Nested Tags

Type	(For typed objects only) indicates the text object type to which the data should be applied
Value	contains the text character data

Notes:

- A TextAction having an actionType of "replace" and containing an empty Value (e.g. "<Value></Value>") can be used to delete the existing comment text. TextActions having any other actionType and containing empty Values are ignored.
- For CSM business objects, comment insertions always occur at the beginning (as displayed in Power-Link). As a result, using the "addToEnd" (default) actionType is synonymous with "addToBeginning". The "replace" actionType behavior is the same as for other objects.

See the MaintainText section for examples.

Attachment Commands

Business object attachments, both due to their nature (e.g. "global" versus "specific" attachments) and due to their underlying implementation, have unique requirements that would make using the "standard" System-Link actions (QueryObject, Update, etc.) cumbersome. As a result, attachments are handled with their own set of inquiry and maintenance actions.

In general, these commands will behave in a fashion similar to the "standard" actions. The main difference is that attachment key properties are never directly specified: all attachment keys are derived from both their "owning class", their associated media file, and (in the case of specific attachments) the key of the business object instance that owns the attachment.

QueryAttachmentList

QueryAttachmentList tag

A QueryAttachmentList tag is used to query attachments associated with a domain class or an instance of a domain class. The value of the QueryAttachmentList tag is an expression that selects a list of attachments.

Attributes (may appear in any order)

name	an identifier for this query. **Must be unique within a given System-Link request document.** Queries are named in the context of a given work area. If a subsequent query is requested with the same name for the same work area, any outstanding results for the old query will be reclaimed.
owningDomainClass	(Optional) the fully-qualified domain class to which all attachments in the query belong. Either an owningDomainClass or an owningClassMnemonic must be provided.
owningClassMnemonic	(Optional) the class identifier corresponding to the domain class to which all attachments in the query belong. Either an owningDomainClass or an owningClassMnemonic must be provided.
includeGlobals	(Optional) If set to "true", global attachments (those belonging to all instances of a given domain class) will be included in the list. **If includeGlobals is set to "false", an OwningObjectKey tag must be given.** Defaults to "true".
includeMetaData	(Optional) If set to "true", meta information on the selected properties (e.g. from the PQL "SELECT" clause) will be provided. (At the present time, the meta information for QuetyAttachmentList is limited to the property "heading" value.) Defaults to "false".
maxReturned	the maximum number of instances returned in the associated QueryAttachmentListResponse. For QueryAttachmentList, the value of maxReturned must be greater than or equal to 0.
retainResult	(Optional) If set to "true", more results can later be obtained with the FetchNextAttachments action. Defaults to "true".

Nested Tags (in the order that they must be specified - either a Pql or a QueryDfn tag must be specified.)

Pql	contains a free-format query request. If a Pql tag is given, then no QueryDfn tag should be specified.
QueryDfn	contains references to custom definitions (header, subset, sort) used to create the list. If a QueryDfn tag is given, then no Pql tag should be specified.

OwningObjectKey	(Optional) for specific attachments (those belonging to a single instance of a given domain class), contains the key properties and values for the owning domain object. **Must be provided if includeGlobals is set to "false".**

Example 1

This example fragment queries attachments associated with purchase orders (specified by "owningDomainClass"). Only those attachments specific to purchase order "PAA" (given in the "OwningObjectKey" subtag) will be returned. The Pql subtag is used to specify the fields required and the ordering. MetaData for all properties in the SELECT clause will be included in the response.

```
<QueryAttachmentList name='queryListAttachment_General'
                owningDomainClass='com.mapics.pm.PurchaseOrder'
                includeGlobals='false' includeMetaData='true' maxReturned='10'>

    <Pql>
       <![CDATA[SELECT relatedMediaFile.fileDescription,
                objectType,objectId,relatedMediaFile.view,
                relatedMediaFile.open,relatedMediaFile.print,
                global REQUIRED relatedMediaFile ORDER
                BY objectClass,objectKey]]>
    </Pql>

    <OwningObjectKey>
       <Property path='order'>
          <Value><![CDATA[PAA]]></Value>
       </Property>
    </OwningObjectKey>

</QueryAttachmentList>
```

Example 2

This example is the QueryDfn version of the above example, except that only attachments associated with all purchase orders (global) will be returned. This is due to both the omission of the OwningObjectKey section, as well as the default value ("true") for includeGlobals.

```
<QueryAttachmentList name='queryListAttachment_General'
                owningDomainClass='com.mapics.pm.PurchaseOrder'
                includeMetaData='true' maxReturned='10'>

    <QueryDfn>
       <HeaderDfnKey clientClass='com.mapics.cas.domain.Attachment'><![CDATA[General]]></HeaderDfnKey>
       <SortDfnKey clientClass='com.mapics.cas.domain.Attachment'><![CDATA[(default)]]></SortDfnKey>
    </QueryDfn>

</QueryAttachmentList>
```

QueryAttachmentListResponse tag

A QueryAttachmentListResponse tag encompasses the result of a QueryAttachmentList request. The query result is linked back to the requesting query through the name attribute.

Attributes (may appear in any order)

name	the same name submitted to the QueryAttachmentList action tag.
requestedOwningDomainClass	the owning domain class requested in the QueryAttachmentList request.
actionSucceeded	set to "true" if the query action succeeded, otherwise set to "false"
moreResults	if more results exist than have been displayed, this has a value of "true". Note: these additional results can only be retrieved via FetchNextAttachments if "retainResult" was set to "true" on the QueryAttachmentList action.

Nested Tags

MetaData	if the request specified that metadata was to be included.
ColumnStatistics	if the request contains column statistics (e.g. COUNT, MIN, MAX, AVG)
DomainEntity	contains a single attachment instance.
Exception	if the QueryAttachmentList has failed.

Example

This example is a fragment of a possible response to the QueryAttachmentList Example 1 previously mentioned. Note that no Key sections appear in the DomainEntity elements.

```
<QueryAttachmentListResponse name="queryListAttachment_General"
                             requestedOwningDomainClass="com.mapics.pm.PurchaseOrder"
                             actionSucceeded="true" moreResults="true">
 <MetaData>
   <Descriptor type="heading"
               path="relatedMediaFile.fileDescription"><![CDATA[File description]]></Descriptor>
   <Descriptor type="heading" path="objectType"><![CDATA[Object type]]></Descriptor>
   <Descriptor type="heading" path="objectId"><![CDATA[Object ID]]></Descriptor>
   <Descriptor type="heading" path="relatedMediaFile.view"><![CDATA[View]]></Descriptor>
   <Descriptor type="heading" path="relatedMediaFile.open"><![CDATA[Open]]></Descriptor>
   <Descriptor type="heading" path="relatedMediaFile.print"><![CDATA[Print]]></Descriptor>
   <Descriptor type="heading" path="global"><![CDATA[Global]]></Descriptor>
 </MetaData>
 <DomainEntity domainClass="com.mapics.cas.domain.Attachment">
   <Property path="relatedMediaFile.fileDescription">
      <Value><![CDATA[SpecialOrderDetails.doc]]></Value>
   </Property>
   <Property path="objectType">
      <Value><![CDATA[PURCHASE ORDER]]> </Value>
   </Property>
   <Property path="objectId">
      <Value><![CDATA[PAA]]></Value>
   </Property>
   <Property path="relatedMediaFile.view">
      <Value>1</Value>
   </Property>
   <Property path="relatedMediaFile.open">
      <Value>0</Value>
   </Property>
```

```
    <Property path="relatedMediaFile.print">
       <Value>0</Value>
    </Property>
    <Property path="global">
       <Value>0</Value>
    </Property>
 </DomainEntity>
 <DomainEntity domainClass="com.mapics.cas.domain.Attachment">

    …
```

</**QueryAttachmentListResponse**>

QueryAttachmentObject

QueryAttachmentObject tag

The QueryAttachmentObject tag is used to query a single object. The value of the QueryAttachmentObject tag is an expression that selects exactly one object.

Attributes (may appear in any order)

name	an identifier for this query. **Must be unique within a given System-Link request document.** Actions are named in the context of a given work area. If a subsequent action is requested with the same name for the same work area, any outstanding results for the old action will be reclaimed.
owningDomainClass	(Optional) the fully-qualified domain class to which the attachment in the query belongs. Either an owningDomainClass or an owningClassMnemonic must be provided.
owningClassMnemonic	(Optional) the class identifier corresponding to the domain class to which the attachment in the query belongs. Either an owningDomainClass or an owningClassMnemonic must be provided.
includeMetaData	(Optional) if set to "true", meta information on the selected properties (e.g. from the PQL "SELECT" clause) will be provided. (At the present time, the meta information for QueryObject is limited to the property "label" value.) Default is "false".
showResult	(Optional) if set to "false", a successful object retrieval will result in no response being generated. This is useful in combination with retainResult set to "true" when only actions based upon the "named object" are of interest. Default is "true".
retainResult	(Optional) If set to "true", the resulting "named object" (with name set to the name of this QueryObject name identifier) will be retained for use in subsequent "value references". Default is "false".

Nested Tags (in the order that they must be specified – if a Pql tag is specified, then no CardFileDfnKey may appear.)

Pql	contains a free-format query request. This statement must return zero or one object. (If more than one object would be returned, an exception will be returned.) If a Pql tag is specified, then no CardFileDfnKey tag may be specified.
CardFileDfnKey	(Optional) identifies the card file to be used to determine which properties to display for the selected object. If no CardFileDfnKey is given, and if no Pql tag is present, then the default card file is used from the user's preferences.
MediaFilePath	The absolute path of the media file associated with the attachment.

OwningObjectKey	(Optional) for a specific attachment (one belonging to a single instance of a given domain class), contains the key properties and values for the owning domain object. **Must be omitted for a global attachment.**

Example 1

This example fragment requests the attachment object specific to purchase order "PAA", associated with the media file "SpecialDetails.doc".

```
QueryAttachmentObject name='queryAttachmentObject_Attachment_Default'
                      owningDomainClass='com.mapics.pm.PurchaseOrder' includeMetaData='true'>
  <Pql>
     <![CDATA[SELECT fileDescription,global,createUser,
              createDate,changeUser,changeDate,pathAndName,
              file,view,open,print,createUserMF,createDateMF,
              changeUserMF,changeDateMF,userFieldCodeA,
              userFieldCodeB,userFieldCodeC,userFieldDate1,
              userFieldDate2,userFieldDate3,userFieldSwitchA,
              userFieldSwitchB,userFieldSwitchC,userFieldSwitchD,
              userFieldText25,userFieldText40]]>
  </Pql>

  <MediaFilePath><![CDATA[C:\Attachments\SpecialDetails.doc]]></MediaFilePath>

  <OwningObjectKey>
     <Property path='order'>
        <Value><![CDATA[PAA]]></Value>
     </Property>
  </OwningObjectKey>
</QueryAttachmentObject>
```

Example 2

This example fragment uses the CardFileDfnKey to query a global attachment associated with all purchase orders, with associated media file "OverallRecommendations.pdf". Note that if the CardFileDfnKey element were omitted, the default card file from the user's preferences would be used.

```
<QueryAttachmentObject name='queryAttachmentObject_Attachment_Default'
                       owningDomainClass='com.mapics.pm.PurchaseOrder' includeMetaData='true'>
   <CardFileDfnKey objectId='SATTACH-CF-DEFAULT'
                   clientClass='com.mapics.cas.domain.Attachment'></CardFileDfnKey>
   <MediaFilePath><![CDATA[C:\General\OverallRecommendations.pdf]]></MediaFilePath>
</QueryAttachmentObject>
```

QueryAttachmentObjectResponse tag

A QueryAttachmentObjectResponse tag encompasses the result of a QueryAttachmentObject request. The query result is linked back to the requesting query through the name attribute.

Attributes (may appear in any order)

name	the same name submitted to the QueryAttachmentObject tag. If "retainResult" was set to "true", this name can be used to reference the retrieved object for later value references.
requestedOwningDomainClass	the owning domain class requested by the QueryAttachmentObject tag.
actionSucceeded	set to "true" if the query action succeeded, otherwise set to "false"

Nested Tags

MetaData	if the request specified that metadata was to be included.
DomainEntity	contains a Domain Object instance.
Exception	if the QueryObject request failed.

Example

This example is a possible response fragment to the first QueryAttachmentObject example previously mentioned. Note that no Key section appears in the DomainEntity element.

```
<QueryAttachmentObjectResponse name="queryAttachmentObject_Attachment_Default"
                               requestedOwningDomainClass="com.mapics.pm.PurchaseOrder"
                               actionSucceeded="true">
  <MetaData>
    <Descriptor type="label" path="fileDescription"><![CDATA[File description]]></Descriptor>
    <Descriptor type="label" path="global"><![CDATA[Global]]></Descriptor>
    <Descriptor type="label" path="createUser"><![CDATA[Create user]]></Descriptor>

     …

    <Descriptor type="label" path="userFieldText40"><![CDATA[User field - text 40]]></Descriptor>
  </MetaData>
  <DomainEntity domainClass="com.mapics.cas.domain.Attachment">
    <Property path="fileDescription">
      <Value><![CDATA[SpecialDetails.doc]]></Value>
    </Property>
    <Property path="global">
      <Value>0</Value>
    </Property>
    <Property path="createUser">
      <Value />
    </Property>

     …

    <Property path="userFieldText40">
      <Value />
    </Property>
  </DomainEntity>

</QueryAttachmentObjectResponse>
```

FetchNextAttachments tag

A FetchNextAttachments tag is used in conjunction with QueryAttachmentListResponse to display any additional attachment instances that these response tags did not display (because their "maxReturned" values were less than the total number of attachments in their result sets).

Attributes (may appear in any order)

name	an identifier for this FetchNext action. **Must be unique within a given System-Link request document.** Actions are named in the context of a given work area. If a subsequent action is requested with the same name for the same work area, any outstanding results for the old action will be reclaimed.
queryAttachmentName	the query for which the additional results are to be displayed. **The named query will not be found if the QueryAttachmentList action had its retainResults attribute set to "false".**
includeMetaData	(Optional) If set to "true", meta information on the selected properties (e.g. from the PQL "SELECT" clause) will be provided. (At the present time, the meta information for FetchNextAttachments is limited to the property "heading" value.) Defaults to "false".
maxReturned	the maximum number of instances returned.

Notes:

- QueryAttachmentList names are only valid in the scope of their associated SessionHandle and WorkHandle. Because of this, any FetchNextAttachments action must be performed inside of a Request using the same SessionHandle and WorkHandle with which the named query was originally run. If the Request uses a SessionHandle or WorkHandle different from that originally specified, the FetchNextAttachments would not find the remaining result set.
- The metainformation returned by FetchNextAttachments will be identical to that returned by the associated QueryAttachmentList action. Inclusion of metadata on the FetchNextAttachments action by definition will increase the amount of data returned in the FetchNextAttachmentsResponse, and thus the associated network overhead. It is up to the user to determine whether this additional load is outweighed by the cost of storing the original metadata values from the QueryAttachmentListResponse.

Example

This example shows the use of an existing session and work area (resetting the work area maxIdle time to 60 msec), to retrieve up to twenty additional domain objects from a previous query "queryPoAttachments".

```
<SystemLink version='1.0' hostVersion='1.0'>
  <Request sessionHandle='f345' workHandle='6c97d4' maxIdle='60'>
    <FetchNextAttachments name='moreAttachments' queryAttachmentName='queryPoAttachments'
                          maxReturned='20'/>
  </Request>
</SystemLink>
```

FetchNextAttachmentsResponse tag

A FetchNextAttachmentsResponse tag encompasses the result of a FetchNextAttachments request. The fetch result is linked back to the requesting fetch through the name attribute.

Attributes (may appear in any order)

name	the same name submitted to the FetchNextAttachments action tag.
queryAttachmentName	name of the query specified in the FetchNextAttachments request.
actionSucceeded	set to "true" if the fetch action succeeded, otherwise set to "false"
moreResults	if there are more results in the query that have not yet been displayed, this is set to true.

Nested Tags

MetaData	if the request specified that metadata was to be included.
DomainEntity	contains a single attachment instance.
Exception	if the FetchNextAttachments operation could not complete.

Example 1

This example is a possible response to the FetchNext example previously mentioned. Note that no Key sections appear in the DomainEntity elements.

```
<SystemLink version='1.0' hostVersion='1.0'>
  <Response sessionHandle='f345' workHandle='6c97d4'>
   <FetchNextAttachmentsResponse name='moreAttachments'
                                  queryAttachmentName='queryPoAttachments'
                                  actionSucceeded='true' moreResults='false'>
    <DomainEntity domainClass="com.mapics.cas.domain.Attachment">
      <Property path="relatedMediaFile.fileDescription">
        <Value><![CDATA[RepComments01.doc]]></Value>
      </Property>
      <Property path="objectType">
        <Value><![CDATA[PURCHASE ORDER]]> </Value>
      </Property>
      <Property path="objectId">
        <Value><![CDATA[PAA]]></Value>
      </Property>
      <Property path="relatedMediaFile.view">
        <Value>1</Value>
      </Property>
      <Property path="relatedMediaFile.open">
        <Value>0</Value>
      </Property>
      <Property path="relatedMediaFile.print">
        <Value>0</Value>
      </Property>
      <Property path="global">
        <Value>0</Value>
      </Property>
    </DomainEntity>
    <DomainEntity domainClass="com.mapics.cas.domain.Attachment">

    …
```

```
      </DomainEntity>
    </FetchNextAttachmentsResponse>
  </Response>
</SystemLink>
```

Example 2

This is an example response when a FetchNextAttachments action is used to request additional domain objects from a query that has no more elements to be displayed in its result set.

```
<SystemLink version='1.0' hostVersion='1.0'>
  <Response sessionHandle='f345' workHandle='6c97d4'>
    <FetchNextAttachmentsResponse name='anotherFetch'
                                  queryAttachmentName='queryAnotherList'
                                  actionSucceeded='false' moreResults='false'>
    <Exception name='com.pjx.eScript.RequestDataException'>
     <Text><![CDATA[Named query not found]]></Text>
     <Message type='error'>
      <Text><![CDATA[Named query not found.]]></Text>
      <DetailedText><![CDATA[The named query is not found or has no further elements to
enumerate.]]></DetailedText>
     </Message>
     <Message type='information'>
      <Text><![CDATA[Check your XML]]></Text>
      <DetailedText><![CDATA[Please check your XML to ensure that the named query exists and has further
elements.]]></DetailedText>
     </Message>
    </Exception>
   </FetchNextAttachmentsResponse>
  </Response>
</SystemLink>
```

Create and Copy

CreateAttachment tag

A CreateAttachment tag is used to create a new instance of an attachment (link between a business object class or instance and a media file). If the optional SourceAttachment nested tag is specified, then this action becomes a Copy.

Note that a media file corresponding to the specified MediaFilePath may or may not exist. If it does not exist, then a new media file will be created using the associated parameters specified in the CreateAttachment action. If the media file already exists, then values for the same media-file-associated parameters will be taken from the media file (and cannot be overridden in the CreateAttachment property list). Only the attachment "user fields" can be specified regardless of whether the media file exists or not.
Attachment copy restrictions:
- If a "specific → specific" copy is made, the source attachment and destination attachment must not be associated with the same business object instance.
- If a "global → global" copy is made, the source attachment and destination attachment must not have the same associated business object class.

Note that there is no requirement that source and destination attachments need to both be specific or both be global.

Attributes (may appear in any order)

name	an identifier for this Create action. **Must be unique within a given System-Link request document.** Actions are named in the context of a given work area. If a subsequent action is requested with the same name for the same work area, any outstanding results for the old action will be reclaimed.
owningDomainClass	(Optional) the fully-qualified domain class to which the newly-created attachment will belong. Either an owningDomainClass or an owningClassMnemonic must be provided.
owningClassMnemonic	(Optional) the class identifier corresponding to the domain class to which the newly-created attachment will belong. Either an owningDomainClass or an owningClassMnemonic must be provided.
retainResult	(Optional) If set to "true", the resulting "named object" (with name set to the name of this Create name identifier) will be retained for use in subsequent "value references". Default is "false".

Nested Tags (in the order that they must be specified)

LogicalTransaction	(Optional) If specified, the maintenance action is performed in the scope of the logical transaction given.
ReasonCode	(Optional) If specified, the maintenance action is performed with the reason code given.
ApplyTemplate	(Optional) If specified, the named Create or Copy (as applicable) template will be used. **Note: unlike the custom definition key tags in query operations (e.g. HeaderDfnKey), no default Template will be used if ApplyTemplate is not specified.**
MediaFilePath	The absolute path of the media file associated with the attachment.
SourceAttachment	(Optional) If specified, this action becomes a Copy action (i.e. the values in maintainable fields in the named object will be copied to the new object). **The SourceAttachment cannot be specific to the same owning domain class instance, or global to the same owning domain class.**
OwningObjectKey	(Optional) for a specific attachment (one belonging to a single instance of a given domain class), contains the key properties and values for the owning domain object. **Must be omitted for a global attachment.**
DomainEntity	specifies the properties for the new object

Notes:

- Unlike the maintenance actions for "standard" business objects, attachment maintenance actions do not support a Key section in their DomainEntity subtags. All key property values are derived.
- Most maintenance actions (Create / Copy, Update, and Delete) are "auto-commit": each action runs and commits in its own separate transaction. The exceptions to this are those actions that are implicitly grouped (i.e. the associated business objects have the related operations performed on the UJOB, e.g. creation of a ManufacturingOrder) or explicitly grouped (i.e. appear after a StartTransactionGroup action in the request document).

Example 1

This example fragment shows an attachment specific to purchase order "PAA" being created, using the reason code "NEWATT". For purposes of example, assume that the media file "Schematic01.cad" does not exist. As a result, media-file-related fields (non-"user fields", e.g. "open", "view", "print") can be specified (and will be used to create the media file). Since no SourceAttachment appears, this is not a Copy action.

<**CreateAttachment name**='createAttachment_Attachment'

```
                    owningDomainClass='com.mapics.pm.PurchaseOrder'>
    <ReasonCode><![CDATA[NEWATT]]></ReasonCode>

    <MediaFilePath><![CDATA[C:\OrderInfo\Schematic01.cad]]></MediaFilePath>

    <OwningObjectKey>
     <Property path='order'>
       <Value><![CDATA[PAA]]></Value>
     </Property>
    </OwningObjectKey>

    <DomainEntity>
     <Property path='print'>
       <Value><![CDATA[0]]></Value>
     </Property>
     <Property path='userFieldDate3'>
       <Value><![CDATA[0]]></Value>
     </Property>
     <Property path='open'>
       <Value><![CDATA[0]]></Value>
     </Property>
     <Property path='userFieldSwitchD'>
       <Value><![CDATA[0]]></Value>
     </Property>
     <Property path='view'>
       <Value><![CDATA[1]]></Value>
     </Property>

        …

    </DomainEntity>
</CreateAttachment>
```

Example 2

This example shows a copy action being performed. Note in particular that the source attachment is specific to
purchase order "PAA", whereas the new attachment is global to all Vendors (as no OwningObjectKey is specified).
Since the media file exists (as this is a copy), only the user fields may appear in the DomainEntity section.

```
<CreateAttachment name='createAttachment_Attachment'
                    owningDomainClass='com.mapics.pm.Vendor'>

    <MediaFilePath><![CDATA[C:\OrderInfo\Schematic01.cad]]></MediaFilePath>

    <SourceAttachment owningDomainClass='com.mapics.pm.PurchaseOrder'>
      <OwningObjectKey>
       <Property path='order'>
         <Value><![CDATA[PAA]]></Value>
       </Property>
      </OwningObjectKey>
    </SourceAttachment>

    <DomainEntity>
     <Property path='userFieldDate3'>
       <Value><![CDATA[0]]></Value>
     </Property>
     <Property path='userFieldSwitchD'>
       <Value><![CDATA[0]]></Value>
     </Property>
```

```
        …

        </DomainEntity>
</CreateAttachment>
```

CreateAttachmentResponse tag

A CreateAttachmentResponse tag encompasses the result of a CreateAttachment request. The CreateAttachmentResponse is linked back to its corresponding CreateAttachment action via the name attribute. Unlike the "standard" CreateResponse, no Key subtags will appear in the CreateAttachment response. As a result, a successful Create / Copy will have no subtags, whereas a failed Create / Copy will contain an exception.

Attributes (may appear in any order)

name	the same name submitted to the CreateAttachment action tag.
actionSucceeded	set to "true" if the create action succeeded, otherwise set to "false"

Nested Tags

Exception	(one or more) if the Create or Copy operation could not complete.

Example

This example corresponds to a successful completion of Create Example 1:

  <**CreateAttachmentResponse name**='createAttachment_Attachment' **actionSucceeded**='true'/>

UpdateAttachment

UpdateAttachment tag

An UpdateAttachment tag is used to perform an UpdateAttachment operation on an existing attachment. As the other fields are tied to the related media file, only "user fields" may be changed via an UpdateAttachment action.

Attributes (may appear in any order)

name	an identifier for this Update action. **Must be unique within a given System-Link request document.** Actions are named in the context of a given work area. If a subsequent action is requested with the same name for the same work area, any outstanding results for the old action will be reclaimed.
owningDomainClass	(Optional) the fully-qualified domain class to which the attachment belongs. Either an owningDomainClass or an owningClassMnemonic must be provided.
owningClassMnemonic	(Optional) the class identifier corresponding to the domain class to which the attachment belongs. Either an owningDomainClass or an owningClassMnemonic must be provided.
retainResult	(Optional) If set to "true", the resulting "named object" (with name set to the name of this Update name identifier) will be retained for use in subsequent "value references". Default is "false".

Nested Tags (in the order that they must be specified)

LogicalTransaction	(Optional) If specified, the maintenance action is performed in the scope of the logical transaction given.
ReasonCode	(Optional) If specified, the maintenance action is performed with the reason code given.

ApplyTemplate	(Optional) If specified, the named Update template will be used. **Note: unlike the custom definition key tags in query operations (e.g. HeaderDfnKey), no default Template will be used if ApplyTemplate is not specified.**
MediaFilePath	The absolute path of the media file associated with the attachment.
OwningObjectKey	(Optional) for a specific attachment (one belonging to a single instance of a given domain class), contains the key properties and values for the owning domain object. **Must be omitted for a global attachment.**
DomainEntity	specifies the properties to be updated

Notes:

- Unlike the maintenance actions for "standard" business objects, attachment maintenance actions do not support a Key section in their DomainEntity subtags. All key property values are derived.
- Most maintenance actions (Create / Copy, Update, and Delete) are "auto-commit": each action runs and commits in its own separate transaction. The exceptions to this are those actions that are implicitly grouped (i.e. the associated business objects have the related operations performed on the UJOB, e.g. creation of a ManufacturingOrder) or explicitly grouped (i.e. appear after a StartTransactionGroup action in the request document).

Example

This example shows some user field values being updated on the specific attachment for purchase order "PAAA", with associated media file "Schematic01.cad".

```
<UpdateAttachment name='updateAttachment_Attachment'
                  owningDomainClass='com.mapics.pm.PurchaseOrder'>

    <MediaFilePath><![CDATA[C:\OrderInfo\Schematic01.cad]]></MediaFilePath>

    <OwningObjectKey>
      <Property path='order'>
        <Value><![CDATA[PAAA]]></Value>
      </Property>
    </OwningObjectKey>

    <DomainEntity>
      <Property path='userFieldCodeC'>
        <Value><![CDATA[F]]></Value>
      </Property>

        …

      <Property path='userFieldSwitchB'>
        <Value><![CDATA[0]]></Value>
      </Property>
    </DomainEntity>
</UpdateAttachment>
```

UpdateAttachmentResponse tag

An UpdateAttachmentResponse tag encompasses the result of an UpdateAttachment request. The UpdateAttachmentResponse is linked back to its corresponding UpdateAttachment action via the name attribute. If the UpdateAttachment action fails, the UpdateAttachmentResponse will contain an exception detailing the reason for failure.

Attributes (may appear in any order)

name	the same name submitted to the UpdateAttachment action tag.
actionSucceeded	set to "true" if the update action succeeded, otherwise set to "false"

Nested Tags

Exception	(one or more) if the UpdateAttachment operation could not complete.

Example

This example shows a possible response if the UpdateAttachment example fails:
```
<System-Link>
  <Request sessionHandle='f345' workHandle='6c97d4'>
   <UpdateAttachmentResponse name='updateAttachment_Attachment' actionSucceeded='false'>
      <Exception name="com.pjx.eScript.RequestDataException">
        <Text><![CDATA[Attachment MediaFile not found]]></Text>
        <Message type="error">
           <Text>
              <![CDATA[The specified MediaFile 'C:\OrderInfo\Schematic01.cad' could not be found]]>
           </Text>
        </Message>
      </Exception>
   </UpdateAttachmentResponse>
  </Request>
</System-Link>
```

DeleteAttachment

DeleteAttachment tag

A DeleteAttachment tag is used to perform a deletion of an existing attachment. Note: the deletion of an attachment does not delete its associated media file, even if that media file were created via a prior CreateAttachment action.

Attributes (may appear in any order)

name	an identifier for this DeleteAttachment action. **Must be unique within a given System-Link request document.** Actions are named in the context of a given work area. If a subsequent action is requested with the same name for the same work area, any outstanding results for the old action will be reclaimed.
owningDomainClass	(Optional) the fully-qualified domain class to which the attachment belongs. Either an owningDomainClass or an owningClassMnemonic must be provided.
owningClassMnemonic	(Optional) the class identifier corresponding to the domain class to which the attachment belongs. Either an owningDomainClass or an owningClassMnemonic must be provided.

Nested Tags (in the order that they must be specified)

LogicalTransaction	(Optional) If specified, the maintenance action is performed in the scope of the logical transaction given.
ReasonCode	(Optional) If specified, the maintenance action is performed with the reason code given.
MediaFilePath	The absolute path of the media file associated with the attachment.

OwningObjectKey	(Optional) for a specific attachment (one belonging to a single instance of a given domain class), contains the key properties and values for the owning domain object. **Must be omitted for a global attachment.**

Notes:
- Most maintenance actions (Create / Copy, Update, and Delete) are "auto-commit": each action runs and commits in its own separate transaction. The exceptions to this are those actions that are implicitly grouped (i.e. the associated business objects have the related operations performed on the UJOB, e.g. creation of a ManufacturingOrder) or explicitly grouped (i.e. appear after a StartTransactionGroup action in the request document).
- If the Delete action is performed against an object that is currently stored as a "named object", that named object will no longer be valid.

Example

This example fragment shows the deletion of an attachment specific to purchase order "PAAA", with associated media file "Schematic01.cad".

```
<DeleteAttachment name='deleteAttachment'
                    owningDomainClass='com.mapics.pm.PurchaseOrder'>

    <MediaFilePath><![CDATA[C:\OrderInfo\Schematic01.cad]]></MediaFilePath>

    <OwningObjectKey>
      <Property path='order'>
        <Value><![CDATA[PAAA]]></Value>
      </Property>
    </OwningObjectKey>

</DeleteAttachment>
```

DeleteAttachmentResponse tag

A DeleteAttachmentResponse tag encompasses the result of a DeleteAttachment request. The DeleteAttachmentResponse is linked back to its corresponding DeleteAttachment action via the name attribute. If the DeleteAttachment action fails, the DeleteAttachmentResponse will contain an exception detailing the reason for failure.

Attributes (may appear in any order)

name	the same name submitted to the DeleteAttachment action tag.
actionSucceeded	set to "true" if the delete action succeeded, otherwise set to "false"

Nested Tags

Exception	(one or more) if the DeleteAttachment operation could not complete.

Example

This example shows a possible response if the DeleteAttachment example fails:
```
<System-Link>
  <Request sessionHandle='f345' workHandle='6c97d4'>
    <DeleteAtttachmentResponse name='deleteAttachment' actionSucceeded='false'>
      <Exception name='com.pjx.eScript.SecurityException'>
        <Text><![CDATA[Security violation during delete]]></Text>
        <Message type='error'>
          <Text><![CDATA[User 'john doe' not authorized to delete 'OLDB']]></Text>
        </Message>
```

```
        </Exception>
      </DeleteAttachmentResponse>
    </Request>
</System-Link>
```

MediaFilePath

MediaFilePath tag

A MediaFilePath tag is used to identify the media file associated with the attachment object action.

Attributes

(none)

Nested Tags

a String value

See the QueryAttributeObject and attribute maintenance tag sections for example usage.

OwningObjectKey

OwningObjectKey tag

An OwningObjectKey tag is used to indicate the domain object instance to which an attachment is associated. An OwningObjectKey is not used when referring only to global attachments.

Attributes

(none)

Nested Tags

| Property | each property tag holds a key value of the object instance. |

See the attribute inquiry and maintenance tag sections for example usage.

SourceAttachment

SourceAttachment tag

A SourceAttachment tag is used within a CreateAttachment action tag to perform a copy.

Attachment copy restrictions:
- If a "specific → specific" copy is made, the source attachment and destination attachment must not be associated with the same business object instance.
- If a "global → global" copy is made, the source attachment and destination attachment must not have the same associated business object class.

Note that there is no requirement that source and destination attachments need to both be specific or both be global.

Attributes

owningDomainClass	(Optional) the fully-qualified domain class to which the source attachment belongs. <u>Either an owningDomainClass or an owningClassMnemonic must be provided.</u>
owningClassMnemonic	(Optional) the class identifier corresponding to the domain class to which the source attachment belongs. <u>Either an owningDomainClass or an owningClassMnemonic must be provided.</u>

Nested Tags

OwningObjectKey	(Optional) for a specific attachment (one belonging to a single instance of a given domain class), contains the key properties and values for the owning domain object. **Must be omitted for a global attachment.**

See the CreateAttachment tag section for example usage.

Document Processing Commands

Document processing commands affect the way that the actions in a given document are actually processed by System-Link. This includes both conditional entities, as well as transactional entities.

IfCondition, EndIfCondition

IfCondition, EndIfCondition tag

An IfCondition-EndIfCondition tag pair is used to conditionally run one or more actions. These conditional blocks may be used to run actions based upon the presence of earlier business object(s), run maintenance actions only if earlier maintenance actions succeeded, etc..

Each IfCondition action may contain one or more Condition subentities: the test of each must pass (i.e. logical AND) in order for the actions between the IfCondition-EndIfCondition pair to be run.

Attribute (IfCondition only)

name	an identifier for this IfCondition action. **Must be unique within a given System-Link request document.** Actions are named in the context of a given work area. If a subsequent action is requested with the same name for the same work area, any outstanding results for the old action will be reclaimed.

Nested Tag (IfCondition only)

Condition	Indicates a single condition that must pass in order for the conditional block to be executed.

Notes:
- IfCondition-EndIfCondition blocks may be nested. EndIfConditions are not "named": each EndIfCondition entity will close the current inner-most condition block. If an outer IfCondition does not succeed (i.e. one or more of its Condition subentities does not pass), all nested IfCondition-EndIfCondition blocks (along with any other actions inside the outer block) will be skipped.
- Condition blocks are only valid within a single System-Link request document. All IfCondition actions must have corresponding EndIfCondition actions within the same request document.

Example

This example fragment shows the creation of a PurchaseOrderItem being conditioned on the presence of the associated PurchaseOrder. Note the use of the "existenceCheck" attribute for the QueryObject action. This will result in an "actionSucceeded = 'true'" being displayed in the QueryObjectResponse even if the PurchaseOrder is not found. (Internally, an ObjectNotFound is always flagged as an "action failure". See the QueryObject section for details.):

```
<Request sessionHandle='*current' workHandle='*new' broker='EJB' maxIdle='1000'>

    <QueryObject name='queryObject_MyPO' domainClass='com.mapics.pm.PurchaseOrder'
                existenceCheck='true'>
        <Pql>
            <![CDATA[SELECT order WHERE order = 'P000236']]>
        </Pql>
    </QueryObject>
    <IfCondition name='myPOPresent'>
        <Condition actionName=' queryObject_MyPO ' actionStatus='succeeded'/>
    </IfCondition>
```

```
    <Create name='createObject_PurchaseOrderItem' domainClass='com.mapics.pm.PoItem'>
      <MaintenanceOptions>
        <Option optionName='includeVendorItemComments'>
          <Value><![CDATA[1]]></Value>
        </Option>
        <Option optionName='includeItemReceivingOperations'>
          <Value><![CDATA[0]]></Value>
        </Option>
      </MaintenanceOptions>
      <DomainEntity>
        <Key>
          <Property path='order'>
            <Value><![CDATA[P000236]]></Value>
          </Property>
        </Key>
        <Property path='blanketItem'>
          <Value><![CDATA[0]]></Value>
        </Property>
        <Property path='warehouse'>
          <Value><![CDATA[MPA]]></Value>
        </Property>

        ...
    </Create>

    <EndIfCondition />

  </Request>
```

IfConditionResponse, EndIfConditionResponse tag

IfConditionResponse and EndIfConditionResponse tags encompass the result of an IfCondition-EndIfCondition conditional block. If the IfCondition action does not succeed (i.e. one or more of its Conditions does not pass), no response tags will appear between that IfConditionResponse and the matching EndIfConditionResponse.

Attributes (IfConditionResponse only - may appear in any order)

name	the same name submitted to the IfCondition action tag.
conditionPassed	set to "true" if the IfCondition's Condition(s) passed, otherwise set to "false"

Attribute (EndIfConditionResponse only)

ifName	the same name submitted to the corresponding IfCondition action tag.

Nested Tags

Exception	if the IfCondition or EndIfCondition action could not be performed.

Example 1

This example is the fragment of a possible response to the IfCondition example previously mentioned. In this case, the named PurchaseOrder does not exist. (Note, however, that the actionSucceeded attribute on the QueryObject action is still set to "true". This is due to the existenceCheck attribute setting on the QueryObject action.)

```
<Response sessionHandle='166aa18:10f97141af5:-7fe0' workHandle='166aa18:10f97141af5:-7fdf'>
  <QueryObjectResponse name='queryObject_MyPO' requestedDomainClass='com.mapics.pm.PurchaseOrder'
                   actionSucceeded='true'>
    <Exception name='com.pjx.eScript.ObjectNotFoundException'>
```

```
          <Text><![CDATA[Object does not exist]]></Text>
          <Message type='error'>
            <Text><![CDATA[The named object was not found.]]></Text>
          </Message>
        </Exception>
      </QueryObjectResponse>
      <IfConditionResponse name='myPOPresent' conditionPassed='false'>
      </IfConditionResponse>
      <EndIfConditionResponse ifName='myPOPresent'>
      </EndIfConditionResponse>
    </Response>
```

Example 2

This example is the fragment of a possible response to the IfCondition example previously mentioned. In this case, the named PurchaseOrder does exist. The conditional block "myPoPresent" is therefore executed.

```
  <Response sessionHandle='166aa18:10f97141af5:-7fe0' workHandle='166aa18:10f97141af5:-7fdf'>
    <QueryObjectResponse name='queryObject_MyPO' requestedDomainClass='com.mapics.pm.PurchaseOrder'
                    actionSucceeded='true'>
      <DomainEntity domainClass='com.mapics.pm.PurchaseOrder'>
        <Key>
          <Property path='order'>
            <Value><![CDATA[P000236]]></Value>
          </Property>
        </Key>
        <Property path='order'>
          <Value><![CDATA[P000236]]></Value>
        </Property>
      </DomainEntity>
    </QueryObjectResponse>
    <IfConditionResponse name='myPOPresent' conditionPassed='true'>
    </IfConditionResponse>

    <CreateResponse name='createObject_PurchaseOrderItem' domainClass='com.mapics.pm.PoItem'
                    actionSucceeded='true'>
      <Key>
        <Property path='order'>
          <Value><![CDATA[P000236]]></Value>
        </Property>
      </Key>
    </CreateResponse >

    <EndIfConditionResponse ifName='myPOPresent'>
    </EndIfConditionResponse>
  </Response>
```

Condition

Condition tag

The Condition tag is used inside of an IfCondition action to specify a single condition which is evaluated to determine whether or not the conditional block should be executed.

Attributes (may appear in any order)

actionName	name of the action whose status will be evaluated

actionStatus	status which, if matching the status of the named action ("actionName"), will allow the Condition to pass. May be one of the following: "succeeded", "failed", "moreResults", "noMoreResults".

Notes:

- "moreResults" and "noMoreResults" are associated with the current state of the result set associated with the actionName, rather than the state of the result set after the initial Query action. As an example, Conditions with status "moreResults" and "noMoreResults" will reflect whether a Fetch action on the associated named result set can or cannot be run, respectively.
- when using actionStatus of "moreResults" and "noMoreResults", make sure to set the actionName to equal that matching the initial Query action name, not any subsequent Fetch actions.

For an example of Condition, see the examples under the IfCondition, EndIfCondition tag.

StartTransactionGroup

StartTransactionGroup tag

A StartTransactionGroup tag is used to group all subsequent maintenance actions in a request document as a single transaction. Normally, most maintenance actions are "auto-commit": each maintenance action commits prior to the next action being run.

Attributes: none

Nested Tags: none

Notes:

- Transaction grouping only affects maintenance actions. If an inquiry action depends upon the results of one or more grouped transactions, it must be run in a subsequent request document.
- A StartTransactionGroup action only affects subsequent maintenance actions up to the end of the containing request document. Maintenance actions prior to the StartTransactionGroup action, as well as maintenance actions in subsequent request documents (regardless of sessionHandle and workHandle) are not affected.
- Some objects perform some server validation operations only at time of commit. If these operations fail, the maintenance actions will be flagged as "succeeded", but one or more exceptions will appear at the close of the document.
- There are a handful of business object classes that always implicitly result in transaction grouping. These classes are the ones that process on the UJOB.
- Only one transaction grouping can be present per request document. Any StartTransactionGroup action performed while a transaction grouping is active will have no effect.
- Validations for maintenance actions within a transaction grouping that depend on the result of a prior maintenance action in the group will not pass except for header/detail create actions. In this case, a "detail" create action which follows the create action for the associated "header" will pass the existence check it performs to ensure the header exists.

Example

This example fragment shows a Vendor being changed, followed by a PurchaseOrder and PurchaseOrderItems being created and a second Vendor being changed. Note that the update action "updateObject_Vendor1" is committed immediately, but the second update action "updateObject_Vendor2" is grouped with the PurchaseOrder and PurchaseOrderItem create actions.

```
<Request sessionHandle='*current' workHandle='*new' broker='EJB' maxIdle='1000'>
```

```
<Update name='updateObject_Vendor1' domainClass='com.mapics.pm.Vendor'>
 ...
</Update>


<StartTransactionGroup />

<Create name='createObject_PurchaseOrder' domainClass='com.mapics.pm.PurchaseOrder'>
 ...
</Create>

<Create name='createObject_PurchaseOrderItem1' domainClass='com.mapics.pm.PoItem'>
 ...
</Create>
<Create name='createObject_PurchaseOrderItem2' domainClass='com.mapics.pm.PoItem'>
 ...
</Create>
<Create name='createObject_PurchaseOrderItem3' domainClass='com.mapics.pm.PoItem'>
 ...
</Create>

<Update name='updateObject_Vendor2' domainClass='com.mapics.pm.Vendor'>
 ...
</Update>

</Request>
```

StartTransactionGroupResponse tag

StartTransactionGroupResponse tags are only generated if exceptions occur when starting the grouping transaction.

Attributes: none

Nested Tags

Exception	if the StartTransactionGroup action could not be performed.

Common Subtags

QueryDfn

QueryDfn tag

The QueryDfn tag is used in list query actions (i.e. QueryList, QueryAttachmentList, and QueryRelationship) to allow the desired list to be specified using a header ("view"), subset, sort, and prompted subset values.   (The alternate is to specify the list using the Pql tag.)

Attributes
(none)

Nested Tags (in the order that they must be specified)

HeaderDfnKey	(Optional) specifies the header definition to be used. If the HeaderDfnKey tag is not specified, the default is used from the user's preferences.
SubsetDfnKey	(Optional) specifies the subset definition to be used. If the SubsetDfnKey tag is not specified, the default is used from the user's preferences.
PromptedSubsetValues	(Optional) if the specified subset has prompted properties, values for each property must be specified here
SortDfnKey	(Optional) specifies the sort definition to be used. If the SortDfnKey tag is not specified, the default is used from the user's preferences.

Note:

- Not all domain classes support headers, subsets, or sorts. For these, the PQL tag must be used. Use the Power-Link customization panes, or see the ListCustomDfns tag to determine which definitions are present for a given domain class.

For an example of QueryDfn, see Example 2 under the QueryList tag. See also the ListCustomDfns tag.

Constraint

Constraint tag

The Constraint tag is used in a QueryList or QueryAttachmentList action to limit the list returned to be those objects related to a specific domain object by a one-to-many relationship. (An example of this would be a QueryList requesting a list of all Warehouses for a given Item.)

Attributes (may appear in any order)

domainClass	(Optional) the domain class of the from object (specified by the Key). Either a domainClass or a classMnemonic must be provided.
classMnemonic	(Optional)) the class identifier corresponding to the domain class of the from object (specified by the Key). Either a domainClass or a classMnemonic must be provided.
relationshipName	name of the relationship that will be traversed to constrain the QueryList result set

Nested Tags

Key	holds the properties that form a unique key to the from object.

For an example of Constraint, see Example 2 under the QueryList tag.

PromptedSubsetValues

PromptedSubsetValues tag

If a QueryDfn specifies a Subset definition that has values that are provided at time of use (i.e. are "prompted"), then these are specified in SystemLink by the PromptedSubsetValues tag. (In addition to the using the Power-Link customization interface, one can determine the prompted properties via the ListCustomDfns action.)

Attributes

(none)

Nested Tags

CriteriaSpec	(One or more) used to specify the value for the prompted property

For an example of PromptedSubsetValues, see Example 2 under the QueryList tag. See also the ListCustomDfns tag.

CriteriaSpec

CriteriaSpec tag

The CriteriaSpec tag is used to provide a value to a single prompted property inside of a PromptedSubsetValues tag.

Attributes (may appear in any order)

path	(Optional) the property path for the property. <u>Either a path or an alias must be specified.</u>
alias	(Optional) the alias (field name) for the property. <u>Either a path or an alias must be specified.</u>

Nested Tags

Operator	(Optional) if more than one prompted property use the same path, the Operator tag must be given to help distinguish to which prompted property this CriteriaSpec belongs.
Value	used to specify the value for the property
ValueRef	use the same value for this property as that found on the given property for the "named object" (returned by QueryObject, Create/Copy, or Update).

Notes:

- If more than one prompted property uses the same path and the same Operator, then the values provided will be applied in the order that they are specified.
- Values provided for dates via the Value tag must be absolute (in YYYYMMDD format). Relative values such as "today + 2 months" are not supported.

For an example of CriteriaSpec, see Example 2 under the QueryList tag. See also the ListCustomDfns tag.

Operator

Operator tag

An Operator tag is used inside of a CriteriaSpec to further refine the prompted property assignment. If more than one prompted property on a given subset shares the same path, then this tag is required in order to help distinguish between them.

Attributes

(none)

Nested Tags

a String value corresponding to the Operator type returned for the prompted subset by the ListCustomDfns action.

For an example of Operator, see Example 2 under the QueryList tag. See also the ListCustomDfns tag.

MetaData

MetaData tag

The MetaData tag is used to provide metadata for the properties returned for inquiry commands.

For MetaData associated with text objects, the TextDescriptor subtag will appear. Otherwise, the Descriptor subtag will appear.

Attributes

(none)

Nested Tags (either Descriptor or TextDescriptor subtags will appear, but never both)

Descriptor	one or more descriptor tags will be present (one for each property in the PQL SELECT clause or QueryDfn / CardFileDfn).
TextDescriptor	one or more text descriptor tags will be present (one containing the text card label, and one for each text object type appearing in the response).
Exception	if the MetaData request could not complete.

Example 1

This example is a snippet from the QueryObjectResponse section:

```
<MetaData>
 <Descriptor type='label' path='order'>
  <![CDATA[Order]]>
 </Descriptor>
 <Descriptor type='label' path='relatedBuyer.buyerName'>
  <![CDATA[Buyer Name (related)]]>
 </Descriptor>
 <Descriptor type='label' path='relatedVendor.vendorName1'>
  <![CDATA[Vendor Name (related)]]>
 </Descriptor>
</MetaData>
```

Example 2

This example is a snippet from the QueryTextResponse section:

```
<MetaData>
  <TextDescriptor type="label">
    <Value><![CDATA[Advanced Comments]]></Value>
  </TextDescriptor>
  <TextDescriptor type="label">
    <Type><![CDATA[AA]]></Type>
    <Value><![CDATA[Engineering]]></Value>
  </TextDescriptor>
  <TextDescriptor type="label">
    <Type><![CDATA[BB]]></Type>
    <Value><![CDATA[Purchasing]]></Value>
  </TextDescriptor>
  <TextDescriptor type="label">
    <Type><![CDATA[CC]]></Type>
    <Value><![CDATA[Miscellaneous]]></Value>
  </TextDescriptor>
</MetaData>
```

Descriptor

Descriptor tag

Each Descriptor tag is used to describe the metadata for a single property (otherwise known as a "metaproperty"). These descriptors contain the "path" of each property, the descriptor "type", and the descriptor "value" for the given Domain Object class. (For the current version of System-Link, QueryList and QueryRelationship requests will only return values for the "heading" metaproperty, and QueryObject requests will only return the values for the "label" metaproperty.

Attributes (may appear in any order)

type	the "type" of metaproperty (currently limited to "heading" and "label", as described above).
path	the path of the associated property.

Nested Tags

A String corresponding to the value of the indicated metaproperty type for the given property.

See the MetaData tag section for an example.

TextDescriptor

TextDescriptor tag

Each TextDescriptor tag is used to describe the metadata for a single text object or text object type. For a simple ("non-typed") text object, a single TextDescriptor will appear, with the value of the corresponding relationship description. For an advanced ("typed") text object, subsequent TextDescriptors will appear showing the type alias (description) for each type involved in the action.

Attributes

type	the "type" of metaproperty (currently limited to "label").

Nested Tags

Type	used to indicate the text object type
Value	used to indicate the text object label value

See the QueryTextResponse and FetchNextTextResponse tag sections for an example.

TypedTextValue

TypedTextValue tag

The TypedTextValue tag appears in a QueryTextResponse and a FetchNextTextResponse to display advanced ("typed") text data. Each text type in the response will have an associated TypedTextValue.

Attributes
(none)

Nested Tags

Type	used to indicate the text object type
Value	used to indicate the text object value
Exception	if the text object request failed for the given type

For an example of TypedTextValue, see Example 2 under the QueryTextResponse tag.

Pql

Pql tag

A Pql tag is used by inquiry actions to specify the object or list of objects to be returned.

Attributes

usingAliasName s	(Optional) If set to "true", all properties in the PQL must all be specified using aliases (field names). If "false", all properties in the PQL are specified using property paths. Default is "false".

Nested Tags
a String specifying the PQL statement.

Note:

- the PQL statement must use either aliases or property names. Using a mixture of aliases and property names is not supported.

PQL (similar to SQL) is used by the inquiry actions to directly specify the object or list of objects to retrieve. (Detailed discussion of PQL is beyond the scope of this document. The easiest way to generate the base PQL for an inquiry action is via the System-Link request option in Power-Link.)

A PQL statement may have two kinds of "substitution variables", both designated by "&" delimiters:
1. references in a QueryRelationship action to properties in its closest outer QueryObject or QueryRelationship, or

2. "value references": references to the value of properties from "named objects" (obtained via QueryObject, Create, Copy, or Update).

(Note: a value reference must be used under the same sessionHandle and workHandle that was specified when the named object was obtained. Otherwise, the desired object will not be found.)

Substitution variables of the first type are detailed under the QueryRelationship tag. Value reference substitution variables can occur in any PQL statement, and have a syntax similar to that used in the ValueRef maintenance tag:

&(name of object).property&

Please note: in PQL for a QueryRelationship tag, a value reference cannot be made in which the specified named object is created by the QueryRelationship's outer QueryObject call. An example of this unsupported use is:

```
<QueryObject name=myQueryObject … retainResult='true'>
        …
        <QueryRelationship … >
                <Pql><![CDATA[SELECT buyerName
                                WHERE buyer=&(myQueryObject).buyer&]]></Pql>
        <./QueryRelationship>
</QueryObject>
```

Example

This example uses the PQL from the previous QueryObject Example 1, except that the order property has its value specified by a value reference substitution variable. (In this case, the named object was obtained from a Create action named "createPoItem" that appeared earlier in the request.)

```
<SystemLink version='1.0' hostVersion='1.0'>
  <Login userId='john doe' password='12345' maxIdle='300'
      properties='com.mapics.cas.domain.EnvironmentId=25,
                  com.mapics.cas.domain.SystemName=USATLD06.INFOR.COM,
                  com.mapics.cas.user.LanguageId=en' />
  <Request sessionHandle='*current' workHandle='*new' maxIdle='30'>
        …

   <QueryObject name='querySinglePo' domainClass='com.mapics.pm.PurchaseOrder'
        includeMetaData='true' retainResult='false'>
    <Pql>
     <![CDATA[SELECT order, relatedBuyer.buyerName, relatedVendor.vendorName1
        WHERE order=&(createPoItem).order&]]>
    </Pql>
   </QueryObject>
  </Request>
</SystemLink>
```

ColumnStatistics

ColumnStatistics tag

A ColumnStatistics tag is used by inquiry list actions to return any column statistics requested (e.g. COUNT, MIN, MAX). Column statistics are either requested explicitly in a PQL statement, or implicitly as part of a header / view definition.

Attributes
(none)

Nested Tags

Calculation	contains the data for a single column statistic
Exception	if the column statistic request resulted in errors

Example 1

This example fragment shows an explicit request for column statistics data using PQL. (The corresponding request using custom definitions would have this column statistics information contained in the header / view state, and thus would not be reflected in the request itself.)

```
<QueryList name='queryListPurchaseOrder_CalcView' domainClass='com.mapics.pm.PurchaseOrder'
        includeMetaData='true' maxReturned='10'>
    <Pql>
        <![CDATA[SELECT order,relatedVendor.amountLastYear,
            actualSpecialChargesLocalCurrency,actualSpecialCharges,
            actualFreight,relatedVendor.amountYearToDate,
            vendor,vendorName,orderStatus,releaseDate,
            revision,lastRevisionDate,buyer,relatedBuyer.buyerName
            CALCULATE COUNT(order),MAX(relatedVendor.amountLastYear),
            MIN(actualSpecialChargesLocalCurrency),
            AVG(actualSpecialCharges),SUM(actualFreight)
            ORDER BY order]]>
    </Pql>
</QueryList>
```

Example 2

This example fragment shows a possible result of a request containing the fragment shown in Example 1.

```
<QueryListResponse name='queryListPurchaseOrder_MyCalculateView'
                requestedDomainClass='com.mapics.pm.PurchaseOrder' actionSucceeded='true'
                moreResults='true'>
    <MetaData>
        <Descriptor type='heading' path='order'><![CDATA[Order]]></Descriptor>
        …
    </MetaData>
    <ColumnStatistics>
        <Calculation path='order' function='COUNT'>
            <Value>599</Value>
        </Calculation>
        <Calculation path='relatedVendor.amountLastYear' function='MAX'>
            <Value>75.00</Value>
        </Calculation>
        <Calculation path='actualSpecialChargesLocalCurrency' function='MIN'>
            <Value>0.00</Value>
        </Calculation>
        <Calculation path='actualSpecialCharges' function='AVG'>
            <Value>0.00</Value>
        </Calculation>
        <Calculation path='actualFreight' function='SUM'>
            <Value>0.00</Value>
        </Calculation>
    </ColumnStatistics>
    <DomainEntity domainClass='com.mapics.pm.PurchaseOrder'>
        …
    </DomainEntity>
```

Calculation

Calculation tag

A Calculation tag is used to return the results of a single column statistics calculation.

Attributes (may appear in any order)

path	the property path for the property
function	the function used in the calculation. May be one of the following: AVG, COUNT, MAX, MIN, SUM

Nested Tags (either Value or NullValue may appear)

Value	value of the property as a String.
NullValue	specifies that the value of the option is null. (Note: only valid for options that support null values.)
Exception	if the column statistic calculation resulted in errors

Note:

- if a calculation has no value (as opposed to a value of zero or a null value – example: the MIN over a property for an inquiry with a result set containing no records), neither a Value nor a NullValue nested tag will appear.

For an example of TypedTextValue, see Example 2 under the QueryTextResponse tag.

DomainEntity

DomainEntity tag
The DomainEntity tag is used to contain a single Domain Object instance. A DomainEntity will usually have a Key as its first element. Zero or more properties can follow the key.

Note:
- DomainEntity tags only possess attributes when they appear inside a response. DomainEntity tags inside requests do not have attributes.
- If appearing within a Create request for an object in which all key values are system generated, the Key section should be empty ("<Key></Key>") or omitted.
- When appearing within any attachment-related action request or response, the Key section will be omitted.

Attributes

domainClass	(Only present in responses) the class of this DomainEntity

Nested Tags

Key	(Optional for create requests, and not included for attachment-related actions) holds the properties that form a unique key to the object.
Property	each property tag holds a value of the object instance. These properties will include the key properties for inquiry actions, but should not include the key properties for maintenance actions.

See any of the Query Response tag sections for examples

HeaderDfnKey, SubsetDfnKey, SortDfnKey, CardFileDfnKey, ApplyTemplate

HeaderDfnKey, SubsetDfnKey, SortDfnKey, CardFileDfnKey, ApplyTemplate tag
These tags are used to specify the instance of the header, subset, sort, card file, and template definitions to be used. They are grouped here due to common functionality and format.

Attributes (may appear in any order)

objectId	the object Id string uniquely identifying the desired definition. This is obtained via the ListCustomDfns action.
clientClass	(Optional) the business object class for which this definition is used (e.g. "com.mapics.pm.Buyer"). This corresponds to the "domainClass" for QueryObject, Create, etc. Either a clientClass or a classMnemonic must be provided.
classMnemonic	(Optional) the class identifier corresponding to the business object class for which this definition is used (e.g. "BYR"). This corresponds to the "classMnemonic" for QueryObject, Create, etc. Either a domainClass or a classMnemonic must be provided.

Nested Tags
A String giving the name of the definition to use.

All of these definition tags may reference the definition either by objectId or by specifying the definition name. (objectId is preferred, as it guarantees uniqueness. In the case where the XML request could be used across multiple installations, the definition name must be used.) If the definition name is given without the objectId being specified, then the first definition of the proper type and clientClass with a title that matches that name is used. If both objectId and a definition name are given, the objectId is used.

See the QueryList tag section for examples.

HostJobType, HostReportType

HostJobType, HostReportType tag

These tags are used to specify the host process (host job or host report) to be used in a RunHostJob and CreateHostReport subaction. They are grouped here due to common functionality and format.

Attributes

mnemonic	the unique identifier for the host process.

Nested Tags

A String giving the name of the host process.

The HostJobType or HostReportType tag may reference the host process either by mnemonic or by specifying the host process name. (mnemonic is preferred, as it guarantees uniqueness.) If both mnemonic and a host process name are given, the mnemonic is used.

Key

Key tag

A key tag is used to group those properties in a property list that belong to an object's key.

Attributes
(none)

Nested Tags

Property	each property tag holds a value of the object instance.

See the Property tag section for example usage.

Property

Property tag

A Property tag is used to indicate a given Domain Object property, as well as its associated value.

Attributes (may appear in any order)

path	(Optional) the property path for the property. Either a path or an alias must be specified.
alias	(Optional) the alias (field name) for the property. Either a path or an alias must be specified.

Nested Tags (either Value, ValueRef, or NullValue may be specified)

Value	value of the property as a String.
ValueRef	(Only in a System-Link Request document) use the same value for this property as that found on the given property for the "named object" (returned by QueryObject, Create / Copy, or Update).

NullValue	specifies that the value of the option is null. (Note: only valid for options that support null values.)

Note:

- for date Properties, values specified via the Value tag must be absolute (in YYYYMMDD format). Relative values such as "today + 2 months" are not supported.

Example

The example below is a System-Link fragment showing two Property tags: one using the property path and a Value subtag and one using the property alias and a ValueRef subtag. Note that the ValueRef subtag obtains its value from the named object "myBuyerQuery" (in this case apparently obtained using an earlier QueryObject request), from property "buyerName".

<**Property path**='buyerName'>
  <Value><![CDATA[Jones and Company]]></Value>
</**Property**>

<**Property alias**=' bynmbm'>
  <ValueRef><![CDATA[(myBuyerQuery).buyerName]]></ValueRef>
</**Property**>

See the Value, ValueRef, and NullValue tag descriptions for further details.

LogicalTransaction

LogicalTransaction tag

A LogicalTransaction tag is used in a maintenance action to specify the logical transaction under which the action is performed.

Attributes

(none)

Nested Tags

a String value

See the Create tag section for example usage.

ReasonCode

ReasonCode tag

A ReasonCode tag is used in a maintenance action to specify the reason code for the action.

Attributes

(none)

Nested Tags

a String value

See the Create, Update, and Delete tag sections for example usage.

Type

Type tag

A Type tag is used in actions involving advanced ("typed") text objects to indicate associated type.

Attributes

(none)

Nested Tags

a String value

See the QueryText and MaintainText tag sections for example usage.

Value

Value tag

A Value tag indicates the data value (represented by a String) for a given property or text object. Value tags appear as subtags of several different tags:
In requests: Property, CriteriaSpec, TextAction
In responses: Property, QueryTextResponse, FetchNextTextResponse, TextDescriptor, TypedTextValue

Attributes

moreChars	(Under the QueryTextResponse, FetchNextTextResponse, and TypedTextValue tags only) indicates that the associated text object has further characters to be displayed. Default is "false".

Nested Tags
a String value

See the Property tag and QueryTextResponse sections for example usage.

ValueRef

ValueRef tag

A ValueRef tag is used by the Property to indicate that the property's value should be taken from the specified property on the specified "named object". The ValueRef tag is only used in a System-Link Request document where properties are explicitly named (i.e. Create, Copy (including the SourceObject), Update, Key, and CriteriaSpec). This tag will never appear as part of a System-Link response.

Attributes
(none)

Nested Tags
a String value

A ValueRef tag is used to perform an action on one object using the value from another object obtained dynamically via a previous QueryObject, Create / Copy, or Update action. (Note: ValueRef must be used under the same sessionHandle and workHandle that was specified when the named object was obtained. Otherwise, the desired object will not be found.)

The named object (given the same name as the action that was used to obtain it) and property on that object must be specified with the following format:

> (name of object).property

For example, to obtain the value for property "buyerName" on the Buyer object retrieved by previous QueryObject action named "myBuyerQuery", the ValueRef would look like:

> <**ValueRef**><![CDATA[(myBuyerQuery).buyerName]]></**ValueRef**>

Note: If a Delete action is performed against an object that is currently stored as a "named object", that named object will no longer be valid.

NullValue

NullValue tag

A NullValue tag indicates that the value for a given property is null. NullValue tags can appear in both requests and responses.

Attributes

(none)

Nested Tags

(none)

Note: the NullValue tag is only relevant for properties (and databases) that support null values. If null values are not supported for a given property, the NullValue tag will never be produced in the corresponding response. If NullValue
is used in a request for this property, an error message will be returned.

Example

The example below is a System-Link fragment showing a property with a null value.

```
<Property path='myExtensionProperty'>
  <NullValue />
</Property >
```

Exception

Exception tag

An Exception tag is used to communicate an error that occurred while processing a request action. An exception contains a text description and zero or more messages.

System-Link will return ten possible exception classes as values for the "name" attribute. They are categorized as:
- exceptions caused by user input that require user correction (e.g. specifying a bad property name),
- exceptions caused by system configuration that requires user correction (e.g. providing a bad password), and
- exceptions that are "internal" to XA and / or System-Link that require the attention of a system administrator.

Exception classes

Internal Exception

"com.pjx.eScript.InternalException" — Flags exceptions for error conditions that require the attention of a system administrator.
Each exception will be accompanied by a log entry, with the SessionHandle and WorkHandle attached to both for cross-reference

RequestFormat Exception

"com.pjx.eScript.RequestFormatException" — Flags a Request that has syntactic or semantic errors. Instances of RequestFormatExceptions include request documents containing undefined XML tags and misspelling of tag attributes (e.g."maxIdle").

RequestData Exception

"com.pjx.eScript.RequestDataException" — Flags a Request that has invalid user input data.
Instances of RequestDataExceptions include the following:
- two named actions are given the same name in a single document
- invalid value for maxIdle or maxReturned
- invalid FetchNext query name
- invalid user-specified Domain Class or Property
- errors in the PQL statement

ObjectNotFound Exception

"com.pjx.eScript.ObjectNotFoundException" — Indicates that the specified object does not exist.

ObjectAlreadyExists Exception

"com.pjx.eScript.ObjectAlreadyExistsException" — (Create and Copy) Indicates that an object of the same domain class and key already exists.

WrongNumberObjects Exception

"com.pjx.eScript.WrongNumberObjectsException" — Indicates that a QueryObject request has been given a PQL statement that returns more than one object.

Security Exception

"com.pjx.eScript.SecurityException"

Flags all violations of Security (field, object, login, logout) Instances of SecurityExceptions include session and workspace non-existence (either due to timeout or to invalid handles).

ConcurrentUse Exception

"com.pjx.eScript.ConcurrentUseException"

(maintenance-only) Indicates that another user has the given object locked.

Transaction Exception

"com.pjx.eScript.TransactionException"

Indicates that errors occurred during transaction operations.

Throwable Exception

"java.lang.Throwable"

A generic exception from the servlet. The most typical Throwable contains the text "Unable to communicate with System-Link server. Please verify that the environment specified was correct and the XA Java Server UJOB is started and in an active state." This means that the servlet was unable to communicate with the XA Java Server. You may need to configure the Primary TCP Port on the servlet or start the XA Java Server.

Attributes

name	the Java class name of the associated exception instance.

Nested Tags

Text	provides a general description of the exception.
Message	provides a more detailed description of the exception. An exception can contain zero or more messages.

Example

This example shows a possible response to a failed login.

```
<SystemLink version='1.0' hostVersion='1.0'>
 <LoginResponse>
  <Exception name='com.pjx.eScript.SecurityException'>
   <Text><![CDATA[Login Exception]]></Text>
   <Message type='error'>
    <Text><![CDATA[Unable to login user 'john doe': com.pjx.eScript.SecurityException:
com.ibm.as400.access.AS400SecurityException: Password is incorrect.]]></Text>
   </Message>
   <Message type='information'>
    <Text><![CDATA[Contact system administration]]></Text>
    <DetailedText><![CDATA[Contact your system administrator.]]></DetailedText>
   </Message>
  </Exception>
 </LoginResponse>
</ SystemLink >
```

Message

Message tag

A Message tag is used to communicate a line of text to the user.

Attributes

type	must be "error", "warning", or "information".

Nested Tags

Text	provides a general description.
DetailedText	this entity is optional and when specified, provides additional details.

See the Exception tag section for example usage.

Response Forwarding Commands

The following commands are used with the System-Link Response Forwarding feature. This feature, in conjunction with the proper configuration, will enable the associated System-Link response document to be sent to one or more designated recipients. For further information on this feature, please consult *Chapter 5.* at the beginning of this guide.

RecipientList

RecipientList tag

The RecipientList tag is used to specify one or more response forwarding recipients that will receive the System-Link response document. If a RecipientList section is included in a request document, it must appear after the Login section (if present), and before the Request section.

Attributes (may appear in any order)

sessionHandle	identifies the login session and user for this request. A session handle value of "*current" <u>must</u> be specified if the request was preceded by a login action in the same document. The request will be executed only if the login is successful.
copyForSender	If set to "true", the entire response document will be returned to the sender. If set to "false", only the LoginResponse, the RecipientListResponse, the outer tag of the Response action, and the LogoutResponse will be returned. Defaults to "true"

Nested Tags (in the order that they must be specified)

Recipient	specifies a single recipient to receive the response

Notes:
- the response document sent to the recipients will only contain the Response section, with the sessionHandle and workHandle attributes obscured. No session and workspace information will be sent to the recipients.
- if any of the recipients are not valid (i.e. are not in the current configuration), the entire RecipientList action will fail: none of the recipients will receive the response document, and the entire response document will be returned to the sender (regardless of the "copyForSender" setting).

Example

This example fragment indicates that two recipients, "RCV1" and "RCV2", should receive the response document. The original sender will not receive the document, since "copyForSender" is set to "false".

```
<Login userId='user1' password='mypass' maxIdle='0'
      properties='com.mapics.cas.domain.EnvironmentId=25,
               com.mapics.cas.domain.SystemName=USATLD06.INFOR.COM,
               com.mapics.cas.user.LanguageId=en' />
<RecipientList sessionHandle='*current' copyForSender='false'>
    <Recipient recipientId='RCV1'/>
    <Recipient recipientId='RCV2'/>
</RecipientList>

<Request sessionHandle='*current' workHandle='*new'
        broker='EJB' maxIdle='100000'>
...
```

`</Request>`

RecipientListResponse tag

A RecipientListResponse tag encompasses the result of a RecipientList request. The response will indicate if the response document could be sent to each recipient, as well as any exceptions encountered.

Attributes (may appear in any order)

sessionHandle	identifies the login session and user for this response.
actionSucceeded	set to "true" if the RecipientList action succeeded, otherwise set to "false"

Nested Tags

RecipientResponse	contains the status of a single recipient
Exception	if the RecipientList has failed.

Example 1

This example is a possible sender response fragment to the RecipientList Example 1 previously mentioned. In this example, all recipients were found in the current configuration, and each successfully received the response document. (Note that no nested subelements appear under the Response tag. This is due to the "copyForSender" setting in the previous example, as well as the lack of configuration errors in the RecipientList action.)

```
<LoginResponse actionSucceeded='true'>
  <SessionHandle value='f345'/>
</LoginResponse>

<RecipientListResponse sessionHandle='f345' actionSucceeded='true'>
  <Recipient recipientId='RCV1' messageSent='true'/>
  <Recipient recipientId='RCV2' messageSent='true'/>
</RecipientListResponse>

<Response sessionHandle='f345' workHandle='6c97d4'>
```

Example 2

This example is another possible sender response fragment to the RecipientList Example 1. In this example, all recipients were found in the current configuration, however recipient "RCV2" was not active (and thus did not receive the response document).

```
<LoginResponse actionSucceeded='true'>
  <SessionHandle value='f345'/>
</LoginResponse>

<RecipientListResponse sessionHandle='f345' actionSucceeded='false'>
  <Recipient recipientId='RCV1' messageSent='true'/>
  <Recipient recipientId='RCV2' messageSent='false'>
```

```
  <Exception name='com.pjx.eScript.CommunicationException'>
   <Text><![CDATA[Receiver communication failure]]></Text>
   <Message type='error'>
     <Text><![CDATA[The message could not be sent to receiver 'RCV2'.]]></Text>
     <DetailedText><![CDATA[Receiver 'RCV2' was not found.]]></DetailedText>
   </Message>
  </Exception>
 </Recipient>
</RecipientListResponse>

<Response sessionHandle='f345' workHandle='6c97d4'>
```

Recipient

Recipient tag

The Recipient tag is used to specify a single recipient that will receive the response document corresponding to the System-Link request.

Attributes (may appear in any order)

recipientId	the identifier for the recipient

Nested Tags

(none)

Note:
- Prior to use with a System-Link request, each recipient must be configured. For further information, please consult *Chapter 5.* at the beginning of this guide.

See the RecipientList tag for example usage.

RecipientResponse tag

A RecipientResponse tag encompasses the result of a Recipient specification.. The response will indicate if the response document could be sent to the corresponding recipient, as well as any exceptions encountered.

Attributes (may appear in any order)

recipientId	the identifier for the recipient
messageSent	if "true", the message was successfully received by the recipient. Otherwise, messageSent will be sent to "false", and an exception will detail the reason for the failure

Nested Tags

Exception	if the message could not be sent to the given Recipient.

See the RecipientListResponse tag for examples.

Utility Actions for Request Generation

The tags listed in this section are designed to assist in the creation of System-Link request documents. They list meta-information (classes, definition information) rather than actual database values.

ListCustomDfns

ListCustomDfns tag

The ListCustomDfns tag allows the construction of System-Link requests by listing the custom definitions available for use by the "DfnKey" and "ApplyTemplate" tags.

Attributes (may appear in any order)

name	an identifier for this list action. **Must be unique within a given System-Link request document.** Actions are named in the context of a given work area. If a subsequent action is requested with the same name for the same work area, any outstanding results for the old action will be reclaimed.
customDfnType	type of custom definition to list. Must be one of the following: "header", "subset", "sort", "cardfile", or "template"
clientClass	(Optional) the business object class for which this definition is used (e.g. "com.mapics.pm.Buyer"). This corresponds to the "domainClass" for QueryObject, Create, etc. Either a clientClass or a classMnemonic must be provided.
classMnemonic	(Optional) the class identifier corresponding to the business object class for which this definition is used (e.g. "BYR"). This corresponds to the "classMnemonic" for QueryObject, Create, etc. Either a domainClass or a classMnemonic must be provided.
access	specifies the security for the custom definitions in the list. Must be one of the following: "public", "private", or "both". "public" will return only public definitions. "private" will return only definitions private to the current user ID (used in the Login action). "both" will return both public and private definitions.

Nested Tags

(none)

Example

This example is a System-Link fragment that lists all of the public and private (to the current user) subset definitions currently defined for PoItem.

> <**ListCustomDfns name**='poSubsets' **customDfnType**='sort'
> **clientClass**='com.mapics.pm.PoItem' **access**='both'/>

ListCustomDfnsResponse tag

The ListCustomDfnsResponse tag contains the response of a ListCustomDfns action.

Attributes (may appear in any order)

name	the same name submitted to the ListCustomDfns action tag.
actionSucceeded	set to "true" if the ListCustomDfns action succeeded, otherwise set to "false"

Nested Tags

BasicCustomDfnInfo	contains the relevant information for a single custom definition
Exception	if the ListCustomDfns operation could not complete.

Example

This example is a System-Link fragment that lists a possible response to the example for the ListCustomDfns tag. The first definition is public, and the second definition is private to user "jackson". Note that the second subset contains a prompted property.

```
<ListCustomDfnsResponse name='poSubsets' actionSucceeded='true'>
    <BasicCustomDfnInfo objectId='f345' userid='(public)'>
        <DfnName><![CDATA[A Public Subset]]></DfnName>
    </BasicCustomDfnInfo>
    <BasicCustomDfnInfo objectId='f678' userid='jackson'>
        <DfnName><![CDATA[Buyer equals…]]></DfnName>
        <PromptedProperty path='buyer'>
            <OperatorType><![CDATA[Equals]]></OperatorType>
        </PromptedProperty>
    </BasicCustomDfnInfo>
</ListCustomDfnsResponse>
```

ListDomainClassInfo

ListDomainClassInfo tag

The ListDomainClassInfo tag lists domain class information applicable to System-Link actions.

Attributes (may appear in any order)

name	an identifier for this list action. **Must be unique within a given System-Link request document.** Actions are named in the context of a given work area. If a subsequent action is requested with the same name for the same work area, any outstanding results for the old action will be reclaimed.
domainClass	(Optional) the domain class for which information should be displayed. If "*all" is specified, then all applicable domain classes will be displayed. **Note that use of the "*all" option will take longer than if an individual class is specified** (especially if actions and properties are requested as well). Either a domainClass or a classMnemonic must be provided.
classMnemonic	(Optional) the class identifier corresponding to the domain class for which information should be displayed. If "*all" is specified, then all applicable domain classes will be displayed. **Note that use of the "*all" option will take longer than if an individual class is specified** (especially if actions and properties are requested as well). Either a domainClass or a classMnemonic must be provided.
showTitles	(Optional) if "true", then the default form title and list title for this class are displayed. Default is "false".
showActions	(Optional) if "true", then the System-Link actions allowed for this class are displayed. These will be described in the ListDomainClassInfoResponse tag section. Default is "false".
showOptions	(Optional) if "true", then the maintenance options for create and copy actions are displayed for this class (if present). Default is "false".

showRelationships	(Optional) if "true", then the relationships for this class are displayed. Default is "false".
showTextInfo	(Optional) if "true", then the text object (comments) information for this class is displayed. Default is "false".
showHostJobs	(Optional) if "true", then all host jobs defined for this class are displayed. Default is "false"
showHostReports	(Optional) if "true", then all host reports defined for this class are displayed. Default is "false".
displayProperties	(Optional) selects the display of properties on the domain class. Must be set to one of the following: "none", "keyOnly", "keySeparated", and "mixed". "none" will show no property information. "keyOnly" will only show the key properties inside of a <KeyInfo> tag pair. "keySeparated" will show the key properties inside of a <KeyInfo> tag pair, and non-key properties below them. "mixed" will show all properties (key and non-key) together as a list of properties. Default is "none".
showExtendedPropertyInfo	(Optional) if "true", then all properties with discrete domains will have both their values and descriptions displayed (using the PropertyExtendedInfo entity). Default is "false".

Nested Tags

(none)

Note:
- Some host jobs and host reports are displayed by Power-Link on multiple business objects (e.g. "Generate Reorder Recommendations" is defined on Warehouse, but is also displayed on Reorder Recommendations). In these cases, the host process in question will only be listed for (and usable with) the business object over which it is defined.

Example

This example is a System-Link fragment that lists the actions and properties (keySeparated) for Buyer.

> <**ListDomainClassInfo name**='buyerInfo' **domainClass**='com.mapics.pm.Buyer'
> **showActions**='true' **displayProperties**='keySeparated'/>

ListDomainClassInfoResponse tag

The ListDomainClassInfoResponse tag contains the response from a ListDomainClassInfo action.

Attributes (may appear in any order)

name	the same name submitted to the ListDomainClassInfo action tag.
domainClass	the domain class submitted to the ListDomainClassInfo action tag (or "*all" if specified).
actionSucceeded	set to "true" if the ListDomainClassInfo action succeeded, otherwise set to "false"

Nested Tags

DomainClassInfo	contains the relevant information for a single domain class
Exception	if the ListDomainClassInfo operation could not complete.

Example 1

This example is a System-Link fragment that is a partial response to the example for ListDomainClassInfo. (Note: not all properties are shown for the class.)

```
<ListDomainClassInfoResponse name='buyerInfo' domainClass='com.mapics.pm.Buyer'
                                actionSucceeded='true'>
        <DomainClassInfo domainClass="com.mapics.pm.Buyer" classMnemonic='BYR'>
                <ActionsAllowed canCopy="true" canCreate="true" canDelete="true"
                                canDisplay="true" canUpdate="true" />
                <KeyInfo>
                        <PropertyInfo path="buyer" format="string" canCreate="true"
                                canUpdate="false"><![CDATA[Buyer]]></PropertyInfo>
                                isNullAllowed="false"><![CDATA[Buyer]]></PropertyInfo>
                </KeyInfo>
                <PropertyInfo path="ifmApprover" format="string" canCreate="true"
                                canUpdate="true"><![CDATA[IFM approver]]></PropertyInfo>
                                isNullAllowed="false"><![CDATA[IFM approver]]></PropertyInfo>
                <PropertyInfo path="buyerRating" format="decimal" canCreate="false"
                                canUpdate="false"><![CDATA[ Buyer rating]]></PropertyInfo>
                                isNullAllowed="false"><![CDATA[Buyer rating]]></PropertyInfo>
                <PropertyInfo path="averageOrders" format="integer" canCreate="true"
                                canUpdate="true"><![CDATA[Average orders]]></PropertyInfo>
                                isNullAllowed="false"><![CDATA[Average orders]]></PropertyInfo>
                <PropertyInfo path="numberOfUnderShipments" format="integer" canCreate="false"
                                canUpdate="false"><![CDATA[Number of under
shipments]]></PropertyInfo>
                                isNullAllowed="false"><![CDATA[Number … ]]></PropertyInfo>
                <PropertyInfo path="lastMaintainedDate" format="date" canCreate="false"
                                canUpdate="false"><![CDATA[Last maintained date]]></PropertyInfo>
                                isNullAllowed="false"><![CDATA[Last maintained date]]></PropertyInfo>
                <PropertyInfo path="userFieldSwitchC" format="boolean" canCreate="true"
                                canUpdate="true"><![CDATA[User field - switch c]]></PropertyInfo>
                                isNullAllowed="false"><![CDATA[User field – switch c] </PropertyInfo>
                        …
        </DomainClassInfo>
</ListDomainClassInfoResponse>
```

Example 2

This example is another System-Link fragment showing title, relationship, option, and text object information. Note that some copy options appear nested within other copy options. (Note that not all information of each type is shown.)

```
<ListDomainClassInfoResponse name='poInfo' domainClass='com.mapics.pm.PurchaseOrder'
                             actionSucceeded='true'>
   <DomainClassInfo domainClass="com.mapics.pm.PurchaseOrder" classMnemonic="POR">
      <FormTitle><![CDATA[Purchase Order]]></FormTitle>
      <ListTitle><![CDATA[Purchase Orders]]></ListTitle>

      <CopyOptionInfo>
         <RelationshipOptionInfo relationshipName="relatedPoComments"
                                 toClass="com.mapics.pm.PoComment"
                                 hasCopyTemplate="false" />
         <RelationshipOptionInfo relationshipName="relatedPoItems"
                                 toClass="com.mapics.pm.PoItem"
                                 hasCopyTemplate="true">
            <RelationshipOptionInfo relationshipName="relatedPoItemComments"
                                    toClass="com.mapics.pm.PoItemComment"
                                    hasCopyTemplate="false" />
            <RelationshipOptionInfo relationshipName="relatedPoItemReleases"
                                    toClass="com.mapics.pm.PoItemRelease"
                                    hasCopyTemplate="true">
               <RelationshipOptionInfo relationshipName="relatedPoItemReleaseComments"
                                       toClass="com.mapics.pm.PoItemReleaseComment"
                                       hasCopyTemplate="false" />
            </RelationshipOptionInfo>

            <RelationshipOptionInfo relationshipName="relatedPoItemOperations"
                                    toClass="com.mapics.pm.PoItemReceivingOperation"
                                    hasCopyTemplate="true" />
         </RelationshipOptionInfo>
      </CopyOptionInfo>

      <RelationshipInfo relationshipName="relatedBuyer" toClass="com.mapics.pm.Buyer"
                        relationshipType="Many-to-1"><![CDATA[Buyer]]></RelationshipInfo>

       ...

      <RelationshipInfo relationshipName="relatedPoItems" toClass="com.mapics.pm.PoItem"
                        relationshipType="1-to-Many"><![CDATA[Purchase Order Items]]>
      </RelationshipInfo>

      <TextRelationshipInfo relationshipName="relatedPoComments" typed="true">
         <TextDescriptor type="label">
            <Value><![CDATA[Purchase Order Comments]]></Value>
         </TextDescriptor>
         <TextDescriptor type="label">
            <Type><![CDATA[P]]></Type>
            <Value><![CDATA[Order]]></Value>
         </TextDescriptor>
         <TextDescriptor type="label">
            <Type><![CDATA[C]]></Type>
            <Value><![CDATA[Closeout]]></Value>
         </TextDescriptor>
         <TextDescriptor type="label">
            <Type><![CDATA[T]]></Type>
            <Value><![CDATA[Internal]]></Value>
```

```
            </TextDescriptor>
        </TextRelationshipInfo>
    </DomainClassInfo>
</ListDomainClassInfoResponse>
```

Example 3

This example is another System-Link fragment illustrating create option info.

```
<ListDomainClassInfoResponse name='poItemInfo' domainClass='com.mapics.pm.PoItem'
                              actionSucceeded='true'>
    <DomainClassInfo domainClass="com.mapics.pm.PoItem" classMnemonic="PORITM">
        <CreateOptionInfo>
            <OptionInfo optionName="includeVendorItemComments" optionType="boolean"
                    constraint="boolean">
                <OptionDescriptor>
                    <Value><![CDATA[1]]></Value>
                    <Description><![CDATA[Yes]]></Description>
                </OptionDescriptor>
                <OptionDescriptor>
                    <Value><![CDATA[0]]></Value>
                    <Description><![CDATA[No]]></Description>
                </OptionDescriptor>
            </OptionInfo>
            <OptionInfo optionName="includeItemReceivingOperations" optionType="boolean"
                    constraint="boolean">
                <OptionDescriptor>
                    <Value><![CDATA[1]]></Value>
                    <Description><![CDATA[Yes]]></Description>
                </OptionDescriptor>
                <OptionDescriptor>
                    <Value><![CDATA[0]]></Value>
                    <Description><![CDATA[No]]></Description>
                </OptionDescriptor>
            </OptionInfo>
        </CreateOptionInfo>
        <CopyOptionInfo>

            …

        </CopyOptionInfo>
    </DomainClassInfo>
</ListDomainClassInfoResponse>
```

Example 4

This example is another System-Link fragment illustrating host job, host report, and extended property information.

```
<ListDomainClassInfoResponse name='hostProcsWithExtended'
                             domainClass='com.mapics.mm.Warehouse' actionSucceeded='true'>
    <DomainClassInfo domainClass='com.mapics.mm.Warehouse' classMnemonic='WHS'>
        <HostJobInfo canCreate='true'>
            <HostProcessDescriptor mnemonic='ALCQTYADT'><![CDATA[Audit Allocation Quantities]]>
            </HostProcessDescriptor>
            <PropertyInfo path='description' format='string' canCreate='true' canUpdate='true'
                    isNullAllowed='false'><![CDATA[Description]]></PropertyInfo>
        </HostJobInfo>
        <HostJobInfo canCreate='true'>
```

```xml
        <HostProcessDescriptor mnemonic='LOCQTYADTS'><![CDATA[Audit Location Quantities]]>
        </HostProcessDescriptor>
        <PropertyInfo path='description' format='string' canCreate='true' canUpdate='true'
                    isNullAllowed='false'><![CDATA[Description]]></PropertyInfo>
        <PropertyExtendedInfo path='detailSummaryOption' format='boolean' constraint='boolean'
                            canCreate='true' canUpdate='true' isNullAllowed='false'>
          <Description><![CDATA[Report format]]></Description>
          <ConstrainedValues>
              <Value><![CDATA[1]]></Value>
              <Description><![CDATA[Detail]]></Description>
          </ConstrainedValues>
          <ConstrainedValues>
              <Value><![CDATA[0]]></Value>
              <Description><![CDATA[Summary]]></Description>
          </ConstrainedValues>
        </PropertyExtendedInfo>
      </HostJobInfo>
      <HostJobInfo canCreate='true'>

      ...
      <HostReportInfo canCreate='true'>
          <HostProcessDescriptor mnemonic='MORPRG'><![CDATA[M.O. Transaction Register]]>
          </HostProcessDescriptor>
          <PropertyInfo path='description' format='string' canCreate='true' canUpdate='true'
                      isNullAllowed='false'><![CDATA[Description]]></PropertyInfo>
      </HostReportInfo>
      <HostReportInfo canCreate='true'>
          <HostProcessDescriptor mnemonic='ORDSHTPRT'><![CDATA[Order Shortages]]>
          </HostProcessDescriptor>
          <PropertyInfo path='description' format='string' canCreate='true' canUpdate='true'
                      isNullAllowed='false'><![CDATA[Description]]></PropertyInfo>
      </HostReportInfo>
    </DomainClassInfo>
</ListDomainClassInfoResponse>
```

BasicCustomDfnInfo

BasicCustomDfnInfo tag

The BasicCustomDfnInfo tag contains the relevant information for a single custom definition.

Attributes (may appear in any order)

objectId	the object Id string uniquely identifying the desired definition.
userid	If the definition is private, userid will identify the user id of the definition's owner. If the definition is public, userid will have the value "(public)".
subtype	(Optional) the subtype of the custom definition, if one is defined for this custom definition type.

Nested Tags

DfnName	the name or title of the definition
PromptedProperty	(subset only) If the subset definition has prompted properties, this will list the property path, as well as describe the operator type (e.g. "equals", "less than")

For examples, see the ListCustomDfnsResponse tag.

DomainClassInfo

DomainClassInfo tag

The DomainClassInfo tag contains the relevant information for a single domain class.

Attributes (may appear in any order)

domainClass	the name of the domain class.
classMnemonic	the class identifier corresponding to the domain class

Nested Tags

FormTitle	contains the title String displayed for forms. Only given if "showTitles" on ListDomainClassInfo was "true".
ListTitle	contains the title String displayed for lists. Only given if "showTitles" on ListDomainClassInfo was "true".
ActionsAllowed	actions allowed on this domain class. Only given if "showActions" on ListDomainClassInfo was "true".
CreateOptionInfo	contains a list of create options valid for create actions on the domain class. Only given if "showOptions" on ListDomainClassInfo was "true", and if create options are present on the domain class.
CopyOptionInfo	contains a list of copy options valid for copy actions on the domain class. Only given if "showOptions" on ListDomainClassInfo was "true", and if copy options are present on the domain class.

KeyInfo	contains a list of PropertyInfo and PropertyExtendedInfo subelements for the key properties on this domain class. Only given if "displayProperties" on ListDomainClassInfo was set to either "keyOnly" or "keySeparated".
PropertyInfo	lists the relevant information for a single property on the domain class. Only given if "displayProperties" on ListDomainClassInfo was set to either "keySeparated" or "mixed".
PropertyExtendedInfo	lists the relevant information for a single constrained property on the domain class. Only given if "displayProperties" on ListDomainClassInfo was set to either "keySeparated" or "mixed", if "showExtendedPropertyInfo" on ListDomainClassInfo was set to "true", and if the property's values are defined over a discrete domain.
RelationshipInfo	lists the relevant information for a single relationship on the domain class. Only given if "showRelationships" on ListDomainClassInfo was "true", and if one or more relationships are present on the domain class.
TextRelationshipInfo	lists the relevant information for a single text (comment) relationship on the domain class. Only given if "showTextInfo" on ListDomainClassInfo was "true", and if one or more text objects are present on the domain class.
HostJobInfo	lists the relevant information for a single host job defined on the domain class.
HostReportInfo	lists the relevant information for a single host report defined on the domain class.

For examples, see the ListDomainClassInfoResponse tag.

FormTitle

FormTitle tag

A FormTitle tag contains the title used when displaying a single instance of the domain class (e.g. in the Power-Link object browser).

Attributes

(none)

Nested Tags

a String value

See the ListDomainClassInfoResponse tag for example usage.

ListTitle

ListTitle tag

A ListTitle tag contains the title used when displaying a list of instances of the domain class (e.g. in the Power-Link list browser).

Attributes

(none)

Nested Tags

a String value

See the ListDomainClassInfoResponse tag for example usage.

ActionsAllowed

ActionsAllowed tag

The ActionsAllowed tag shows the System-Link actions permitted for objects of the given domain class.

Attributes (may appear in any order)

canCopy	set to "true" if an instance of the domain class can be copied.
canCreate	set to "true" if an instance of the domain class can be created.
canDelete	set to "true" if an instance of the domain class can be deleted.
canDisplay	set to "true" if an instance of the domain class can be displayed.
canUpdate	set to "true" if an instance of the domain class can be updated.

Nested Tags
(none)

See the ListDomainClassInfoResponse tag for example usage.

CreateOptionInfo

CreateOptionInfo tag

The CreateOptionInfo tag shows maintenance options supported for the creation of a new instance of a given domain class.

Attributes
(none)

Nested Tags

OptionInfo	contains the information associated with a single create option

See the ListDomainClassInfoResponse tag for example usage.

CopyOptionInfo

CopyOptionInfo tag

The CopyOptionInfo tag shows maintenance options supported for copying an existing instance of a given domain class.

Attributes

(none)

Nested Tags

RelationshipOptionInfo	contains the information associated with a single option for copying related objects
OptionInfo	contains the information associated with a single "simple" copy option

See the ListDomainClassInfoResponse tag for example usage.

RelationshipOptionInfo

RelationshipOptionInfo tag

The RelationshipOptionInfo tag shows information on a single relationship maintenance option. (An example of this type of copy option would be to copy related Purchase Order Items along with the Purchase Order.)

Attributes (may appear in any order)

relationshipName	the name of the associated relationship
toClass	the domain class of the related business objects
hasCopyTemplate	does the toClass permit copy templates? If hasCopyTemplate is "true", a nested ApplyTemplate option may be specified as part of the matching RelationshipOption.

Nested Tags

RelationshipOptionInfo	contains the information associated with a single option for copying related objects
OptionInfo	contains the information associated with a single "simple" copy option

See the ListDomainClassInfoResponse tag for example usage.

OptionInfo

OptionInfo tag

The OptionInfo tag shows information on a single "simple" (non-relationship and non-template) maintenance option.

Attributes (may appear in any order)

optionName	the name of the option
optionType	the associated type of the option value (e.g. BooleanAttribute, int)
constraint	the constraint over the allowed values in this property's discrete domain. Will be one of the following: "boolean", "list", "range", or "string".

Nested Tags

OptionDescriptor	contains information about a single value for the associated option

Notes:

- If constraint is "boolean", two OptionDescriptors will be given: one for the "true" value and one for the "false" value.
- If constraint is "list", one OptionDescriptor will be given for each valid value.
- If constraint is "range", two OptionDescriptors will be given: one for each of the range bounds.
- If constraint is "string", then any String value is permitted.

See the ListDomainClassInfoResponse tag for example usage.

OptionDescriptor

OptionDescriptor tag

The OptionDescriptor tag contains information about a single value for a given option.

Attributes
(none)

Nested Tags (either Value or NullValue will be present)

Value	the option value (e.g. "1")
Null Value	indicates that the option is null value
Description	the value's description (e.g. "True")

Description

Description tag

A Description tag describes a given data entity.

Attributes
(none)

Nested Tags
a String value

KeyInfo

KeyInfo tag

The KeyInfo tag contains a list of key properties for the given domain class.

Attributes
(none)

Nested Tags

PropertyInfo	contains information about a single key property

See the ListDomainClassInfoResponse tag for example usage.

PropertyInfo

PropertyInfo tag

The PropertyInfo tag contains the relevant information for a single property on the given domain class.

Attributes (may appear in any order)

path	the property path for the property.
format	the format for this property. Will be one of the following: "string", "date", "time", "integer", "long", "short", "decimal", "double", "float", "boolean", or "timestamp". Note: boolean properties will use "0" for "false" and "1" for "true".
canCreate	set to "true" if this property can be set during a Create action.
canUpdate	set to "true" if this property can be maintained, or "false" if it is read-only.
isNullAllowed	set to "true" if null is a valid value for the property.

Nested Tags
A String listing the description of the property.

See the ListDomainClassInfoResponse tag for example usage.

PropertyExtendedInfo

PropertyExtendedInfo tag

The PropertyExtendedInfo tag displays information for a single property defined over a discrete domain. PropertyExtendedInfo will only appear if the "showExtendedPropertyInfo" attribute in the ListDomainClassInfo request action was set to "true".

Attributes (may appear in any order)

path	the property path for the property.
format	the format for this property. Will be one of the following: "string", "date", "time", "integer", "long", "short", "decimal", "double", "float", "boolean", or "timestamp". Note: boolean properties will use "0" for "false" and "1" for "true".
constraint	the constraint over the allowed values in this property's discrete domain. Will be one of the following: "boolean", "list", or "range".
canCreate	set to "true" if this property can be set during a Create action.
canUpdate	set to "true" if this property can be maintained, or "false" if it is read-only.
isNullAllowed	set to "true" if null is a valid value for the property.

Nested Tags

Description	the description of the property
ConstrainedValue	contains information about a single value for the associated property

Notes:

- If constraint is "boolean", two ConstrainedValues will be given: one for the "true" value and one for the "false" value.
- If constraint is "list", one ConstrainedValues will be given for each valid value.
- If constraint is "range", two ConstrainedValues s will be given: one for each of the range bounds.

See the ListDomainClassInfoResponse tag for example usage.

ConstrainedValue

ConstrainedValue tag

The ConstrainedVAlue tag contains information about a single value for a given property that is constrained over a discrete domain.

Attributes
(none)

Nested Tags (either Value or NullValue will be present)

Value	the option value (e.g. "1")

Null Value	indicates that the option is null value
Description	the value's description (e.g. "True")

RelationshipInfo

RelationshipInfo tag

The RelationshipInfo tag contains the relevant information for a single relationship on the domain class.

Attributes (may appear in any order)

relationshipName	the property path for the relationship.
toClass	the domain class for the related business objects
relationshipType	the cardinality of the relationship. Valid values are "Many-to-1" and "1-to-Many".

Nested Tags

A String listing the description of the relationship.

See the ListDomainClassInfoResponse tag for example usage.

TextRelationshipInfo

TextRelationshipInfo tag

The RelationshipInfo tag contains the relevant information for a single text (comment) relationship on the domain class.

Attributes (may appear in any order)

relationshipName	the property path for the relationship to the text object class
typed	is text object typed? Valid values are "true" and "false".

Nested Tags

TextDescriptor	describes the text object, as well as each type (if the text object is typed)

See the ListDomainClassInfoResponse tag for example usage.

HostJobInfo

HostJobInfo tag

The HostJobInfo tag contains the relevant information for a single host job defined over the domain class

Attributes (may appear in any order)

canCreate	set to "true" if the current user is authorized to run this host job

Nested Tags

HostProcessDescriptor	describes the indentification information used to select the given host job
PropertyInfo	lists the relevant information for a single property on the host job.
PropertyExtendedInfo	lists the relevant information for a single constrained property on the host job. Only given if "displayProperties" on ListDomainClassInfo was set to either "keySeparated" or "mixed", if "showExtendedPropertyInfo" on ListDomainClassInfo was set to "true", and if the property's values are defined over a discrete domain.

See the ListDomainClassInfoResponse tag for example usage.

HostReportInfo

HostReportInfo tag

The HostReportInfo tag contains the relevant information for a single host report defined over the domain class

Attributes (may appear in any order)

canCreate	set to "true" if the current user is authorized to run this host report

Nested Tags

HostProcessDescriptor	describes the indentification information used to select the given host report
PropertyInfo	lists the relevant information for a single property on the host report.
PropertyExtendedInfo	lists the relevant information for a single constrained property on the host report. Only given if "displayProperties" on ListDomainClassInfo was set to either "keySeparated" or "mixed", if "showExtendedPropertyInfo" on ListDomainClassInfo was set to "true", and if the property's values are defined over a discrete domain.

See the ListDomainClassInfoResponse tag for example usage.

HostProcessDescriptor

HostProcessDescriptor tag

The HostProcessDescriptor tag is used to describe the host process (host job or host report) identification information. This information will be specified in the HostJobType or HostReportType tags as part of a RunHostJob or CreateHostReport subaction respectively.

Attributes

mnemonic	the unique identifier for the host process.

Nested Tags

A String giving the name of the host process.

The corresponding HostJobType or HostReportType tag may reference the host process either by mnemonic or by specifying the host process name. (mnemonic is preferred, as it guarantees uniqueness.) If both mnemonic and a host process name are given, the mnemonic is used.

See the ListDomainClassInfoResponse tag for example usage.

Utility Methods for Administration

The tags listed in this section are for miscellaneous administration functions.

RefreshCustomizationData

RefreshCustomizationData tag

The RefreshCustomizationData tag clears the cached copies of any customization data in the System-Link Server instances. If a Power-Link performs customization maintenance, this command causes the System-Link Server instances to refetch the updated data from the server. (This is similar to the "File→Refresh" menu action in Power-Link.)

Note: since the caches are not persistent, stopping and restarting the System-Link Servers will have the same effect.

Attributes

name	an identifier for this refresh action. **Must be unique within a given System-Link request document.** Actions are named in the context of a given work area. If a subsequent action is requested with the same name for the same work area, any outstanding results for the old action will be reclaimed.

Nested Tags

(none)

Example

This example is a System-Link fragment that forces a customization refresh on all System-Link Server instances for the environment.

<**RefreshCustomizationData name**='refresh1' />

RefreshCustomizationDataResponse tag

A RefreshCustomizationDataResponse tag encompasses the result of a RefreshCustomizationData request. The RefreshCustomizationDataResponse is linked back to its corresponding RefreshCustomizationData action via the name attribute. If the RefreshCustomizationData action fails, the RefreshCustomizationDataResponse will contain an exception detailing the reason for failure.

Attributes

name	the same name submitted to the RefreshCustomizationData action tag.
actionSucceeded	set to "true" if the RefreshCustomizationData action succeeded, otherwise set to "false"

Nested Tags

Exception	(one or more) if the RefreshCustomizationData operation could not complete.

Chapter 11.   Document Type Definitions (DTD)

These DTDs are used as-is by the programs on both ends to help parse the requests and responses.

SystemLinkRequest.DTD

```
<?xml version='1.0' encoding='ISO-8859-1'?>

<!ELEMENT System-Link (Login?, RecipientList?, Request?, Logout?)>
<!ATTLIST System-Link
        version CDATA "1.0"
        hostVersion CDATA "1.0"
        formatForDisplay (true|false) "false"
        includeNewlines (true|false) #IMPLIED
        useTimeZone (default|GMT) "default">

<!ELEMENT Login EMPTY>
<!ATTLIST Login
        principal CDATA #IMPLIED
        userId CDATA #IMPLIED
        credentials CDATA #IMPLIED
        password CDATA #IMPLIED
        maxIdle CDATA #REQUIRED
        properties CDATA #REQUIRED>

<!ELEMENT Logout EMPTY>
<!ATTLIST Logout
        sessionHandle CDATA #REQUIRED>

<!ELEMENT RecipientList (Recipient*)>
<!ATTLIST RecipientList
        sessionHandle CDATA #REQUIRED
        copyForSender (true|false) "true">

<!ELEMENT Recipient EMPTY>
<!ATTLIST Recipient
        recipientId CDATA #REQUIRED>

<!ELEMENT Request (QueryList?|QueryAttachmentList?|QueryObject?|
                QueryAttachmentObject?|
                Create?|Update?|Delete?|
                CreateAttachment?|UpdateAttachment?|DeleteAttachment?|
                FetchNext?|FetchNextAttachments?|
                FetchNextMaintenanceHistory?|FetchNextText?|
                IfCondition?|EndIfCondition?|StartTransactionGroup?|
                ListCustomDfns?|ListDomainClassInfo?|
                ListCurrentEnvironment?|ListSessionInfo?|
                RefreshCustomizationData?)+>
<!ATTLIST Request
        sessionHandle CDATA #REQUIRED
        workHandle CDATA #REQUIRED
        broker (EJB) "EJB"
        maxIdle CDATA #REQUIRED>

<!-- QueryList -->
<!ELEMENT QueryList ((QueryDfn|Pql), Constraint?, QueryText*,
                (RunHostJob|CreateHostReport)*)>
<!ATTLIST QueryList
        name CDATA #REQUIRED
        domainClass CDATA #IMPLIED
        classMnemonic CDATA #IMPLIED
```

```
                includeMetaData (true|false) "false"
                maxReturned CDATA #REQUIRED
                retainResult (true|false) "true">


<!-- QueryAttachmentList -->
<!ELEMENT QueryAttachmentList ((QueryDfn|Pql), OwningObjectKey?)>
<!ATTLIST QueryAttachmentList
                name CDATA #REQUIRED
                owningDomainClass CDATA #IMPLIED
                owningClassMnemonic CDATA #IMPLIED
                includeGlobals (true|false) "true"
                includeMetaData (true|false) "false"
                maxReturned CDATA #REQUIRED
                retainResult (true|false) "true">


<!-- QueryDfn -->
<!ELEMENT QueryDfn (HeaderDfnKey?, SubsetDfnKey?, PromptedSubsetValues?,
SortDfnKey?)>

<!ELEMENT HeaderDfnKey (#PCDATA)*>
<!ATTLIST HeaderDfnKey
                objectId CDATA #IMPLIED
                clientClass CDATA #IMPLIED
                classMnemonic CDATA #IMPLIED>

<!ELEMENT SubsetDfnKey (#PCDATA)*>
<!ATTLIST SubsetDfnKey
                objectId CDATA #IMPLIED
                clientClass CDATA #IMPLIED
                classMnemonic CDATA #IMPLIED>

<!ELEMENT SortDfnKey (#PCDATA)*>
<!ATTLIST SortDfnKey
                objectId CDATA #IMPLIED
                clientClass CDATA #IMPLIED
                classMnemonic CDATA #IMPLIED>

<!ELEMENT PromptedSubsetValues (CriteriaSpec+)>

<!ELEMENT CriteriaSpec (Operator?, (Value|ValueRef))>
<!ATTLIST CriteriaSpec
                path CDATA #IMPLIED
                alias CDATA #IMPLIED>

<!ELEMENT Operator (#PCDATA)*>

<!-- OwningObjectKey - for Attachment functions -->
<!ELEMENT OwningObjectKey (Property+)>

<!-- MediaFile elements - for Attachment functions -->
<!ELEMENT MediaFilePath (#PCDATA)*>

<!-- Constraint -->
<!ELEMENT Constraint (Key)>
<!ATTLIST Constraint
                domainClass CDATA #IMPLIED
                classMnemonic CDATA #IMPLIED
                relationshipName CDATA #REQUIRED>


<!-- QueryObject -->
<!ELEMENT QueryObject ( ((Key, CardFileDfnKey?) | Pql), QueryText*,
                        QueryMaintenanceHistory?,
                        (RunHostJob|CreateHostReport)*,
```

```
                              QueryRelationship*)>
<!ATTLIST QueryObject
        name CDATA #REQUIRED
        domainClass CDATA #IMPLIED
        classMnemonic CDATA #IMPLIED
        includeMetaData (true|false) "false"
        showResult (true|false) "true"
        retainResult (true|false) "false"
        existenceCheck (true|false) "false"
        alwaysReturnKey (true|false) "false">


<!-- QueryAttachmentObject -->
<!ELEMENT QueryAttachmentObject ((CardFileDfnKey? | Pql), MediaFilePath,
                                 OwningObjectKey?)>
<!ATTLIST QueryAttachmentObject
        name CDATA #REQUIRED
        owningDomainClass CDATA #IMPLIED
        owningClassMnemonic CDATA #IMPLIED
        includeMetaData (true|false) "false"
        showResult (true|false) "true"
        retainResult (true|false) "false">


<!ELEMENT CardFileDfnKey (#PCDATA)*>
<!ATTLIST CardFileDfnKey
        objectId CDATA #IMPLIED
        clientClass CDATA #IMPLIED
        classMnemonic CDATA #IMPLIED>


<!-- QueryRelationship -->
<!ELEMENT QueryRelationship ((QueryDfn|Pql), QueryText*, (QueryRelationship)*)>
<!ATTLIST QueryRelationship
        name CDATA #REQUIRED
        relationshipName CDATA #REQUIRED
        includeMetaData (true|false) "false"
        maxReturned CDATA #REQUIRED>



<!-- QueryText -->
<!ELEMENT QueryText (TextFilter?)>
<!ATTLIST QueryText
        name CDATA #REQUIRED
        relationshipName CDATA #REQUIRED
        includeMetaData (true|false) "false"
        maxCharsReturned CDATA #REQUIRED
        retainResult (true|false) "true">

<!ELEMENT TextFilter (Type+)>

<!-- QueryMaintenanceHistory -->
<!ELEMENT QueryMaintenanceHistory (QueryDfn|Pql)>
<!ATTLIST QueryMaintenanceHistory
        name CDATA #REQUIRED
        includeMetaData (true|false) "false"
        maxReturned CDATA #REQUIRED
        retainResult (true|false) "true">



<!-- Host Processes -->
<!ELEMENT CreateHostReport (HostReportType, ProcessContent,
                            ProcessConfirmation?, ProcessOutput?)>
<!ATTLIST CreateHostReport
        name CDATA #REQUIRED>
```

```
<!ELEMENT RunHostJob (HostJobType, ProcessContent, ProcessConfirmation?)>
<!ATTLIST RunHostJob
        name CDATA #REQUIRED>

<!ELEMENT HostReportType (#PCDATA)*>
<!ATTLIST HostReportType
        mnemonic CDATA #IMPLIED>

<!ELEMENT HostJobType (#PCDATA)*>
<!ATTLIST HostJobType
        mnemonic CDATA #IMPLIED>

<!ELEMENT ProcessContent (Property*)>

<!ELEMENT ProcessConfirmation (EmailHeader, EmailBody?)>

<!ELEMENT EmailHeader (AddressTo*, AddressCc*, AddressBcc*, Subject?)>

<!ELEMENT AddressTo (#PCDATA)*>
<!ELEMENT AddressCc (#PCDATA)*>
<!ELEMENT AddressBcc (#PCDATA)*>
<!ELEMENT Subject (#PCDATA)*>

<!ELEMENT EmailBody (#PCDATA)*>

<!ELEMENT ProcessOutput ((EmailHeader, EmailBody?)?)>
<!ATTLIST ProcessOutput
        saveFormatText (true|false) "false"
        saveFormatHtml (true|false) "false"
        saveFormatPdf (true|false) "false"
        attachTo (none|global) "none"
        emailAttachmentFormat (none|text|html|pdf) "none">


<!-- FetchNext -->
<!ELEMENT FetchNext EMPTY>
<!ATTLIST FetchNext
        name CDATA #REQUIRED
        queryName CDATA #REQUIRED
        includeMetaData (true|false) "false"
        maxReturned CDATA #REQUIRED>

<!ELEMENT FetchNextAttachments EMPTY>
<!ATTLIST FetchNextAttachments
        name CDATA #REQUIRED
        queryAttachmentName CDATA #REQUIRED
        includeMetaData (true|false) "false"
        maxReturned CDATA #REQUIRED>

<!ELEMENT FetchNextText EMPTY>
<!ATTLIST FetchNextText
        name CDATA #REQUIRED
        queryTextName CDATA #REQUIRED
        includeMetaData (true|false) "false"
        maxCharsReturned CDATA #REQUIRED>

<!ELEMENT FetchNextMaintenanceHistory EMPTY>
<!ATTLIST FetchNextMaintenanceHistory
        name CDATA #REQUIRED
        queryHistoryName CDATA #REQUIRED
        includeMetaData (true|false) "false"
        maxReturned CDATA #REQUIRED>
```

```
<!-- ApplyTemplate - for Create / Copy and Update -->
<!ELEMENT ApplyTemplate (#PCDATA)*>
<!ATTLIST ApplyTemplate
          objectId CDATA #IMPLIED
          clientClass CDATA #IMPLIED
          classMnemonic CDATA #IMPLIED>


<!-- ReasonCode - for all maintenance functions -->
<!ELEMENT ReasonCode (#PCDATA)*>


<!-- LogicalTransaction - for all maintenance functions -->
<!ELEMENT LogicalTransaction (#PCDATA)*>


<!-- SourceObject - used with Create for a Copy -->
<!ELEMENT SourceObject (Key)>
<!ATTLIST SourceObject
          domainClass CDATA #IMPLIED
          classMnemonic CDATA #IMPLIED>


<!-- SourceAttachment - used with CreateAttachment for a Copy -->
<!ELEMENT SourceAttachment (OwningObjectKey?)>
<!ATTLIST SourceAttachment
          owningDomainClass CDATA #IMPLIED
          owningClassMnemonic CDATA #IMPLIED>


<!-- MaintenanceOptions - create and copy options -->
<!ELEMENT MaintenanceOptions (RelationshipOption|Option)+>

<!ELEMENT RelationshipOption (ApplyTemplate?, (RelationshipOption|Option)*)>
<!ATTLIST RelationshipOption
          relationshipName CDATA #REQUIRED>


<!ELEMENT Option (Value|ValueRef|NullValue)>
<!ATTLIST Option
          optionName CDATA #REQUIRED>


<!-- Create, Update, Delete -->
<!ELEMENT Create (LogicalTransaction?, ReasonCode?, ApplyTemplate?, SourceObject?,
                  MaintenanceOptions?, DomainEntity, MaintainText*)>
<!ATTLIST Create
          name CDATA #REQUIRED
          domainClass CDATA #IMPLIED
          classMnemonic CDATA #IMPLIED
          retainResult (true|false) "false">

<!ELEMENT Update (LogicalTransaction?, ReasonCode?, ApplyTemplate?, DomainEntity,
                  MaintainText*)>
<!ATTLIST Update
          name CDATA #REQUIRED
          domainClass CDATA #IMPLIED
          classMnemonic CDATA #IMPLIED
          retainResult (true|false) "false">

<!ELEMENT Delete (LogicalTransaction?, ReasonCode?, DomainEntity)>
<!ATTLIST Delete
          name CDATA #REQUIRED
          domainClass CDATA #IMPLIED
          classMnemonic CDATA #IMPLIED>


<!ELEMENT MaintainText (TextAction+)>
<!ATTLIST MaintainText
          name CDATA #REQUIRED
```

```
          relationshipName CDATA #REQUIRED>

<!ELEMENT TextAction (Type?, (Value|ValueRef))>
<!ATTLIST TextAction
          valueType (manual) #REQUIRED
          actionType (addToBeginning|addToEnd|replace) "addToEnd">

<!ELEMENT CreateAttachment (LogicalTransaction?, ReasonCode?, ApplyTemplate?,
                            MediaFilePath, SourceAttachment?,
                            OwningObjectKey?, DomainEntity?)>
<!ATTLIST CreateAttachment
          name CDATA #REQUIRED
          owningDomainClass CDATA #IMPLIED
          owningClassMnemonic CDATA #IMPLIED
          retainResult (true|false) "false">

<!ELEMENT UpdateAttachment (LogicalTransaction?, ReasonCode?, ApplyTemplate?,
                            MediaFilePath, OwningObjectKey?, DomainEntity)>
<!ATTLIST UpdateAttachment
          name CDATA #REQUIRED
          owningDomainClass CDATA #IMPLIED
          owningClassMnemonic CDATA #IMPLIED
          retainResult (true|false) "false">

<!ELEMENT DeleteAttachment (LogicalTransaction?, ReasonCode?, MediaFilePath,
                            OwningObjectKey?)>
<!ATTLIST DeleteAttachment
          name CDATA #REQUIRED
          owningDomainClass CDATA #IMPLIED
          owningClassMnemonic CDATA #IMPLIED>

<!-- IfCondition, EndIfCondition -->
<!ELEMENT IfCondition (Condition+)>
<!ATTLIST IfCondition
          name CDATA #REQUIRED>

<!ELEMENT Condition EMPTY>
<!ATTLIST Condition
          actionName CDATA #REQUIRED
          actionStatus (succeeded|failed|moreResults|noMoreResults) #REQUIRED>

<!ELEMENT EndIfCondition EMPTY>

<!-- StartTransactionGroup -->
<!ELEMENT StartTransactionGroup EMPTY>

<!-- ListCustomDfns -->
<!ELEMENT ListCustomDfns EMPTY>
<!ATTLIST ListCustomDfns
          name CDATA #REQUIRED
          customDfnType (header|subset|sort|cardFile|template) #REQUIRED
          clientClass CDATA #IMPLIED
          classMnemonic CDATA #IMPLIED
          access (private|public|both) "public">

<!-- ListDomainClasses -->
<!ELEMENT ListDomainClassInfo EMPTY>
<!ATTLIST ListDomainClassInfo
          name CDATA #REQUIRED
          domainClass CDATA #IMPLIED
          classMnemonic CDATA #IMPLIED
          showTitles (true|false) "false"
          showActions (true|false) "false"
```

```
            showOptions (true|false) "false"
            showRelationships (true|false) "false"
            showTextInfo (true|false) "false"
            showHostJobs (true|false) "false"
            showHostReports (true|false) "false"
            displayProperties (none|keyOnly|keySeparated|mixed) "none"
            showExtendedPropertyInfo (true|false) "false">


<!-- ListCurrentEnvironment -->
<!ELEMENT ListCurrentEnvironment EMPTY>
<!ATTLIST ListCurrentEnvironment
            name CDATA #REQUIRED>


<!-- ListSessionInfo -->
<!ELEMENT ListSessionInfo EMPTY>
<!ATTLIST ListSessionInfo
            name CDATA #REQUIRED
            showLocale (true|false) "false"
            showLocalCurrency (true|false) "false">


<!-- RefreshCustomizationData -->
<!ELEMENT RefreshCustomizationData EMPTY>
<!ATTLIST RefreshCustomizationData
            name CDATA #REQUIRED>


<!-- Common elements -->
<!-- Keys and Properties - maybe values - could have exceptions as well -->
<!ELEMENT DomainEntity (Key?, Property*)>

<!ELEMENT Pql (#PCDATA)*>
<!ATTLIST Pql
            usingAliasNames (true|false) "false">

<!ELEMENT Key (Property*)>

<!ELEMENT Property (Value+|ValueRef+|NullValue+)>
<!ATTLIST Property
            path CDATA #IMPLIED
            alias CDATA #IMPLIED>

<!ELEMENT Type (#PCDATA)*>

<!ELEMENT NullValue EMPTY>

<!ELEMENT Value (#PCDATA)*>

<!ELEMENT ValueRef (#PCDATA)*>
```

SystemLinkResponse.DTD

```
<?xml version='1.0' encoding='ISO-8859-1'?>

<!ELEMENT System-Link ((Exception|LoginResponse)?,
                       (Exception|RecipientListResponse)?,
                       (Exception|Response)?,
                       (Exception|LogoutResponse)?)>
<!ATTLIST System-Link
          version CDATA #REQUIRED
          hostVersion CDATA #REQUIRED>

<!ELEMENT LoginResponse (Exception*, SessionHandle?)>
<!ATTLIST LoginResponse
          actionSucceeded (true|false) #REQUIRED>

<!ELEMENT SessionHandle EMPTY>
<!ATTLIST SessionHandle
          value CDATA #REQUIRED>

<!ELEMENT LogoutResponse (Exception?)>
<!ATTLIST LogoutResponse
          actionSucceeded (true|false) #REQUIRED>


<!ELEMENT RecipientListResponse (Exception | RecipientResponse)*>
<!ATTLIST RecipientListResponse
          sessionHandle CDATA #REQUIRED
          actionSucceeded (true|false) #REQUIRED>


<!ELEMENT RecipientResponse (Exception*)>
<!ATTLIST RecipientResponse
          recipientId CDATA #REQUIRED
          messageSent (true|false) #REQUIRED>


<!ELEMENT Response ((Exception|QueryListResponse|QueryAttachmentListResponse|
                    QueryObjectResponse|QueryAttachmentObjectResponse|
                    CreateResponse|UpdateResponse|DeleteResponse|
                    CreateAttachmentResponse|UpdateAttachmentResponse|
                    DeleteAttachmentResponse|
                    FetchNextResponse|FetchNextAttachmentsResponse|
                    FetchNextMaintenanceHistoryResponse|FetchNextTextResponse|
                    IfConditionResponse|EndIfConditionResponse|
                    StartTransactionGroupResponse|
                    ListCustomDfnsResponse|
                    ListDomainClassInfoResponse|
                    ListCurrentEnvironmentResponse|
                    ListSessionInfoResponse|
                    RefreshCustomizationDataResponse)*)>
<!ATTLIST Response
          sessionHandle CDATA #REQUIRED
          workHandle CDATA #REQUIRED>

<!ELEMENT QueryListResponse ((Exception|MetaData)?,(Exception|ColumnStatistics)?,
                             (Exception|(DomainEntity,QueryTextResponse*))*,
                             (Exception|RunHostJobResponse|
                                        CreateHostReportResponse)*)>
<!ATTLIST QueryListResponse
          name CDATA #REQUIRED
          requestedDomainClass CDATA #REQUIRED
          actionSucceeded (true|false) #REQUIRED
          moreResults (true|false) "false">

<!ELEMENT QueryAttachmentListResponse
((Exception|MetaData)?,(Exception|ColumnStatistics)?,
                                (Exception|DomainEntity)*)>
```

```
<!ATTLIST QueryAttachmentListResponse
        name CDATA #REQUIRED
        requestedOwningDomainClass CDATA #REQUIRED
        actionSucceeded (true|false) #REQUIRED
        moreResults (true|false) "false">

<!ELEMENT QueryObjectResponse ((Exception|MetaData)?,(Exception|DomainEntity?),
                                (Exception|QueryTextResponse)*,
                                (Exception|QueryMaintenanceHistoryResponse)?,
                                (Exception|RunHostJobResponse|
                                        CreateHostReportResponse)*,
                                (Exception|QueryRelationshipResponse)*) >
<!ATTLIST QueryObjectResponse
        name CDATA #REQUIRED
        requestedDomainClass CDATA #REQUIRED
        actionSucceeded (true|false) #REQUIRED>


<!ELEMENT QueryAttachmentObjectResponse ((Exception|MetaData)?,
(Exception|DomainEntity?))>
<!ATTLIST QueryAttachmentObjectResponse
        name CDATA #REQUIRED
        requestedOwningDomainClass CDATA #REQUIRED
        actionSucceeded (true|false) #REQUIRED>

<!ELEMENT FetchNextResponse ((Exception|MetaData)?,(Exception|DomainEntity))*>
<!ATTLIST FetchNextResponse
        name CDATA #REQUIRED
        queryName CDATA #REQUIRED
        actionSucceeded (true|false) #REQUIRED
        moreResults (true|false) "false">

<!ELEMENT FetchNextAttachmentsResponse ((Exception|MetaData)?,
(Exception|DomainEntity))*>
<!ATTLIST FetchNextAttachmentsResponse
        name CDATA #REQUIRED
        queryAttachmentName CDATA #REQUIRED
        actionSucceeded (true|false) #REQUIRED
        moreResults (true|false) "false">

<!ELEMENT FetchNextMaintenanceHistoryResponse ((Exception|MetaData)?,
                                                (Exception|HistoryEntry)*)>
<!ATTLIST FetchNextMaintenanceHistoryResponse
        name CDATA #REQUIRED
        queryHistoryName CDATA #REQUIRED
        actionSucceeded (true|false) #REQUIRED
        moreResults (true|false) "false">

<!ELEMENT FetchNextTextResponse ((Exception|MetaData)?,
                                (Exception|Value|TypedTextValue+)?)>
<!ATTLIST FetchNextTextResponse
        name CDATA #REQUIRED
        queryTextName CDATA #REQUIRED
        relationshipName CDATA #REQUIRED
        actionSucceeded (true|false) #REQUIRED
        moreChars (true|false) "false">

<!ELEMENT MaintainTextResponse (Exception*)>
<!ATTLIST MaintainTextResponse
        name CDATA #REQUIRED
        relationshipName CDATA #REQUIRED
        actionSucceeded (true|false) #REQUIRED>

<!ELEMENT CreateHostReportResponse (Exception*)>
<!ATTLIST CreateHostReportResponse
        name CDATA #REQUIRED
        actionSucceeded (true|false) #REQUIRED>

<!ELEMENT RunHostJobResponse (Exception*)>
```

```
<!ATTLIST RunHostJobResponse
        name CDATA #REQUIRED
        actionSucceeded (true|false) #REQUIRED>

<!ELEMENT CreateResponse ((Exception+|Key?), (Exception|MaintainTextResponse)*)>
<!ATTLIST CreateResponse
        name CDATA #REQUIRED
        actionSucceeded (true|false) #REQUIRED>

<!ELEMENT UpdateResponse ((Exception*), (Exception|MaintainTextResponse)*)>
<!ATTLIST UpdateResponse
        name CDATA #REQUIRED
        actionSucceeded (true|false) #REQUIRED>

<!ELEMENT DeleteResponse (Exception*)>
<!ATTLIST DeleteResponse
        name CDATA #REQUIRED
        actionSucceeded (true|false) #REQUIRED>

<!ELEMENT CreateAttachmentResponse (Exception*)>
<!ATTLIST CreateAttachmentResponse
        name CDATA #REQUIRED
        actionSucceeded (true|false) #REQUIRED>

<!ELEMENT UpdateAttachmentResponse (Exception*)>
<!ATTLIST UpdateAttachmentResponse
        name CDATA #REQUIRED
        actionSucceeded (true|false) #REQUIRED>

<!ELEMENT DeleteAttachmentResponse (Exception*)>
<!ATTLIST DeleteAttachmentResponse
        name CDATA #REQUIRED
        actionSucceeded (true|false) #REQUIRED>

<!ELEMENT IfConditionResponse (Exception*)>
<!ATTLIST IfConditionResponse
        name CDATA #REQUIRED
        conditionPassed (true|false) #REQUIRED>

<!ELEMENT EndIfConditionResponse (Exception*)>
<!ATTLIST EndIfConditionResponse
        ifName CDATA #REQUIRED>

<!ELEMENT StartTransactionGroupResponse (Exception+)>

<!ELEMENT ListCustomDfnsResponse (Exception|BasicCustomDfnInfo)*>
<!ATTLIST ListCustomDfnsResponse
        name CDATA #REQUIRED
        actionSucceeded (true|false) #REQUIRED>

<!ELEMENT BasicCustomDfnInfo (DfnName, PromptedProperty*)+>
<!ATTLIST BasicCustomDfnInfo
        objectId CDATA #REQUIRED
        userId CDATA #REQUIRED
        subtype CDATA #IMPLIED>

<!ELEMENT DfnName (#PCDATA)*>
<!ELEMENT PromptedProperty (OperatorType)>
<!ATTLIST PromptedProperty
        path CDATA #REQUIRED>
<!ELEMENT OperatorType (#PCDATA)*>


<!ELEMENT ListDomainClassInfoResponse (Exception|DomainClassInfo+)>
<!ATTLIST ListDomainClassInfoResponse
        name CDATA #REQUIRED
        domainClass CDATA #REQUIRED
        actionSucceeded (true|false) #REQUIRED>
```

```
<!ELEMENT DomainClassInfo (Exception|(FormTitle?, ListTitle?, ActionsAllowed?,
                                      CreateOptionInfo?, CopyOptionInfo?,
                                      KeyInfo?,
                                      (PropertyInfo|PropertyExtendedInfo)*,
                                      RelationshipInfo*,
                                      TextRelationshipInfo*,
                                      HostJobInfo*, HostProcessInfo*))>
<!ATTLIST DomainClassInfo
          domainClass CDATA #REQUIRED
          classMnemonic CDATA #REQUIRED>

<!ELEMENT FormTitle (#PCDATA)*>
<!ELEMENT ListTitle (#PCDATA)*>

<!ELEMENT ActionsAllowed EMPTY>
<!ATTLIST ActionsAllowed
          canCopy (true|false) #REQUIRED
          canCreate (true|false) #REQUIRED
          canDelete (true|false) #REQUIRED
          canDisplay (true|false) #REQUIRED
          canUpdate (true|false) #REQUIRED>


<!ELEMENT CreateOptionInfo (OptionInfo+)>

<!ELEMENT CopyOptionInfo (RelationshipOptionInfo|OptionInfo)+>

<!ELEMENT RelationshipOptionInfo (RelationshipOptionInfo|OptionInfo)*>
<!ATTLIST RelationshipOptionInfo
          relationshipName CDATA #REQUIRED
          toClass CDATA #REQUIRED
          hasCopyTemplate (true|false) #REQUIRED>

<!ELEMENT OptionInfo (Exception|OptionDescriptor+)>
<!ATTLIST OptionInfo
          optionName CDATA #REQUIRED
          optionType CDATA #REQUIRED
          constraint (boolean|list|range|string) #REQUIRED>

<!ELEMENT OptionDescriptor ((Value|NullValue), Description)>

<!ELEMENT Description (#PCDATA)*>

<!ELEMENT KeyInfo ((PropertyInfo|PropertyExtendedInfo)+)>

<!ELEMENT PropertyInfo (#PCDATA)*>
<!ATTLIST PropertyInfo
          path CDATA #REQUIRED
          format CDATA #REQUIRED
          canCreate (true|false) #REQUIRED
          canUpdate (true|false) #REQUIRED
          isNullAllowed (true|false) #REQUIRED>

<!ELEMENT PropertyExtendedInfo (Description, ConstrainedValue+)>
<!ATTLIST PropertyExtendedInfo
          path CDATA #REQUIRED
          format CDATA #REQUIRED
          constraint (boolean|list|range) #REQUIRED
          canCreate (true|false) #REQUIRED
          canUpdate (true|false) #REQUIRED
          isNullAllowed (true|false) #REQUIRED>

<!ELEMENT ConstrainedValue ((Value|NullValue), Description)>


<!ELEMENT RelationshipInfo (#PCDATA)*>
<!ATTLIST RelationshipInfo
```

```
            relationshipName CDATA #REQUIRED
            toClass CDATA #REQUIRED
            relationshipType (Many-to-1|1-to-Many) #REQUIRED>

<!ELEMENT TextRelationshipInfo (TextDescriptor+)>
<!ATTLIST TextRelationshipInfo
            relationshipName CDATA #REQUIRED
            typed (true|false) #REQUIRED>


<!ELEMENT HostJobInfo (HostProcessDescriptor,
                        (PropertyInfo|PropertyExtendedInfo)*)>
<!ATTLIST HostJobInfo
            canCreate (true|false) #REQUIRED>

<!ELEMENT HostReportInfo (HostProcessDescriptor,
                          (PropertyInfo|PropertyExtendedInfo)*)>
<!ATTLIST HostReportInfo
            canCreate (true|false) #REQUIRED>

<!ELEMENT HostProcessDescriptor (#PCDATA)*>
<!ATTLIST HostProcessDescriptor
            mnemonic CDATA #REQUIRED>

<!ELEMENT DomainEntity (Exception?, Key?, Property*)>
<!ATTLIST DomainEntity
            domainClass CDATA #REQUIRED>

<!ELEMENT ColumnStatistics (Exception?, Calculation+)>

<!ELEMENT Calculation (Exception|Value|NullValue)>
<!ATTLIST Calculation
            path CDATA #REQUIRED
            function CDATA #REQUIRED>

<!ELEMENT ListCurrentEnvironmentResponse (Exception|EnvironmentInfo+)>
<!ATTLIST ListCurrentEnvironmentResponse
            name CDATA #REQUIRED
            actionSucceeded (true|false) #REQUIRED>

<!ELEMENT EnvironmentInfo (#PCDATA)*>
<!ATTLIST EnvironmentInfo
            environmentId CDATA #REQUIRED
            hostName CDATA #REQUIRED>


<!ELEMENT ListSessionInfoResponse ((Exception|LocaleInfo)?,
                                   (Exception|LocalCurrencyInfo)?)>
<!ATTLIST ListSessionInfoResponse
            name CDATA #REQUIRED
            actionSucceeded (true|false) #REQUIRED>

<!ELEMENT LocaleInfo EMPTY>
<!ATTLIST LocaleInfo
            language CDATA #REQUIRED
            territory CDATA #REQUIRED>

<!ELEMENT LocalCurrencyInfo (#PCDATA)>
<!ATTLIST LocalCurrencyInfo
            currencyId CDATA #REQUIRED
            default (true|false) #REQUIRED>

<!ELEMENT RefreshCustomizationDataResponse (Exception)*>
<!ATTLIST RefreshCustomizationDataResponse
            name CDATA #REQUIRED
            actionSucceeded (true|false) #REQUIRED>
```

```
<!-- Keys and Properties - maybe values - could have exceptions as well -->
<!ELEMENT MetaData (Exception|Descriptor|TextDescriptor)+>

<!ELEMENT Descriptor (#PCDATA)*>
<!ATTLIST Descriptor
          type CDATA #REQUIRED
          path CDATA #REQUIRED>

<!ELEMENT TextDescriptor (Type?, Value)>
<!ATTLIST TextDescriptor
          type CDATA #REQUIRED>

<!ELEMENT Key (Property+)>

<!ELEMENT Property (Exception|Value|NullValue)+>
<!ATTLIST Property
          path CDATA #REQUIRED>

<!ELEMENT QueryRelationshipResponse ((Exception|MetaData)?,
                                     (Exception|ColumnStatistics)?,
                                     (Exception|
                                      (DomainEntity,QueryTextResponse*,
                                       QueryRelationshipResponse*)*))>
<!ATTLIST QueryRelationshipResponse
          name CDATA #REQUIRED
          relationshipName CDATA #REQUIRED
          actionSucceeded (true|false) #REQUIRED
          moreResults (true|false) "false">

<!ELEMENT QueryTextResponse ((Exception|MetaData)?,
                             (Exception|Value|TypedTextValue+)?)>
<!ATTLIST QueryTextResponse
          name CDATA #REQUIRED
          relationshipName CDATA #REQUIRED
          actionSucceeded (true|false) #REQUIRED
          moreChars (true|false) "false">


<!ELEMENT TypedTextValue (Type, (Exception|Value))+>

<!ELEMENT Type (#PCDATA)*>

<!ELEMENT Value (#PCDATA)*>
<!ATTLIST Value
          moreChars (true|false) "false">

<!ELEMENT QueryMaintenanceHistoryResponse ((Exception|MetaData)?,
                                           (Exception|HistoryEntry)*)>
<!ATTLIST QueryMaintenanceHistoryResponse
          name CDATA #REQUIRED
          actionSucceeded (true|false) #REQUIRED
          moreResults (true|false) "false">

<!ELEMENT HistoryEntry ((Exception?, Property*, ChangedDomainValues?))>

<!ELEMENT ChangedDomainValues (Exception|ChangedProperty*)>
<!ELEMENT ChangedDomainValues
          domainClass CDATA #REQUIRED>

<!ELEMENT ChangedProperty (Description?, ValueBefore?, ValueAfter?)>
<!ATTLIST ChangedProperty
          path CDATA #REQUIRED>

<!ELEMENT ValueBefore (#PCDATA)*>
<!ATTLIST ValueBefore
          valid (true|false) "true"
          isNull (true|false) "false">
```

```
<!ELEMENT ValueAfter (#PCDATA)*>
<!ATTLIST ValueAfter
          valid (true|false) "true"
          isNull (true|false) "false">

<!ELEMENT NullValue EMPTY>

<!ELEMENT Exception (Text, Message*)>
<!ATTLIST Exception
          name CDATA #REQUIRED>

<!ELEMENT Message (Text, DetailedText?)>
<!ATTLIST Message
          type (error|warning|information) #REQUIRED>

<!ELEMENT Text (#PCDATA)*>
<!ELEMENT DetailedText (#PCDATA)*>
```

Appendix A. Supported objects

Information regarding the domain classes supported by System-Link can be obtained dynamically via the Utility action "ListDomainClassInfo". Example: to display a System-Link response listing all domain classes supported, the following action would be used:

<ListDomainClassInfo name='listAllClasses' domainClass='*all'/>

For further information on obtaining domain class lists, as well as determining each class' supported actions and properties, please see ListDomainClassInfo in XML Syntax for requests and responses.

Please note the following: If you wish to use System-Link with Integrator objects or XA primary objects that have been extended with Integrator, you must have Enterprise Integrator installed. These objects will not be usable through System-Link with Standard Integrator. In the case of extended XA primary objects, this means that the entire object (not just its extension) will be unavailable through System-Link without installing Enterprise Integrator.