Auction™

Integration Guide
Version 5.0

January 2002
Publication: January 24, 2002 4:03

Commerce One
Corporate Headquarters
4440 Rosewood Drive
Pleasanton CA 94588-3050

www.commerceone.com

Auction Integration Guide, Version 5.0

Copyright © 2002 Commerce One, Inc. All rights reserved.

March 2002

Contents

Preface

Purpose of this Guide

This book describes the Auction 5.0 integration scenarios, components, and processes.

Note: See *Auction Installation and Administration Guide* for the step-by-step process of setting up the integrations described in this book.

Audience

This book is intended for system integrators and global services personnel who are responsible for integrating the Auction application with other Commerce One products. Expert-level knowledge of the following is required:

- Auction Services platform
- ActiveMarket platform
- xCBL (XML Common Business Language)
- XML data structure

In addition, you must expert-level knowledge of the application(s) you intend to integrate or use with the Auction application.

How to Use this Guide

This guide is divided into the following sections:

Chapter	Description
Chapter 1, Overview	An overview of Auction integration concepts and the primary Auction integration component: iHub.
Chapter 2, XML Portal Connector Integration	A technical description of the Auction/XPC integration.
Chapter 3, eMarketPlace Catalog Integration	A technical description of the Auction/eMPC integration.
Chapter 4, Enterprise Buyer Desktop Integration	A technical description of the Auction/EBD integration.
Chapter 5, Transaction Event Collector Integration	A technical description of the Auction/TEC integration.
Appendix A, Error Messages	Definitions of the error messages you may encounter when you perform the integrations described in this guide, as well as possible remedies.
Appendix B, Response Codes	Definitions of the response codes you may encounter when you perform the integrations described in this guide.
Appendix C, Utility Pages	Descriptions of the various utility pages that ship with the Auction application, and are useful when performing the integrations described in this guide.
Appendix D, Sample Data	Examples of Auction-related xCBL Data.

In addition, this guide ships with a separate Excel file called *XCBLProfile.xls*. The *XCBLProfile.xls* indicates which xCBL fields are supported and which are unsupported at this time, and provides comments when necessary.

Related Information

The following documents contain related information:

- *Auction Users Guide*
- *Auction Deployment Guide*
- *Auction Installation and Configuration Guide*
- *eMPC Installation and Administration Guide*
- *Enterprise Buyer Desktop Installation Guide*
- *Enterprise Buyer Desktop Administration Guide*
- *MarketSite Installation Guide*
- *MarketSite Administration Guide*
- *Trading Partner XML Portal Connector Administration Guide*

If You Need Help

Each Commerce One installation with a support contract has one or more persons designated as your technical support. If you cannot resolve a problem by using the Commerce One manuals or on-line help, ask the designated person to contact Technical Support via email:

`csc@commerceone.com`

Please have your Support Company ID Number if possible when you contact Technical Support.

Send Us Your Comments

We value your feedback about our documentation. If you would like to make comments or suggestions about this or other Commerce One documentation, please send email to:

`techpubs@commerceone.com`

So that we can make the best use of your comments, please include the following:

- The exact title of the document. For example, *Enterprise Buyer Desktop Edition Administration Guide*.
- The exact version number, shown on the cover of the document. For example, "Version 2.0, Service Pack 1"
- The date the document was released. This is shown either on the copyright or title pages.
- The exact chapter or section title you want to make a comment about. You may include page numbers, but please also include section titles.
- Your comments.

Thank you.

1 Overview

This chapter introduces the architecture of the Auction integration, describing its objectives, components, and features.

Architecture

The Auction integration architecture addresses the business need of accessing Auction functionality from other Commerce One products, and vice versa. The fundamental capability required of integration is the seamless execution of business activities (or processes) that span multiple system components.

For example, a process spanning the Auction, eMPC, and Enterprise Buyer products is as follows:

1. A buyer creates a reverse auction using the eMPC catalog.
2. Suppliers bid on the items being auctioned, the auction eventually closes, and winners are selected.
3. The buyer creates a purchase order in Enterprise Buyer for the auction winners.

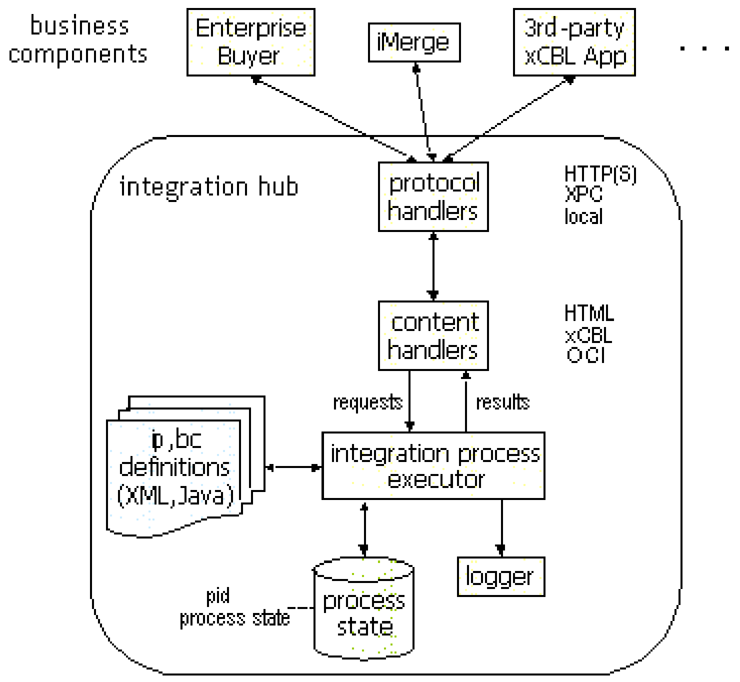
The integration architecture is driven by the following enterprise-level attributes:

- Modularity — A clear separation exists between the integration logic (i.e., the business-level steps that need to be executed) and protocol and content handlers.
- Extensibility — The framework provides support for easily adding new protocol and content handlers, as well as adding new integration programs.
- Scalability — The process execution engine is designed to run in a clustered environment, with all necessary states stored in a database. Each incoming request is handled in a separate thread, and is agnostic about which element of a cluster it is running on.
- Traceability — The steps of the integration process are recorded, providing a simple audit mechanism.
- Manageability — The process execution engine has a built-in suspend/restart capability. It is possible to add a mechanism to alter the execution path of a suspended process.
- Availability — Requests are handled independent of each other, providing a first-level of failure isolation. The execution engine supports hot addition/ swap of cluster elements. It is possible to add hot version upgrade of integration processes.

An extensible and modular integration framework is achieved by approaching the integration as the execution of integration-related processes. Besides providing this primary capability, the integration framework handles different protocols (such as HTTP) and content formats (such as OCI and xCBL), and provides services such as logging, security and transactions.

Figure 1-1 illustrates the high-level structure of the integration hub, also called iHub.

Figure 1-1 High-level view of the Integration Hub.



The heart of the hub is an execution engine that executes integration programs. Supporting this engine are:

- Various protocol handlers that communicate with external components.
- Content handlers that convert various content formats into a standard internal form. (Commerce One has chosen XML because it is a superset of the formats that are currently handled.)
- A logging mechanism that can be used to audit all communication through the hub.

The execution engine is capable of asynchronous program execution. For this purpose, it uses a database to store the state of suspended programs. The database is also used as a synchronization mechanism when the hub is deployed in a clustered environment.

2 XML Portal Connector Integration

This chapter describes the functionality of the Auction/XPC (XML Portal Connector) integration. The purpose of the integration is to create auctions using the XCBL 3.5 AuctionEventCreate document, and receive auction result information when winners are determined at the end of the bidding process.

Integration Overview

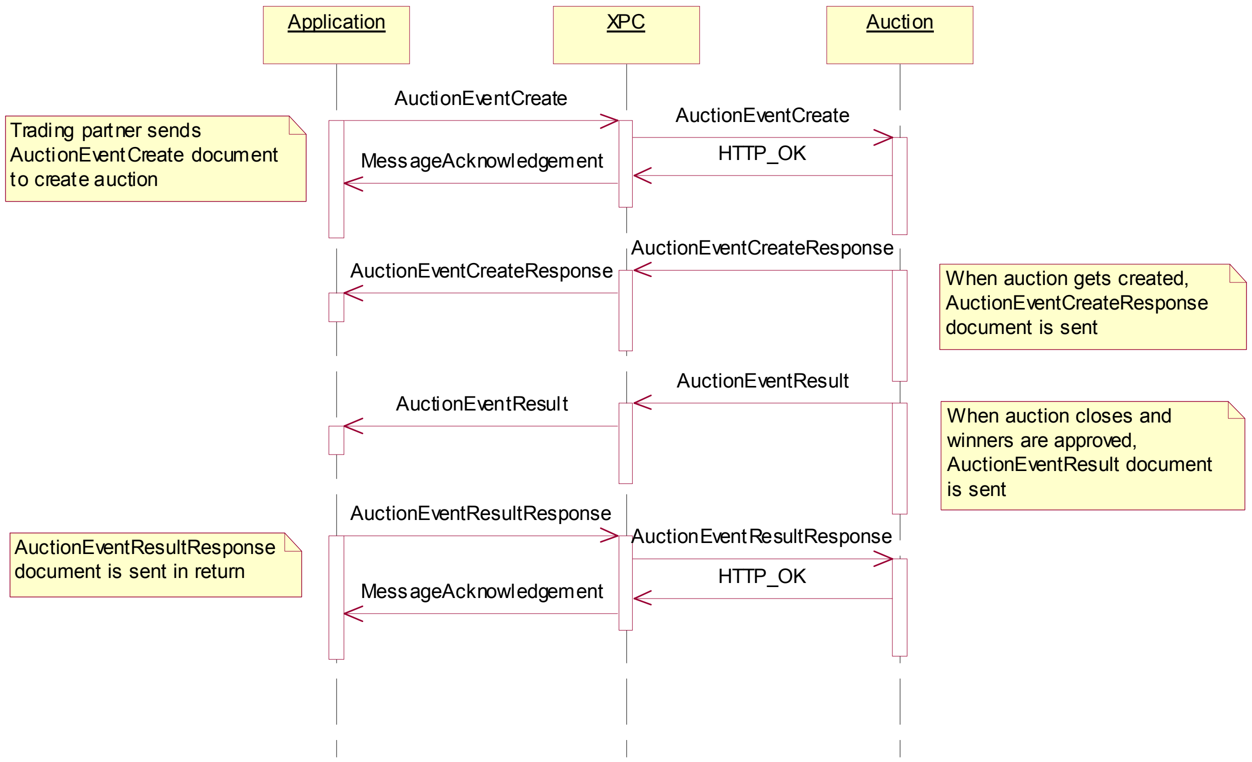
The Auction/XPC integration consists of two integration programs: one for handling the auction event creation and a second for handling the auction result.

The first program starts when iHub receives an envelope containing an AuctionEventCreate document. After checking the credentials of the sender, the program processes the document and creates the auction event according to the information specified. It then generates a response document, which it sends back to the sender and other interested parties to acknowledge the auction event creation if the response is desired.

The second program starts after the first program successfully creates the auction. For each auction lot created, one program is started to handle the corresponding result. The program is suspended until the auction lot is closed and all winners for the auction lot are determined and approved. When the program resumes, it retrieves the winner information, generates the corresponding auction result document, and sends the result document to the originator and the list of interested parties as specified in the AuctionEventCreate document. If the AuctionResultResponse document is expected, the program is once again suspended. When the corresponding response document is received, it processes the document and saves it in the database.

Figure 2-2 shows the flow diagram for the XPC integration.

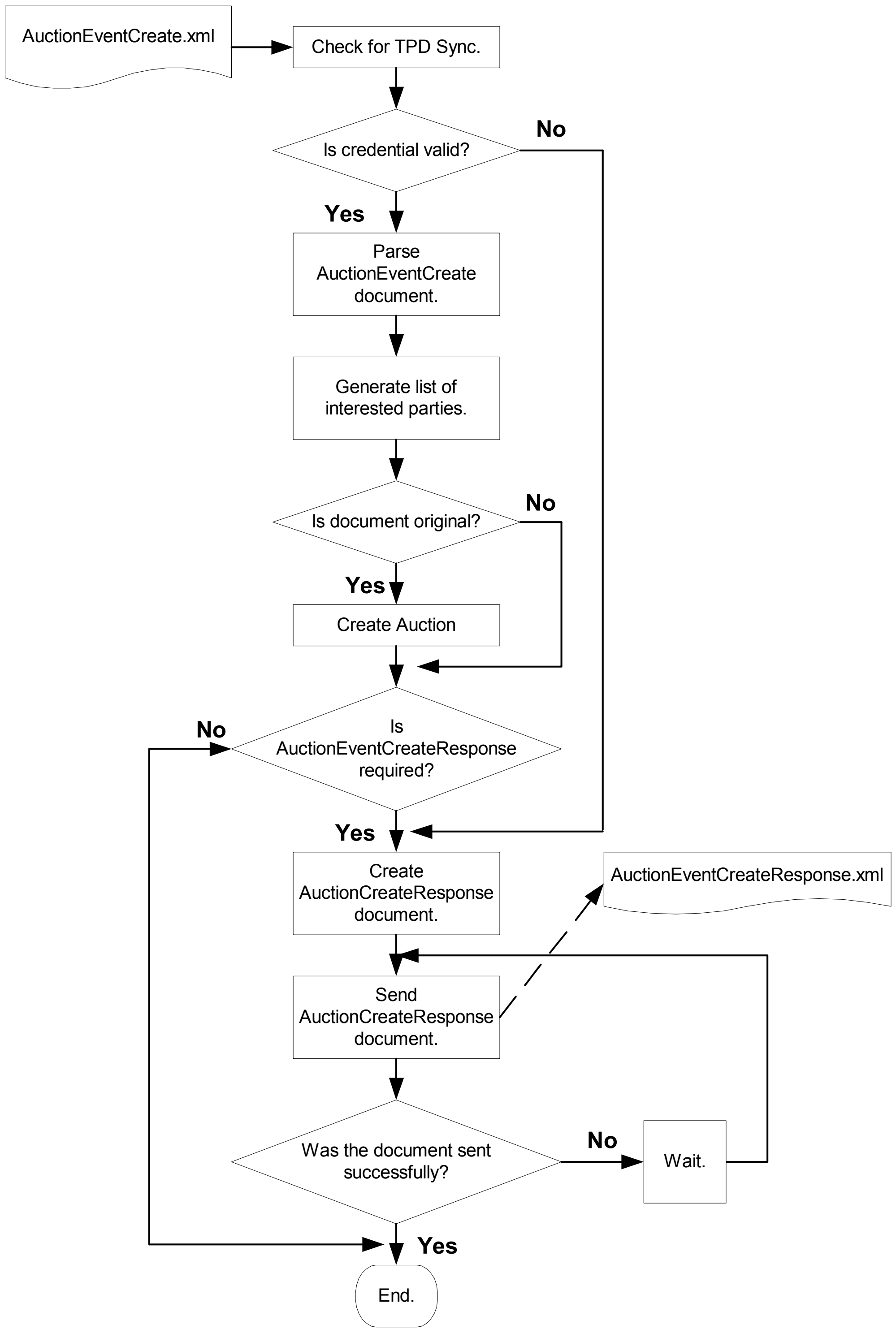
Figure 2-2 XPC/Auction Integration Flow Diagram


Application
XPC
Auction
AuctionEventCreate
AuctionEventCreate
Trading partner sends AuctionEventCreate document to create auction
HTTP_OK
MessageAcknowledgement
AuctionEventCreateResponse
AuctionEventCreateResponse
When auction gets created, AuctionEventCreateResponse document is sent
AuctionEventResult
AuctionEventResult
When auction closes and winners are approved, AuctionEventResult document is sent
AuctionEventResultResponse
AuctionEventResultResponse
AuctionEventResultResponse document is sent in return
HTTP_OK
MessageAcknowledgement

Auction Event Create Integration Program

Figure 2-3 shows the flow diagram for the integration program that creates the auction event from the specific AuctionEventCreate xCBL document. Following the diagram are explanations of the processes depicted in the diagram.

Figure 2-3 Integration Program for Auction Event Creation


AuctionEventCreate.xml
Check for TPD Sync.
Is credential valid?
No
Yes
Parse AuctionEventCreate document.
Generate list of interested parties.
Is document original?
No
Yes
Create Auction
No
Is AuctionEventCreateResponse required?
Yes
Create AuctionCreateResponse document.
AuctionEventCreateResponse.xml
Send AuctionCreateResponse document.
Was the document sent successfully?
No
Wait.
Yes
End.

- Check for TPD Sync — The TPID and username are extracted from a credential attached to the incoming envelope. The Auction database is checked to verify the existence of a company with that TPID, as well as the existence of a user (loginID) with that name in that company. Then, the name of the company unit for which the auction is to be created is extracted from the AuctionEventCreate document in the envelope, and a company unit by that name is identified in the Auction database. Finally, the identified user's priviledge to create an auction for this company unit is verified. See Authorization on page 2-11 for information on how this priviledge verification happens.

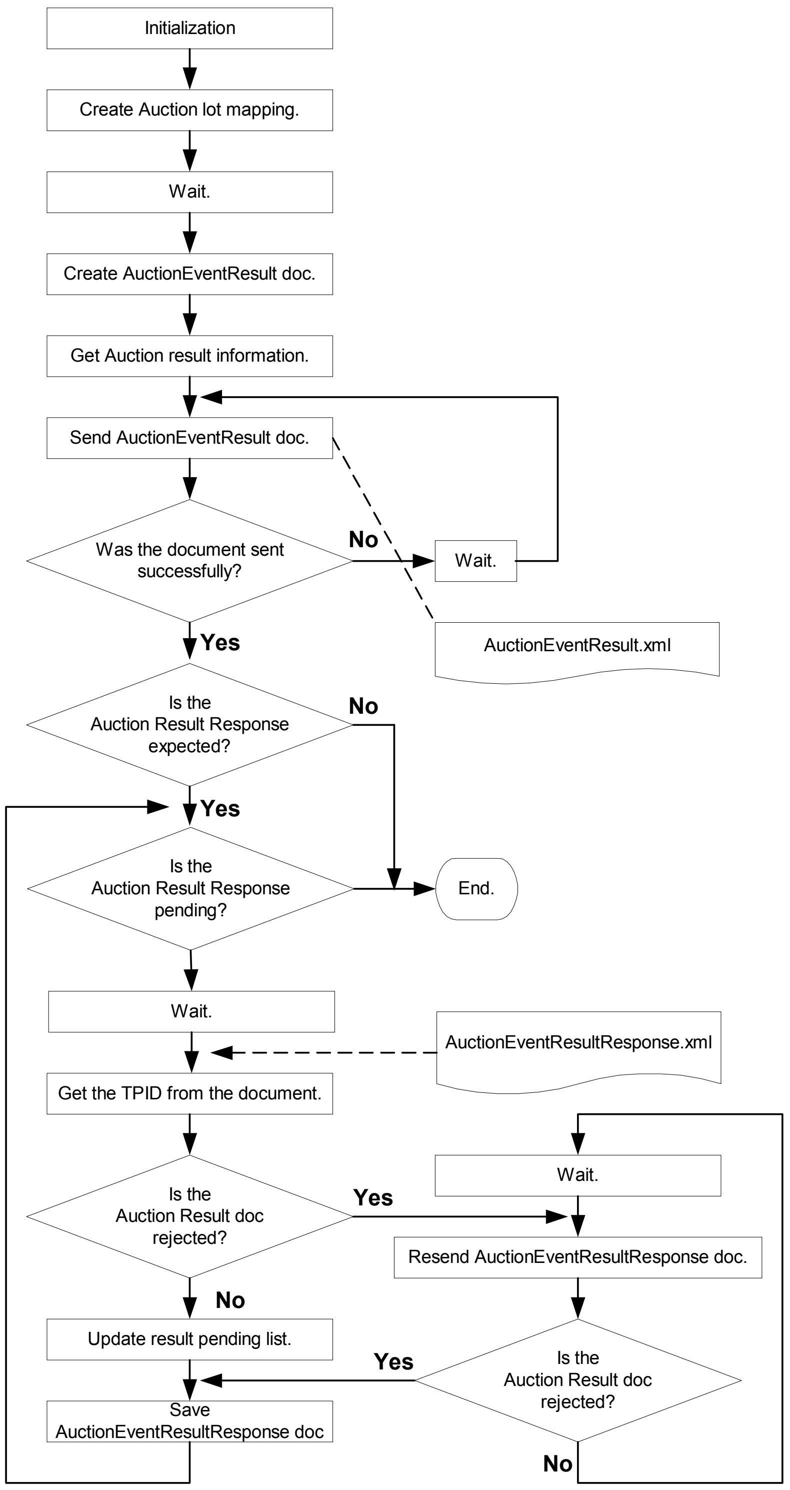
Note: If the Trading Partner Syncronization process has not taken place, the TPID of the sender's company is not known to the Auction database, resulting in the failure of this process. If failure occurs, the TPDSync.jsp utility page can be used to set the TPID for a company in the Auction database. For more information on the TPDSync.jsp utility page, see TPDSync.jsp on page C-3.

- Parse AuctionEventCreate document — Parses the incoming document and retrieves all the information about the auction event to be created.
- Generate list of interested parties — Generates the list of people to whom the AuctionEventCreateResponse document should be sent, as specified in the AuctionEventCreate document.
- Create auction — If the purpose code of the AuctionEventCreate document is Original, an auction is created from the corresponding information specified in the document. If the auction is created successfully, a process is forked for each auction lot created. This acts as an independent integration program that handles the result for that auction lot.
- Create AuctionEventCreateResponse Document — If a response is expected for the auction creation, the corresponding auction event create response document is generated, specifying the status of the auction creation. The appropriate response codes for each auction item, lot, and event are also set accordingly.
- Send AuctionEventCreateResponse Document — After the response document is created, it is sent to each interested party, with the locale for the document set to locale specified for the particular document receiver. If there is an error message generated during the execution of the integration program, the error message, if applicable, is also translated to the proper locale before setting it in the response document. If it fails to transmit the response document, the program is suspended and waits for a resume request to resend the response document. You can use the ResendDocument.jsp page to manually resend the document. For more information, see ResendDocument.jsp on page C-5.

Auction Event Result Integration Program

Figure 2-4 shows the flow diagram for the integration program that retrieves the auction result and sends the corresponding auction result document to the interested parties. Following the diagram are explanations of the processes depicted in the diagram.

Figure 2-4 Integration Program for Auction Event Result


Initialization
Create Auction lot mapping.
Wait.
Create AuctionEventResult doc.
Get Auction result information.
Send AuctionEventResult doc.
Was the document sent successfully?
No
Wait.
AuctionEventResult.xml
Yes
Is the Auction Result Response expected?
No
Yes
Is the Auction Result Response pending?
End.
Wait.
AuctionEventResultResponse.xml
Get the TPID from the document.
Is the Auction Result doc rejected?
Yes
Wait.
Resend AuctionEventResultResponse doc.
No
Update result pending list.
Yes
Is the Auction Result doc rejected?
Save AuctionEventResultResponse doc
No

- Initialization and create auction lot mapping — Once the previous program successfully creates the auction lot, the program initializes with the appropriate data, and creates the mapping between the auction lot and the process ID. This way, the correct program is resumed when the result of the auction lot is received.
- Create AuctionEventResult document — When the auction lot closes and all winners for the auction lot are approved, an AuctionEventResult document is generated with the appropriate auction lot information being set.
- Get Auction result information — The auction item winner information and the bid details are retrieved and populated into the auction event result document.
- Send AuctionEventResult document — The auction event result document is sent to the auction originator and the list of interested parties specified in the AuctionEventCreate document. If it fails to transmit the result document, the program suspends and waits for a resume request to resend the result document. You can use the ResendDocument.jsp page to manually resend the document. For more information, see ResendDocument.jsp on page C-5.
- Wait for AuctionEventResultResponse document — If the organization is configured to accept AuctionEventResultResponse documents, the program goes into the wait state awaiting the corresponding result response documents.
- Get the TPID from AuctionEventResultResponse document — The TPID of the response document sender is retrieved from the reference in the AuctionEventResponse document to identify the document sender.
- Resend the AuctionEventResultResponse document — The auction result response code specified in the response document is retrieved. If the response code is RejectedResend, the AuctionEventResult document is resent to the response document sender. If it fails to transmit the result document, the program suspends and waits for a resume request to resend the result document. You can use the ResendDocument.jsp page to manually resend the document. For more information, see ResendDocument.jsp on page C-5.
- Update result pending list — If the response code is Accepted or RejectedDoNotRend, the corresponding trading partner is done. The result pending list is updated to indicate that one less response document is expected.
- Save AuctionEventResulResponse Document — The response document is archived and can be looked up using the QueryData.jsp utility page. See QueryData.jsp on page C-2 for more information.

Business Components

The Auction/XPC integration utilizes two business components: the Client Business Component and the Auction Business Component.

The Client Business Component provides services for the integration program to interact with the XPC server. It sends the outgoing xCBL documents to the receiver(s) at the desired destination.

The Auction Business Component provides services for the integration program to interact with the Auction API. It does the following:

1. Authenticates the User.
2. Gets different data objects used for the auction.
3. Create the auction.
4. Gets the bidders and auction winner information.

Exception Handling

There are two types of exceptions that can occur during the execution of the integration process: business exceptions and process exceptions.

Business Exceptions

Business exceptions are caused by invalid data specified in the document. When business exceptions occur, the auction lot involved is not created. When returning the response document to the user, the corresponding response code is set in the document that tells the user about the error that occurred.

Process Exceptions

Process exceptions are caused by internal failures (such as a database failure), or when the document cannot be parsed. When process exceptions occur, the integration process stops, and an ErrorInfo document containing the error information is generated and sent back to the document sender.

Authentication and Authorization

The creation of auctions in the Auction application requires appropriate privileges. In GUI-based interaction with the auction application, these privileges are checked when a user logs in. For auction creation via xCBL documents, a similar check is applied.

Authentication

Every xCBL envelope that is routed through a MarketSite/portal-router includes a credential document that contains the information about the user or trading-partner that sent the document. Typically, this corresponds to the system account that is created when a trading partner registers with a MarketSite. The system account information is contained in the <XPCROOT>/bin/client.prop file. (For information about XPC, see *XPC Developers Guide*.)

When an envelope containing an AuctionEventCreate xCBL document arrives at the Auction application, the credential information is used to find the corresponding company and user in that company. If the credential information is missing, or if the corresponding company/user cannot be found, the envelope is rejected. This process is similar to a single-sign-on setup, in which the sender authenticates with a MarketSite, the receiving TP-XPC trusts the MarketSite, and the Auction application trusts the TP-XPC that is associated with it. Both of these trusts are based on transport-level authentication via user ID/password or certificates.

The Auction application relies on the Trading Partner Synchronization process between a MarketSite and the ActiveMarket Platform underneath the Auction application to create companies and users when a trading partner registers with the MarketSite. It is also possible to manually configure TPID for a company using a the TPDSync.jsp utility page. See TPDSync.jsp on page C-3 for more information.

Authorization

In order for auction creation via xCBL document to succeed, the user specified in the credential document must have the auction creation priviledge for the company unit specified in the InitiatingParty section of the AuctionEventCreate document. The auction creation priviledge is granted by assigning a job function named “Auction Originator” to this user. The Trading Partner Synchronization service configures the appropriate priviledges (job functions) for the user. A user corresponding to the TP-XPC system account can also be manually created and assigned priviledges using the ActiveMarket user interface.

The user's priviledge to create auction for a specific company unit is decided in this way:

- Super user — Can create an auction for any company unit in any company.
- Company admin user — Can create auction for any company unit in his/her company.
- Company unit admin user — Can create an auction for the following:
  - A company unit to which he/she belongs.
  - Company units that he/she manages, and all its children company units.
  - Other company units to which he/she is assigned.

All other users can create auctions for the company unit(s) to which they belong, or any company units to which they are assigned.

Registering Auctions with MarketSite

Note: Prior to registering the Auction application with MarketSite, install both the ActiveMarket and Auction applications, as well as TP XPC for the Auction application. Refer to *ActiveMarket 3.3 System Administration Guide* for information on how to use user interface for adding trading partners, services, privileges and roles in MarketSite for companies and users.

To register the Auction application with MarketSite, do the following:

1. Register Auction TP XPC as a trading partner with MarketSite as a trading partner. As part of the registration process, assign the Auction TP XPC an TPID.
2. Create a Document Destination for this trading partner, with the DDID of this document destination being the same as the TPID of the Auction TP XPC. Make a note of this DDID for use in the following steps.

Defining services in MarketSite

After you install the Auction application, service offerings listed on the right-hand side exist in ActiveMarket. Create the services listed on the left-hand side in MarketSite such that they map to the corresponding ActiveMarket services. Mapping is done based by matching DDID on both sides. Set the DDID for all services to the TPID of the Auction TP XPC mentioned above.

MarketSite Service	ActiveMarket Service Offering
Company Catalog Service	Company Catalog Offering
Auction Organization Service	Auction Organization Offering
Auction Report Service	Auction Report Offering
Auction Originator Service	Auction Originator Offering
Auction Bid Service	Auction Bid Offering
Auction Observer Service	Auction Observer Offering
Auction Guest Service	Auction Guest Offering

For example, if you are defining one service in MarketSite called “Company Catalog Service” and you intend to map it to a service offering called “Company Catalog Offering” in ActiveMarket, do the following:

1. Create the “Company Catalog Service.” Make it both UI-based and Document-based, make Auction trading partner the hosting organization of the service, and assign the DDID to this service. (The DDID is same as the TPID of the Auction TP XPC.)
2. For the “Company Catalog Offering” service offering in ActiveMarket, specify the same DDID. This completes the mapping.

Defining Privileges in MarketSite

After you define these Auctions-related services in MarketSite, add privileges for each of them. Each privilege has a one-to-one match with the job functions defined in ActiveMarket. Make sure the privilege name exactly matches the job function name as shown here.

Privilege	**Job Function Name**
Privileges to be defined for Company Catalog Service.	Company Catalog Administrator
Privileges to be defined for Auction Organization Service.	Auction Organization Administrator
Privileges to be defined for Auction Report Service.	■ Organization Administrator Reports ■ Originator Reports ■ Bidder Reports
Privileges to be defined for Auction Originator Service	Auction Originator
Privileges to be defined for Auction Bid Service	Auction Bidder
Privileges to be defined for Auction Observer Service	Auction Observer
Privileges to be defined for Auction Guest Service	Auction Guest

Defining roles in MarketSite

Once a service is set up on MarketSite, MarketSite trading partners are able to subscribe to it. Once a subscription is approved, that trading partner has access to the privileges associated with the subscribed service. The trading partner administrator can then create roles in MarketSite using these privileges and assign these roles to different users.

3 eMarketPlace Catalog Integration

This chapter describes the functionality of the Auction/eMarketPlace Catalog (eMPC) integration. The objective of this integration is to allow users to obtain auction items from an eMPC catalog, and create the corresponding auction line items in the Auction Wizard automatically.

Integration Overview

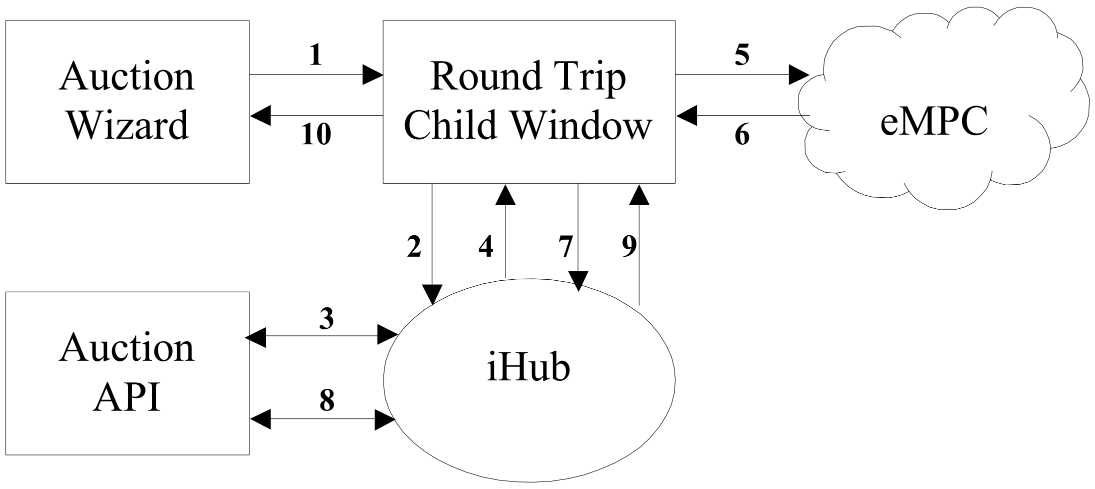
The eMPC round trip consists of two major steps: CertBegin and CertRecv.

CertBegin starts when the user clicks the button from the Auction Wizard to get line items from the eMPC catalog. A new child window opens for the round trip and redirects the user to the eMPC catalog. Once in the eMPC catalog, the user adds items to their "shopping cart." When the user has selected all of the items they desire, they "checks out" and the items they selected are posted to iHub as OCI data.

The integration program resumes with CertRecv, which creates the corresponding line items using the OCI data retrieved. The status is displayed to the user and the control is returned to the Auction Wizard.

Figure 3-5 illustrates the flow for the Auction/eMPC integration.

Figure 3-5 Auction/eMPC Integration Flow Diagram



CertBegin

1. A new child window opens for the Round Trip.
2. iHub intercepts and calls the IP CertBegin.
3. The eMPC URL is obtained by Org using the Auction API.
4. iHub redirects the user to eMPC.
5. The user selects items from the eMPC catalog to add to their “shopping cart.”
6. The user “checks out” after completing their selection.
7. eMPC posts a list of the selected items to iHub via OCI.

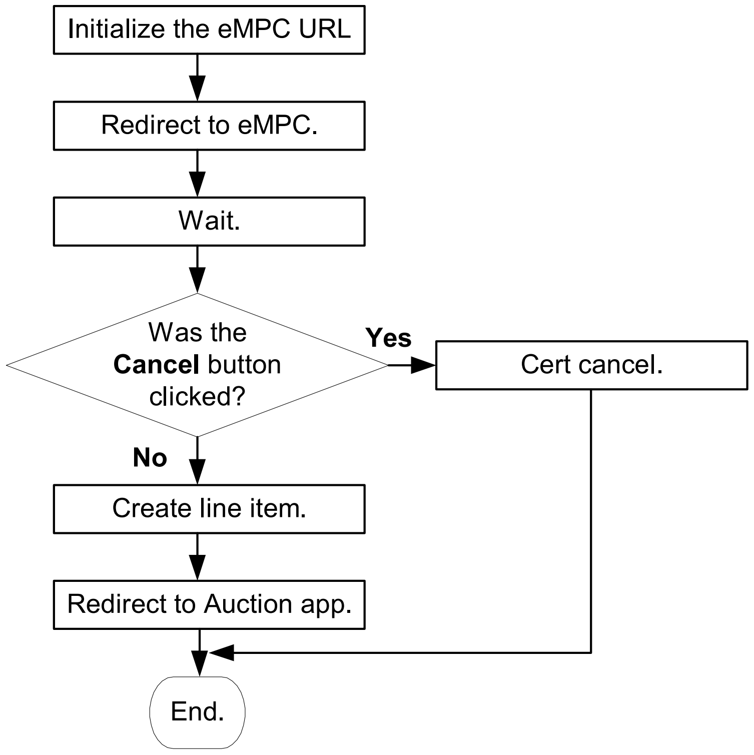
CertRecv

8. The posted items are added to the Auction application using the Auction API.
9. The summary is displays.
10. The child window closes and the user returns to the Auction Wizard.

Integration Program

Figure 3-6 shows the flow diagram for the integration program that is executed during the Auction to eMPC round trip to import auction line items. Following the diagram are explanations of the processes depicted in the diagram.

Figure 3-6 Integration Program for eMPC Round Trip



- Initialize eMPC URL — A create request comes in to import line items from eMPC and a new child browser is opened. Auction and user information are retrieved from the request, and the eMPC URL is computed from the user/org preferences.
- Redirect to eMPC — Redirects the user to the eMPC catalog using the computed URL, specifying "hook_url" as self.
- Wait for eMPC result — The integration program suspends while the user selects items from the eMPC catalog to add to the "shopping cart." When the user "checks out," the results are posted to iHub as OCI data and the integration program resumes.
- Create line item — Retrieves the OCI data incrementally, with the OCI fields being mapped to the relating auction item attributes. The corresponding auction line items are created and inserted into the Auction Item table.
- Redirect to Auction — Displays the status for importing the line item to the user. The child window closes and the user returns to the Auction Wizard.
- Cancellation by user — The user can cancel the line item importing process by clicking the Cancel button on the child window. When this happens, no auction item is created, the database is cleaned up, and the child window closes.

Business Components

Two Business Components are provided for the integration program to interact with other applications: the Client Business Component and the Auction Business Component.

Client Business Component

The Client Business Component provides a mechanism to interact with the child browser. It is used to:

1. Obtain the iHub URL for building the hook URL.
2. Redirect the browser to a different JSP (such as the eMPC page)
3. Forward status information from the integration program to the corresponding JSP.

Auction Business Component

The Auction Business Component provides services for the integration program to interact with the Auction API. It is used to:

1. Get the eMPC URL using the Org API.
2. Create and insert line items into the auction item table using the Auction Management Entity framework.

Exception Handling

There are two types of conditions that can interfere with the normal execution of the integration process: business exceptions and the iHub exceptions.

Business Exceptions

Business exceptions are caused by user error such as supplying invalid data value or data type. When business exceptions occur, the line item involved is not created. When returning the result to the user, the appropriate error message displays, and the status is shown, which tells the user the number of line items that were added successfully and which line items failed to be added.

iHub Exceptions

iHub exceptions are caused by internal failures such as a database failure. When iHub exceptions occur, the integration process stops, the error is logged, and an error page is returned to the user with the appropriate error message.

4 Enterprise Buyer Desktop Integration

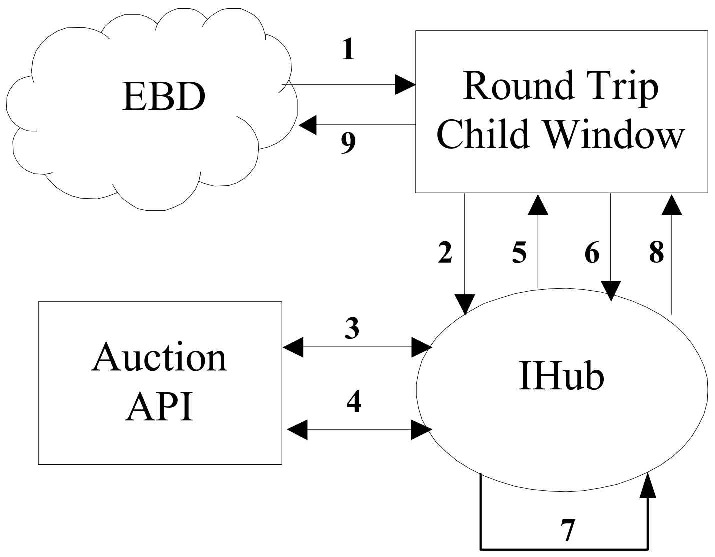
This chapter describes the functionality of the Auction/Enterprise Buyer Desktop (EBD) integration. The objective of the integration is to enable users to create a Purchase Order (PO) in EBD by pulling a list of winners from the Auction application.

Integration Overview

The EBD round trip involves getting an auction winner list and posting the selected winners to EBD. When the EBD user sends a request to get the auction winners, the integration program looks up all the winners who have not been previously selected for all the reverse auctions owned by the originator specified in the request, and displays the winners on the new round trip child window. The user can then select the desired winners and create purchase orders for the selected winners of the auction.

Figure 4-7 illustrates the flow for the EBD integration.

Figure 4-7 Auction/EBD Integration Flow Diagram



Get Winner List

1. A new child window opens for the Round Trip.
2. The Round Trip process begins.
3. iHub accesses the Auction API to authenticate the administrator and retrieve user information.
4. iHub retrieves a list of winners from the Auction API.
5. The winners are displayed to the user.

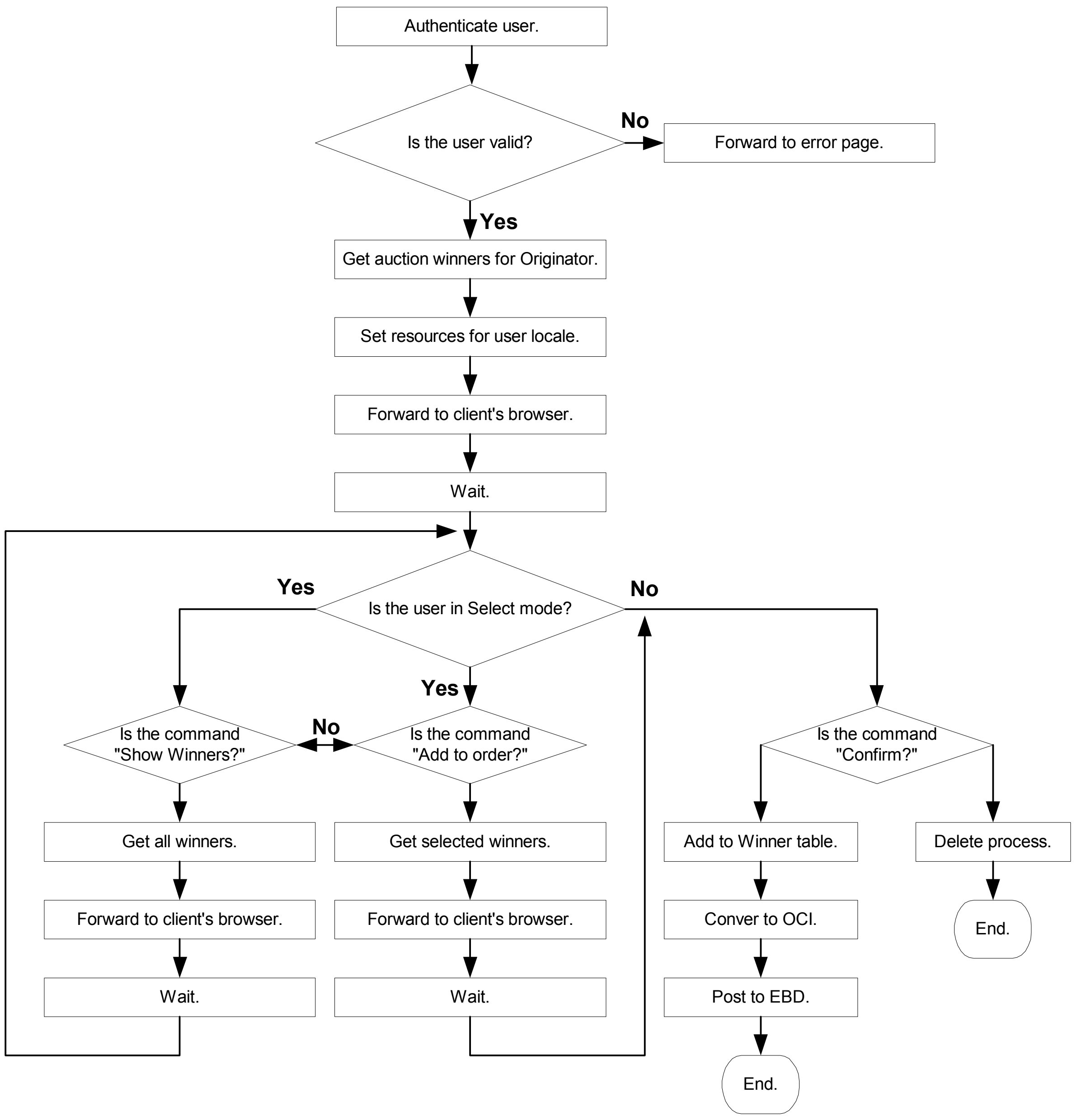
Select Winners

6. The user selects the winners to add to a requisition.
7. The newly selected winners are added to the iHub winner table.
8. iHub converts the data to OCI format and posts the results to the EBD application.
9. The child window closes and the user is returned to EBD.

Integration Program

Figure 4-8 shows the flow diagram for the integration program that is executed during the Auction to EBD integration for retrieving auction winners. Following the diagram are explanations of the processes depicted in the diagram.

Figure 4-8 Integration Program for EBD Round Trip


Authenticate user.
Is the user valid?
No
Forward to error page.
Yes
Get auction winners for Originator.
Set resources for user locale.
Forward to client's browser.
Wait.
Yes
Is the user in Select mode?
No
Yes
Is the command "Show Winners?"
No
Is the command "Add to order?"
Is the command "Confirm?"
Get all winners.
Get selected winners.
Add to Winner table.
Delete process.
Forward to client's browser.
Forward to client's browser.
Conver to OCI.
End.
Wait.
Wait.
Post to EBD.
End.

- Authenticate user — Authenticates the ‘Admin’ user using the credentials specified in the request. Gets the originator information using the Admin Session object.
- Get Auction Winners for Originator — Retrieves the list of auction winners who have not been previously selected in EBD for all the reverse auctions visible to the originator specified.
- Forward to browser client — The retrieved list of auction winners is forwarded to the child browser window. Each winner will be displayed as a row in a table shown together with the corresponding auction item information with a checkbox in which the user can check to select the winner to create purchase order.
- Is user in Select mode? — Checks if the user is in the process of selecting the winners, in which a command of type ‘Show All Winners’, ‘Show Unselected’ or ‘Add To Order’ is issued by clicking the corresponding action button.
- Get All Winners — Retrieves the list of all auction winners for all the reverse auctions visible to the originator specified. This includes those winners who have been previously selected in EBD.
- Get Unselected Winners — This is equivalent to the first step of the integration program that retrieves the list of auction winners for all the reverse auctions visible to the originator, excluding those winners who have been previously selected in EBD.
- Add to iHub Auction Winner Table — When the user confirms the winner selection, the integration program updates the iHub table with the list of winners selected, so that the next time the user chooses to retrieve all unselected winners, these winners are not shown.
- Convert to OCI — Sets a reference in the State object to the Outbound OCI converter. This reference is used by the JSP page to post the OCI elements to EBD.
- Post to EBD — The data for the selected winners are sent back to EBD where purchase orders are created.
- Delete process — The user can cancel the supplier information importing process by clicking the cancel button on the child browser. When this happens, the iHub auction winner table will not be updated. The process for this EBD roundtrip is deleted and the child browser closes.
- Forward to Error Page — If authentication for the user failed, an error page displays on the child browser showing the appropriate error message.

Business Components

Two Business Components are provided for the integration program to interact with other applications: the Client Business Component and the Auction Business Component.

Client Business Component

The Client Business Component provides services for the integration program to interact with the Client Window. It is used to:

1. Display the list of winners retrieved from Auction.
2. Display the error message for any error that has occurred.

Auction Business Component

The Auction Business Component provides services for the integration program to interact with the Auction API. It is used to:

1. Authenticate the User.
2. Retrieve the list of All Winners/Unselected Winners.

Exception Handling

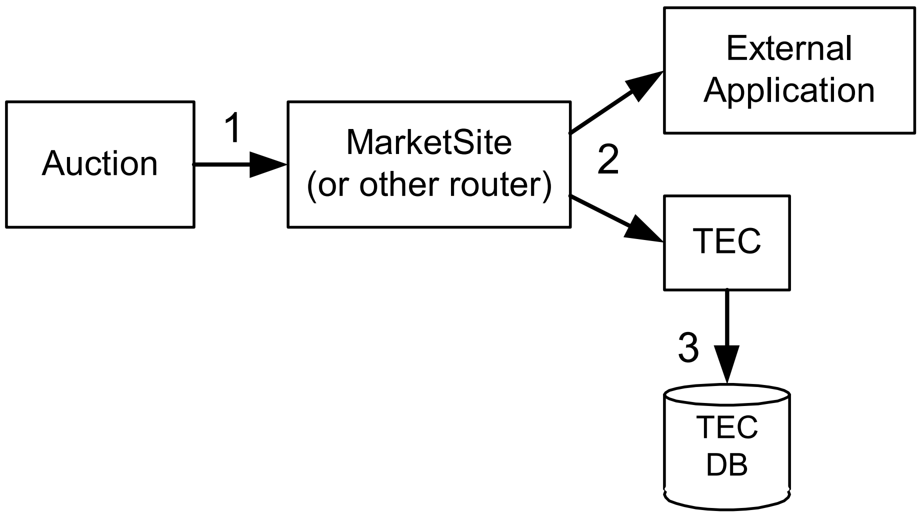
There are several conditions that can interfere with the normal execution of the integration process, such as supplying invalid user credential or database failure. When an error occurs, the error is logged and an error page is returned to the user with appropriate error message.

5 Transaction Event Collector Integration

The Transaction Event Collection (TEC) service is a document service that runs as a layer above the MarketSite platform. This service tracks documents and records their location throughout the platform. You can use these document records to create reports or to establish a billing system framework, for example. Refer to the *Transaction Event Collection Installation and Administration Guide* for more information.

Figure 5-9 illustrates the flow for the TEC integration.

Figure 5-9 Auction/TEC Integration Flow Diagram



1. The Auction application sends an AuctionEvenResult document to MarketSite (or another router).
2. The AuctionEvenResult document is sent to the external application and the TEC application.
3. The TEC application stores a copy of the AuctionEvenResult document in its database.

A Error Messages

This appendix lists and defines the error messages you may encounter when you perform the integrations described in this book.

Error Messages for XPC Integration Programs

Error Message	Description
Tpid {0} not found	Displays when the TPID does not exist. To solve this problem, make sure that the TPDSync was done properly and specify a valid TPID in the following element: <AuctionEventCreate\AuctionEventCreateHeader\InitiatingParty\PartyID\Agency\Ident>
Company with tpId {0} not found	Displays when Multiple Organisations have the same TPID. To solve this problem, change the value in the following element: <AuctionEventCreate\AuctionEventCreateHeader\InitiatingParty\PartyID\Agency\Ident>
Agency Code must be "AssignedByMarketPlace".	Displays when the element <AuctionEventCreate\AuctionEventCreateHeader\InitiatingParty\PartyID\Agency\AgencyCoded> contains the value "Other" but the element <AuctionEventCreate\AuctionEventCreateHeader\InitiatingParty\PartyID\Agency\AgencyCodedOther> does not contain the value "C1AssignedOrgID". To solve this problem, change the value of AgencyCoded to “AssignedByMarketPlace”, or change the AgencyCodedOther element to "C1AssignedOrgID".
Expecting {0} document but found {1}	Displays when the document is not of the expected document type.
Quantity must be a value, not in range	Displays if the quantity for an Auction item is specified as the choice QuantityRange. To solve this problem, modify the element <AuctionEventCreate\ListOfAuctionLotCreateDetail\AuctionLotCreateDetail\ListOfAuctionLineItemDetail\AuctionLineItemDetail\AuctionQuantity> to use the choice QuantityValue.
Base currency not defined	Displays when the currency assigned is not a valid currency. To solve this problem, check the element <AuctionEventCreate\ListOfAuctionLotCreateDetail\AuctionLotCreateDetail\AuctionLot\AuctionLotCurrency\BaseCurrency\CurrencyCoded>
Reference currency not defined	If this message is displays, enter a valid value for the currency coded in the element <AuctionEventCreate\ListOfAuctionLotCreateDetail\AuctionLotCreateDetail\AuctionLot\AuctionLotCurrency\ListOfRateOfExchangeDetail\RateOfExchangeDetail\ReferenceCurrency\CurrencyCoded>

Error Message	**Description**
Target currency not defined	Displays when no currency is entered for the element <AuctionEventCreate\ListOfAuctionLotCreateDetail\AuctionLotCreateDetail\AuctionLot\AuctionLotCurrency\ListOfRateOfExchangeDetail\RateOfExchangeDetail\TargetCurrency\CurrencyCoded>
Currency Code {0} not valid	Displays if an invalid currency code is entered. To solve this problem, check all CurrencyCoded elements and make sure that the valid currency code is supplied.
Exchange rate not defined	If multi-currency is checked, this message displays when a RateOfExchangeDetail is not specified. To solve this problem, enter a value for "RateOfExchangeDetail" in the element <AuctionEventCreate\ListOfAuctionLotCreateDetail\AuctionLotCreateDetail\AuctionLot\AuctionLotCurrency\ListOfRateOfExchangeDetail\RateOfExchangeDetail>
Exchange rate {0} not valid	If multi-currency is checked, this message displays when an invalid RateOfExchange is specified. To solve this problem, enter a valid value for "RateOfExchange" in the element <AuctionEventCreate\ListOfAuctionLotCreateDetail\AuctionLotCreateDetail\AuctionLot\AuctionLotCurrency\ListOfRateOfExchangeDetail\RateOfExchangeDetail\RateOfExchange>
UOM Code not defined.	Displays when a UOM is not specified in the element <AuctionEventCreate/ListOfAuctionLotCreateDetail/AuctionLotCreateDetail/ListOfAuctionLineItemDetail/AuctionLineItemDetail/AuctionQuantity/UnitOfMeasurement/UOMCoded>
UOM Code {0} not valid	Displays when a UOM is not valid to the code present in the ActiveMarket Database in the element <AuctionEventCreate/ListOfAuctionLotCreateDetail/AuctionLotCreateDetail/ListOfAuctionLineItemDetail/AuctionLineItemDetail/AuctionQuantity/UnitOfMeasurement/UOMCoded>
Open price {0} not valid	Displays when an invalid price is entered in the element <AuctionEventCreate\ListOfAuctionLotCreateDetail\AuctionLotCreateDetail\ListOfAuctionLineItemDetail\AuctionLineItemDetail\AuctionLineItemPricingDetail\LineItemOpenPrice>

Error Message	**Description**
Reserve price {0} not valid	Displays when an invalid reserve price is entered in the following element. To solve this problem, correct the value in the element. <AuctionEventCreate\ListOfAuctionLotCreateDetail\AuctionLotCreateDetail\ListOfAuctionLineItemDetail\AuctionLineItemDetail\AuctionLineItemPricingDetail\LineItemReservePrice>
Document received is null	Displays if no document is sent in the envelope. Resend the document to fix this problem.
Part Number {0} is too long. Maximum is {1} characters	Displays if the partnumber exceeds the maximum number of characters allowed. To solve this problem, change the PartID in the following element <AuctionEventCreate\ListOfAuctionLotCreateDetail\AuctionLotCreateDetail\ListOfAuctionLineItemDetail\AuctionLineItemDetail\PartNumbers>
Event Name {0} is too long. Maximum is {1} characters	Displays if the Event Name exceeds the maximum number of characters allowed. To solve this problem, change the following element <AuctionEventCreate\AuctionEventCreateHeader\AuctionEventCreateName>
Lot Name {0} is too long. Maximum is {1} characters	Displays if the Lot Name exceeds the maximum number of characters allowed. To solve this problem, change the following element <AuctionEventCreate\ListOfAuctionLotCreateDetail\AuctionLotCreateDetail\AuctionLot\AuctionLotName >
Item Name {0} is too long. Maximum is {1} characters	Displays if the Line Item Name exceeds the maximum number of characters allowed. To solve this problem, change the element <AuctionEventCreate\ListOfAuctionLotCreateDetail\AuctionLotCreateDetail\ListOfAuctionLineItemDetail\AuctionLineItemDetail\LineItemName>
Time extension id {0} is invalid.	Displays when "Other" is specified as the time extension type. To solve this problem, change the following element to one of the coded values, and not Other. <AuctionEventCreate\ListOfAuctionLotCreateDetail\AuctionLotCreateDetail\TimeExtensionType\TimeExtensionTypeCoded>
Unknown error caused Auction creation to fail.	Displays when an error occurs during creation of the auction lot, and no error message is available. Check all data in the lot and make necessary changes, and try resending the document. For more information look into the Log file.

Error Message	Description
At least one of the lots had error.	The auction event is not created because at least one of the auction lot specified has error. Check the response code for each auction lot and make the change correspondingly.
Multiple category {0} found	More than one category is found with the specified name. To solve the problem, specify a unique category name in the element <AuctionEventCreate\ListOfAuctionLotCreateDetail\AuctionLotCreateDetail\AuctionLot\ListOfAuctionCategory\AuctionCategory\AuctionCategoryName>
Category {0} not found	Displays when the category specified does not exists. To solve the problem, specify the name of an existing category in the element <AuctionEventCreate\ListOfAuctionLotCreateDetail\AuctionLotCreateDetail\AuctionLot\ListOfAuctionCategory\AuctionCategoryName>
Company with orgId {0} not found	Displays when the company with the specified org id does not exist. Make sure that TPDSyn is done properly.
Multiple company with orgId {0} found	Displays when more than one company is found with the specified org id. Make sure that TPDSyn is done properly.
Error finding company with tpId {0}	This message is displayed when performing TPDSyn through the TPDSync.jsp utility page and the company with the specified tpid does not exist.
Multiple category specified for category level {0}	Displays when the correspondong category level is specified more than once. Make sure that unique number is specify in the element <AuctionEventCreate\ListOfAuctionLotCreateDetail\AuctionLotCreateDetail\AuctionLot\ListOfAuctionCategory\AuctionCategory\AuctionCategoryLevel>
Invalid category level {0}. Only {1} categories specified. Make sure that a category is specified for each level.	The specified category level is invalid. To solve this problem, make sure that a number between -1 and total number of category specified, exclusive, is specified in the element <AuctionEventCreate\ListOfAuctionLotCreateDetail\AuctionLotCreateDetail\AuctionLot\ListOfAuctionCategory\AuctionCategory\AuctionCategoryLevel>
At least two category level is required	When the above message displays, make sure at least two category levels are present. Look into the following element <AuctionEventCreate\ListOfAuctionLotCreateDetail\AuctionLotCreateDetail\AuctionLot\ListOfAuctionCategory>

Error Message	**Description**
The root product classification for the company is not defined	Displays when the category specified for level 0 is not found. Make sure that either “Company Catalog” or “Network Catalog” is specified for the element <AuctionEventCreate\ListOfAuctionLotCreateDetail\AuctionLotCreateDetail\AuctionLot\ListOfAuctionCategory\AuctionCategory\AuctionCategoryName>
An internal error occurred while processing your request.	This message is displayed when TPD sync has not taken place or authentication of the document sender or auction originator failed.
Error Info Document	This is a system error and look into the error document sent.Look into the Log file too.
An error occurred while getting document from envelope	Displays if the document contained in the envelope is not extractable by the script. Look into the Log file and try resending the document.
The sender tpid retrieved from the envelope is null.	Displays when the document envelope does not contain the sender ID. Make sure that proper credential is enclosed in the envelope.
Invalid Agency Code specified in AuctionPartners	Displays if AgencyCoded element does not contain a valid value. Change the element below to "AssignedByMarketPlace" or "Other". <AuctionEventCreate\ListOfAuctionLotCreateDetail\AuctionLotCreateDetail\AuctionLot\AuctionLotParticipants\ListOfAuctionPartners\AuctionPartners\Party\PartyID\Agency\AgencyCoded>
Invalid AgencyCodedOther specified in AuctionPartners	Displays when AgencyCoded is set to Other but the AgencyCodedOther does not contain a valid value. Change the following element to "C1AssignedOrgID": <AuctionEventCreate\ListOfAuctionLotCreateDetail\AuctionLotCreateDetail\AuctionLot\AuctionLotParticipants\ListOfAuctionPartners\AuctionPartners\Party\PartyID\Agency\AgencyCodedOther>
Invalid AgencyCodedOther specified in ListOfIdentifier in AuctionPartners	Displays if there is no user id or group id specified. To solve this problem, make sure that valid id is supplied in the following element: <AuctionEventCreate\ListOfAuctionLotCreateDetail\AuctionLotCreateDetail\AuctionLot\AuctionLotParticipants\ListOfAuctionPartners\AuctionPartners\Party\ListOfIdentifier\Identifier\Ident>

Error Message	Description
List Of Identifier cannot be null for AuctionPartners	Displays if there are no bidders are specified for private auction. Partners in the element. Make sure that the following elements are specified: <AuctionEventCreate\ListOfAuctionLotCreateDetail\AuctionLotCreateDetail\AuctionLot\AuctionLotParticipants\ListOfAuctionPartners> and <AuctionEventCreate\ListOfAuctionLotCreateDetail\AuctionLotCreateDetail\AuctionLot\AuctionLotParticipants\ListOfAuctionPartners\AuctionPartners\Party\ListOfIdentifier>
Cant find unique Employee for specified loginName/OrgId	Displays when a unique user cannot be found with the data specified. Make sure that proper id is specified.
Could not find unique company for specified TPID - {0}	Displays when a unique company cannot be found with the correpsonding TPID specified. Make sure that TPDSyn has been performed, and that proper TPID is supplied.
Could not find unique company for specified OrgID - {0}	Displays when a unique company cannot be found for the corresponding org id specified Make sure that valid org id is supplied in the Ident element when AgencyCodedOther is set to "C1AssignedOrgID"
Could not find unique Company Unit for Company, {0}, and CompanyUnit, {1}	Displays when a unique company unit cannot be found for the corresponding company specified Make sure that valid company unit name is supplied in the Ident element when AgencyCodedOther is set to "C1AssignedCompanyUnitID"
Invalid auction line item specified.	Displays when the auction lot failed to be created because there is error in the auction line item. Check the response code in the element <AuctionEventCreateResponse/ListOfAuctionLotCreateResponseDetail/AuctionLotCreateResponseDetail/ListOfAuctionLineItemResponseDetail/AuctionLineItemResponseDetail/AuctionLineItemResponseCoded> for more detail.
Invalid bidder visibility rule specified.	Displays when VisibilityRuleCoded is set to "Other". To solve this problem, modify the following element to one of the supported coded values: <AuctionEventCreate\ListOfAuctionLotCreateDetail\AuctionLotCreateDetail\AuctionLot\AuctionLotRulesProfile\ListOfBidderVisibilityRule\BidderVisibilityRule\VisibilityRuleCoded>

Error Message	Description
Time interval for auction closure alert is not specified.	Displays when there is no time interval specified for the auction closure warning and the visibility rule "SeeTimeLeftForBidding" is set to true. Add the corresponding time interval in the following elements: <AuctionEventCreate\ListOfAuctionLotCreateDetail\AuctionLotCreateDetail\AuctionLot\AuctionLotRulesProfile\ListOfBidderVisibilityRule\BidderVisibilityRule\ListOfVisibilityRuleAdditionalInfo\VisibilityRuleAdditionalInfo>
Invalid time interval for auction closure alert specified.	Displays when an invalid time interval value is specified for the the visibility rule "SeeTimeLeftForBidding". Make sure a positive integer is supplied for the following element: <AuctionEventCreate\ListOfAuctionLotCreateDetail\AuctionLotCreateDetail\AuctionLot\AuctionLotRulesProfile\ListOfBidderVisibilityRule\BidderVisibilityRule\ListOfVisibilityRuleAdditionalInfo\VisibilityRuleAdditionalInfo\AdditionalInfoValue>
Percentage off for from leader is not specified.	Displays when there is no percentage specified and the visibility rule " " is set to true. Add the corresponding percentage in the following elements: <AuctionEventCreate\ListOfAuctionLotCreateDetail\AuctionLotCreateDetail\AuctionLot\AuctionLotRulesProfile\ListOfBidderVisibilityRule\BidderVisibilityRule\ListOfVisibilityRuleAdditionalInfo\VisibilityRuleAdditionalInfo\AdditionalInfoValue>
Invalid percentage off for from leader specified.	Displays when an invalid percentage value is specified for the the visibility rule "SeeBiddersPercentageOffFromLeader". Make sure an integer within 0 and 99 is supplied for the following element: <AuctionEventCreate\ListOfAuctionLotCreateDetail\AuctionLotCreateDetail\AuctionLot\AuctionLotRulesProfile\ListOfBidderVisibilityRule\BidderVisibilityRule\ListOfVisibilityRuleAdditionalInfo\VisibilityRuleAdditionalInfo\AdditionalInfoValue>
The document is not processed because the purpose code is not original	Displays if PurposeCoded does not contain the value "Original". To solve this problem, modify the following element to contain the value "Original" <AuctionEventCreate\AuctionEventCreateHeader\AuctionEventCreatePurpose\PurposeCoded>

Error Message	Description
The category parameter {0} is not defined	Displays when the category parameter specified is not found. To solve this problem, make sure that valid parameter for the selected category is specified in the element <AuctionEventCreate\ListOfAuctionLotCreateDetail\AuctionLotCreateDetail\ListOfAuctionLineItemDetail\AuctionLineItemDetail\ListOfCategoryParameter\CategoryParameter\CategoryParameterName>
Time interval increment is required for dutch auction.	Displays if the bid rule is "Dutch" and no time interval increment is specified. Make sure that the time interval increment is supplied in the following element: <AuctionEventCreate\ListOfAuctionLotCreateDetail\AuctionLotCreateDetail\ListOfAuctionLineItemDetail\AuctionLineItemDetail\ListOfCategoryParameter\CategoryParameter\CategoryParameterName>

Error Messages for eMPC Integration Program

Error Message	Description
Invalid URL for the eMPC	Displays when invalid eMPC URL is configured for the organization. Update the eMPC URL in org preferences, and refer to *Auction Installation and Configuration* Guide for details.
Could not redirect to eMPC.	Displays when it failed to connect to eMPC. Make sure that valid eMPC URL is set under org preference and that the specified instance of eMPC is functioning properly.
Invalid Query String parameter passed.	Displays when there is error cancelling the importing process. If this occurs, contact your administrator.
Error encountered while creating line item.	Displays when creation of a line item fails. OCI Data posted by eMPC might be incorrect or invalid data might be supplied.
There was an error in processing your request. Please contact System Administrator.	Displays during the redirection of the page to the Auction application. Check the Auction site, The error might be in the script.
There was an error in creating a process.	Displays when a new process cannot be created. Check the Database connectivity.

Error Messages for EBD Integration Program

Error Message	Description
Invalid User Name or Password. Please try again.	Displays when an invalid user credential is passed in the request. Make sure that the user is the Administrator of the company when initiating the Round Trip.
Invalid BUYER_USERNAME specified.	Displays when an invalid user name is passed in the Buyer_UserName field in the request.

B Response Codes

This appendix lists and defines the response codes you may encounter when you perform the integrations described in this book.

Auction Event Response Codes

Error Code	Description
Created	Auction Event/lots/items are created successfully.
Rejected	Document rejected because of bad data.
EventNameTooLong	Event Name entered in the following field is exceeded the maximum length of 40 characters.
InvalidAgencyCode	Invalid Agency Code. Agency Code must be "AssignedByMarketPlace".
InvalidTradingPartnerInfo	Invalid Trading Partner Info. This message is displayed when Multiple Organizations have the same TPID. <AuctionEventCreate\AuctionEventCreateHeader\InitiatingParty\PartyID\Agency\Ident>
CredentialsNotSetWithinTheEnvelope	Credential Not Set Within the Envelope. The TIPD coming within credentials is invalid.
TradingPartnerInfoWithinCredentialsIsInvalid	Trading Partner Info Within Credentials is Invalid. Marketsite Participant ID does not corresponds to any one of the entry in ActiveMarket Database.This value is extracted from the credentials of the envelope received.
TradingPartnerInfoDiscrepencyBetweenDocumentAndEnvelope	Trading Partner Info Discrepancy Between Document And Envelope. The Sender ID and the TPID in the document must be the same.
InvalidUserInfoInTheCredentials	Invalid User Info In The Credentials. The user credentials coming inside the envelope couldn't be verified against ActiveMarket database. Check if the Trading Partner Synchronization process has taken place.
InvalidLotDependency	Invalid Lot Dependency. Specify from the list of valid lot dependency coded.

Auction Lot Response Codes

Error Code	Description
AuctionLotCreated	Auction Lot Created. Lot successfully created.
AuctionLotOKButNotCreated	Auction Lot OK But Not Created. Lot creation has failed because other lots may have errors.
InvalidAuctionLotCategory	Invalid Auction Lot Category. Category contained in the following element does not exists in the ActiveMarket database. <AuctionEventCreate\ListOfAuctionLotCreateDetail\AuctionLotCreateDetail\AuctionLot\ListOfAuctionCategory\AuctionCategoryName>
InvalidAuctionLotSubCategory	Invalid Auction Lot Sub Category. Sub Category specified might not exists in ActiveMarket database. Modify the following element <AuctionEventCreate\ListOfAuctionLotCreateDetail\AuctionLotCreateDetail\AuctionLot\ListOfAuctionCategory\AuctionCategoryName>
InvalidAuctionLotName	Invalid Auction Lot Name. Length should not exceed more that 100.
InvalidAuctionLotDescription	Invalid Auction Lot Description. Length should not exceed more that 256.
StartTimeLessThanSystemTime	Start Time Less Than System Time. Change the start time greater than the system time. Auction start time cannot be less that the system time during creation.
EndTimeLessThanStartTime	End Time Less Than Start Time. Lot End time is less than the start time. Make the end time greater than the start time.
InvalidTimeExtensionType	Invalid Time Extension Type. The above error message is displayed when the following element does not contain a valid coded value. Change the below element to reflect one of the values and not "Others". <AuctionEventCreate\AuctionEventCreate\ListOfAuctionLotCreateDetail\AuctionLotCreateDetail\TimeExtensionType\TimeExtensionTypeCoded>
InvalidCategoryLevel	Invalid Category Level Invalid category level {0}. Only {1} categories specified. Make sure that a category is specified for each level.
InvalidBidRule	Invalid Bid Rule. Check the bid rules and select from one of the valid bid rules supported by Auctions.

Auction Line Item Response Codes

Error Code	Description
InvalidItemName	Invalid Item Name. Length should not exceed more that 50.
ItemOKButNotCreated	Item OK But Not Created. The item has valid values but the lot creation has failed causing this error.

C Utility Pages

This appendix describes the various utility pages that ship with the Auction application.

QueryData.jsp

The QueryData.jsp page facilitates the lookup of AuctionEventResultResponse xCBL documents residing in the database.

Location

To access the QueryData.jsp page, log into ActiveMarket and point a browser to the following URL:
`http://<auctions>/am/auction/ihub/utils/QueryData.jsp`

Functional Description

You can conduct a search for a document based on common attributes shared by Each AuctionEventResultResponse xCBL document. These attributes are as follows:

- Correlation ID — A number that is unique for the whole cycle. A document with a specific Correlation ID can be retrieved by entering the Correlation ID in the **Search** field on the QueryData.jsp page.
- Document ID — A number that is unique to the document, and is stored in the <AuctionEventResultResponseID> element. A document with a specific Document ID can be retrieved by entering the Document ID in the **Search** field on the QueryData.jsp page.
- Trading Partner ID (TPID) — The Marketsite Participant ID, which corresponds to the Sender ID in the document. A document containing a specific TPID can be retrieved by entering the TPID in the **Search** field on the QueryData.jsp page.
- Response Code — One of the following values, which can be found in the <AuctionResultResponseCode> element:
  - Accepted
  - RejectedResend
  - RejectedDoNotRend
  - Other

  If you enter a Response Code in the **Search** field, the QueryData.jsp page searches the Response Codes of all the AuctionEventResultResponse documents in the database and returns any matches.

If more you enter more than one field, only the data for the first field enetered is used by the QueryData.jsp page.

Note: A document is added to the result of a query only if the lot associated to the document is accessible to you in the Auction application.

TPDSync.jsp

You can use the TPDSync.jsp page to assign the TPID for an existing Organization in the Auction application if the Trading Partner Synchronization process has not taken place. When a TPID is mapped, the value is added in the AMONETWORKMEMBER table in the ActiveMarket database.

Location

To access the TPDSync.jsp page, log into ActiveMarket and point a browser to the following URL:

```
http://<auctions>/am/auction/ihub/utils/TPDSync.jsp
```

Functional Description

The TPDSync.jsp page contains the following fields:

- Organization ID — When you enter an Organization ID and press the tab button, the TPDSync.jsp page looks for the TPID corresponding to the Organization. If the TPID is present, the TPDSync.jsp page displays the TPID; if it is not present, the field is left blank.
- Trading Partner ID — If this field is blank, you can enter a TPID you want to map with the organization; if this field contains a TPID, you can modify the TPID by editing the value.

Click on the UpdateTPID field when the data entry is complete. Click on the Cancel button to cancel the Trading Partner Synchronization process.

Note: You can set the TPID for an organization only if you have administrative rights for that organization. If you do not have administrative rights for that organization, you can see the TPID but cannot change it.

PerformTPDSync.jsp

The PerformTPDSync.jsp page does the necessary processing to data submitted via the TPDSync.jsp page, and displays the appropriate messages.

The PerformTPDSync.jsp page appears when you click either the UpdateTPID button or the Cancel button on the TPDSync.jsp page. You cannot access this page directly by pointing a browser to a URL.

Functional Description

The PerformTPDSync.jsp page returns one of the messages in the table below.

Message	Description
TPD Sync performed successfully.TPID updated in ActiveMarket database.	Displays when the OrgID exists in the ActiveMarkets database and the TPID entered is unique in the AMONETWORKMEMBER table.
This TPID has already been assigned to another trading partner and cannot be used.	Displays when an existing TPID is mapped to the same or different ORGID.
Invalid org id. No company exists for the org id entered.	Displays when an ORGID that does not exists in the ActiveMarkets database is entered in TPDSync.jsp.
TPD Sync cancelled by User.	Displays when the Cancel button is clicked on TPDSync.jsp.

ResendDocument.jsp

If an outbound document (such as the AuctionEventCreateResponse or AuctionEventResult) is not sent, you can use the ResendDocument.jsp page to resend it.

Location

To access the ResendDocument.jsp page, log into ActiveMarket as a super user and point a browser to the following URL:
`http://<auctions>/am/auction/ihub/utils/ResendDocument.jsp`

Functional Description

When you access the ResendDocument.jsp page, it returns a list of all the documents that failed to get delivered if any exist. If one or more unsent documents exist, the ResendDocument.jsp page displays the following fields and information:

- Select checkbox — Use this checkbox to select specific documents to resend.
- Event Name — The Auction Event name contained in the document, which allows you to distinguish one document from another.
- Document ID — The unique ID for the document. Click this link to access a page that displays the document.
- Document Type — Indicates whether the document is an AuctionEventCreateResponse or AuctionEventResult.
- PID — An internal ID generated by the system identifying the process that will be resumed if the document needs to be resent.
- Resend Selected Documents — A button you can click to send the documents you select. Select a document by checking the Select checkbox next to it.
- Back — Click the Back button to go back to the first page, which lists any unsent documents.

In addition to the fields and information above, the ResendDocument.jsp page may also display the messages in the following table.

Message	**Description**
There are no unsent documents.	Displays when there are no unsent documents in the database.
The selected document(s) have been resent.	Displays when an attempt has been made to resend the selected document(s).

After the ResendDocument.jsp page resends a document, it deletes the entry in the database. When the list of unsent document refreshes, the resent document should be absent from the list. If the resending of the document fails, a new entry is created in the database, and the document once again displays in the list of unsent documents.

D Sample Data

This appendix provides sample xCBL code to further your understanding of the Auction integration.

Specify a Company to Host an Auction

Use the PartyID element to specify the company and/or company unit for which an auction is intended.

There are two ways to use the `InitiatingParty/Party/PartyID` element to specify a company:

- TPID-based Identification
- OrgID-based Identification

TPID-based Identification

Use this option to identify the company by its TPID.

Note: Prior to using this option, run the Trading Partner Synchronization process for the company so that the integration application can find the company by its TPID.

To use this option, do the following:

1. Set `PartyID/Identifier/Agency/AgencyCoded` to "`AssignedByMarketPlace`".
2. Set `PartyID/Identifier/Ident` to the TPID of the company for which you are creating the auction.

Example

```
<InitiatingParty>
   <Party>
      <PartyID>
         <Identifier>
            <Agency>
               <AgencyCoded>AssignedByMarketPlace
               </AgencyCoded>
            </Agency>
            <Ident>536006fe-77ac-1000-9b16-ac14080d0001
            </Ident>
         </Identifier>
      </PartyID>
   </Party>
</InitiatingParty>
```

OrgID-based Identification

Use this option to identify the company by the Org ID assigned to it by the ActiveMarket platform. This option makes it possible to create an auction for a company that does not have a TPID, is not registered with MarketSite, or has not had the Trading Partner Synchronization process completed for it.

To use this option, do the following:

1. Set `PartyID/Identifier/Agency/AgencyCoded` to "`Other`".
2. Set `PartyID/Identifier/Agency/AgencyCodedOther` to "`C1AssignedOrgID`".
3. Set `PartyID/Identifier/Ident` to the companyCode of the company for which you are creating the auction.

Note: Company identification using any agency other than “AssignedByMarketPlace” and “C1AssignedOrgID” is not supported.

Example

```
<InitiatingParty>
   <Party>
      <PartyID>
         <Identifier>
            <Agency>
               <AgencyCoded>Other
               </AgencyCoded>
               <AgencyCodedOther>C1AssignedOrgID
               </AgencyCodedOther>
            </Agency>
            <Ident>MyCompanyID</Ident>
         </Identifier>
      </PartyID>
   </Party>
</InitiatingParty>
```

Specify a Company Unit to Host an Auction (Optional)

You can optionally use the ListOfIdentifiers element when you want to create an auction for a specific company unit in the company specified in the PartyID element.

Note: If the ListOfIdentifiers element is not specified, or does not meet the specification described here, the auction is created for the root company unit of the company.

For the ListOfIdentifiers element to have effect, set one of the instances of the Identifier element under it as follows:

1. Set the `Identifier/Agency/AgencyCoded` element to “`Other`”.
2. Set the `Identifier/Agency/AgencyCodedOther` element to "`C1AssignedCompanyUnitID`".
3. Set the `Identifier/Ident` element to the name of the company unit for which you are creating the auction.

Example One

In this example, the TPID identifies the company, and a company unit called “Unit1” is set as the auction owner.

```
<InitiatingParty>
   <Party>
      <PartyID>
         <Identifier>
            <Agency>
               <AgencyCoded>AssignedByMarketPlace
               </AgencyCoded>
            </Agency>
            <Ident>536006fe-77ac-1000-9b16-ac14080d0001
            </Ident>
         </Identifier>
      </PartyID>
      <ListOfIdentifier>
         <Identifier>
            <Agency>
               <AgencyCoded>Other
               </AgencyCoded>
               <AgencyCodedOther>C1AssignedCompanyUnitID
               </AgencyCodedOther>
               <AgencyDescription>Description for agency
               </AgencyDescription>
            </Agency>
            <Ident>Unit1
            </Ident>
         </Identifier>
      </ListOfIdentifier>
   </Party>
</InitiatingParty>
```

Example Two

In this example, the OrgID identifies the company, and a company unit called “Unit1” is set as the auction owner.

```
<InitiatingParty>
   <Party>
      <PartyID>
         <Identifier>
            <Agency>
               <AgencyCoded>Other
               </AgencyCoded>
               <AgencyCodedOther>C1AssignedOrgID
               </AgencyCodedOther>
            </Agency>
            <Ident>MyCompanyID
            </Ident>
         </Identifier>
      </PartyID>
      <ListOfIdentifier>
         <Identifier>
            <Agency>
               <AgencyCoded>Other
               </AgencyCoded>
               <AgencyCodedOther>C1AssignedCompanyUnitID
               </AgencyCodedOther>
               <AgencyDescription>Description for agency
               </AgencyDescription>
            </Agency>
            <Ident>Unit1
            </Ident>
         </Identifier>
      </ListOfIdentifier>
   </Party>
</InitiatingParty>
```

Specify Individual Bidder Users for Private Auctions

Use the AuctionPartners element to specify bidder users when creating a private auction. To do this, do the following:

1. Specify the company that owns the user group by following the steps in Specify a Company to Host an Auction on page D-2.
2. Set the `AuctionPartners/ListOfIdentifiers/Identifier/Agency/AgencyCoded` element to "`Other`".
3. Set the `AuctionPartners/ListOfIdentifiers/Identifier/Agency/AgencyCodedOther` element to `"C1AssignedUserID"`.
4. Set the `AuctionPartners/ListOfIdentifiers/Identifier/Ident` element to the name of the user.
5. Set the `AuctionPartners/GroupIndicator` element to "`false.`"

Example One

In this example, the TPID identifies the company, and a user called "User1" is set as the bidder.

```
<AuctionPartners>
   <PartyID>
      <Identifier>
         <Agency>
            <AgencyCoded>AssignedByMarketPlace
            </AgencyCoded>
            <AgencyCodedOther/>
               <AgencyDescription/>
         </Agency>
         <Ident>536006fe-77ac-1000-9b16-ac14080d0001
         </Ident>
      </Identifier>
   </PartyID>
   <ListOfIdentifier>
      <Identifier>
         <Agency>
            <AgencyCoded>Other
            </AgencyCoded>
            <AgencyCodedOther>C1AssignedUserID
         </AgencyCodedOther>
         <AgencyDescription/>
         </Agency>
         <Ident>user1
         </Ident>
      </Identifier>
   </ListOfIdentifier>
   <GroupIndicator>false
   </GroupIndicator>
</AuctionPartners>
```

Example Two

In this example, the orgID identifies the company, and a user called “User1” is set as the bidder.

```
<AuctionPartners>
   <PartyID>
      <Identifier>
         <Agency>
            <AgencyCoded>Other
            </AgencyCoded>
            <AgencyCodedOther>C1AssignedOrgID
            </AgencyCodedOther>
            <AgencyDescription/>
         </Agency>
         <Ident>MyCompanyID
         </Ident>
      </Identifier>
   </PartyID>
   <ListOfIdentifier>
      <Identifier>
         <Agency>
            <AgencyCoded>Other
            </AgencyCoded>
            <AgencyCodedOther>C1AssignedUserID
            </AgencyCodedOther>
            <AgencyDescription/>
            </Agency>
            <Ident>user1
            </Ident>
      </Identifier>
   </ListOfIdentifier>
   <GroupIndicator>false
   </GroupIndicator>
</AuctionPartners>
```

Specify Bidder User Groups for Private Auctions

Use the AuctionPartners element to specify bidder user groups for a private auction. To do this, do the following:

1. Specify the company that owns the user group by following the steps in Specify a Company to Host an Auction on page D-2.
2. Set the `AuctionPartners/ListOfIdentifiers/Identifier/Agency/AgencyCoded` element to “`Other`”.
3. Set the `AuctionPartners/ListOfIdentifiers/Identifier/Agency/AgencyCodedOther` element to `"C1AssignedGroupID"`.
4. Set the `AuctionPartners/ListOfIdentifiers/Identifier/Ident` element to both the user group’s company unit and the name of the user group to be selected, using a bar (|) to separate them. For example, if the company unit is “unit1” and the user group is “group1”, enter:

   ```
   unit1|group1
   ```

5. Set the `AuctionPartners/GroupIndicator` element to “true”.

Example One

In this example, the TPID identifies the company, and a user group called “group1” is set as the bidder user group.

```
<AuctionPartners>
   <PartyID>
      <Identifier>
         <Agency>
            <AgencyCoded>AssignedByMarketPlace
            </AgencyCoded>
         <AgencyCodedOther/>
         <AgencyDescription/>
         </Agency>
         <Ident>536006fe-77ac-1000-9b16-ac14080d0001
         </Ident>
      </Identifier>
   </PartyID>
   <ListOfIdentifier>
      <Identifier>
         <Agency>
            <AgencyCoded>Other
            </AgencyCoded>
            <AgencyCodedOther>C1AssignedGroupID
            </AgencyCodedOther>
            <AgencyDescription/>
         </Agency>
         <Ident>unit1|group1
         </Ident>
      </Identifier>
   </ListOfIdentifier>
   <GroupIndicator>true
   </GroupIndicator>
</AuctionPartners>
```

Example Two

In this example, the orgID identifies the company, and a user group called "group1" is set as the bidder.

```
<AuctionPartners>
   <PartyID>
      <Identifier>
         <Agency>
            <AgencyCoded>Other
            </AgencyCoded>
            <AgencyCodedOther>C1AssignedOrgID
            </AgencyCodedOther>
            <AgencyDescription/>
         </Agency>
         <Ident>MyCompanyID
         </Ident>
      </Identifier>
   </PartyID>
   <ListOfIdentifier>
      <Identifier>
         <Agency>
            <AgencyCoded>Other
            </AgencyCoded>
            <AgencyCodedOther>C1AssignedGroupID
            </AgencyCodedOther>
            <AgencyDescription/>
         </Agency>
         <Ident>unit1|group1
         </Ident>
      </Identifier>
   </ListOfIdentifier>
   <GroupIndicator>true
   </GroupIndicator>
</AuctionPartners>
```