opentext™

OpenText™ Intelligent Capture

General Import/Export Modules Guide

This guide contains information about the Intelligent Capture general import/export client modules.

ECPCORE220300-CIO-EN-1

OpenText™ Intelligent Capture
General Import/Export Modules Guide

ECPCORE220300-CIO-EN-1
Rev.: 2022-July-13

This documentation has been created for OpenText™ Intelligent Capture CE 22.3.
It is also valid for subsequent software releases unless OpenText has made newer documentation available with the product, on an OpenText website, or by any other means.

Open Text Corporation

275 Frank Tompa Drive, Waterloo, Ontario, Canada, N2L 0A1

Tel: +1-519-888-7111
Toll Free Canada/USA: 1-800-499-6544 International: +800-4996-5440
Fax: +1-519-888-0677
Support: https://support.opentext.com
For more information, visit https://www.opentext.com

Table of Contents

Chapter 1

General Import/Export Modules

Starting in release 22.3, this single guide contains a compilation of the Intelligent Capture general import/export client modules. In previous releases, these were available as separate guides.

For information about common features and functionalities in the Intelligent Capture client modules, see *OpenText Intelligent Capture - Module Reference (ECPCORE-CMD)*.

Table 1-1: General Import/Export Modules

Module	Original module ID	Link
Standard Import	ecpcore-cim	See "Standard Import Module"
Web Services	ecpcore-cso	See "Web Services Module" on page 19
ODBC Export	ecpcore-cxd	See "ODBC Export Module"
Standard Export	ecpcore-cxb	See "Standard Export Module" on page 109

Chapter 2

Standard Import Module

This section includes the Intelligent Capture general import/export module: **ecpcore-cim**.

This guide presents users with conceptual information about the Standard Import application. It also provides step-by-step instructions on how to use the application. The topics within this section cover basic overview information and introduce the features and functions of the application.

For the specific capabilities of each client module, see *OpenText Intelligent Capture - Administration Guide (ECPCORE-AON)*.

Note: The Standard Import module does not have a setup mode.

2.1 Understanding the Standard Import Module

The Standard Import module performs the actual import as specified in an Import profile. Import profiles specify the following items for import:

- Image files from directories
- Email and attachments from an email server

The Standard Import module can be run as a service in multiple instances, however, every single instance should be associated with a separate Standard Import profile. Standard Import may be deployed across machines. An associated profile must be specified only in one service to avoid two service instances processing same files. For information on using profiles, see *OpenText Intelligent Capture - Designer Guide (ECPCORE-CPD)*.

Specifically, Standard Import performs the following tasks and functions:

- **Import data from multiple sources**: Standard Import transforms data from a folder or an email account to batch data. It can also scan paper documents and import document image files.
- **Profile-based import**: The module gains much of its flexibility through its profile-based nature. It reads the import profiles from Intelligent Capture Server and monitors file system directories and email servers for new data, composes the new data into a node structure of images and values, and then publishes the node structure to IA Server as a batch. Each profile determines the instructions for monitoring only a single folder or a single mailbox. A customer can use a profile to create batches by assigning a specific CaptureFlow. The following types of import profiles are available:

  - File System: Instructions to monitor a file system directory
  - Email: Instructions to monitor an email server
- **Schedule-based polling for files and emails**: The profile defines the polling period and/or polling rule according to which a file directory/email server is polled for data. It also performs periodic checks for any 'Pause' or 'Stop' requests. Exits the flow for a stop request and for a pause request, completes the current work and stops polling until you reset the pause request.
- **Filter-based processing for files**: The module defines filter patterns using regular expressions to select valid file names and extensions for batch processing. It filters the files that match the pattern and then assembles them into a batch. Files that fail the validation process are skipped.
- **Batch assembly and creation**: Standard Import assembles the filtered data and images into a node structure, and creates a batch or batches based on the profile configurations. If the minimum number of files for batch creation is not available, it skips creating the batch and waits until the required number of files is available.
- **Custom scripts**: The Standard Import behavior can be customized using custom scripts. The import profile must be configured to use a script.
- **Multiple output subfolders for files and emails**: The profile determines the output folder into which Standard Import moves the processed items. The unique Listid, assigned to the list of selected items for each import cycle, is used to name the following subfolders within the output folder:
  - `$Success`: Contains all successfully imported items, which are moved to this subfolder.
  - `$Error`: Contains files that encounter an irrecoverable error at runtime.
  - `$Skipped`: Contains files that fail to pass through the filter that decides the file names and extensions to be processed.
  - `$ExtractedFiles`: Contains the files extracted from a zip file. The zip file itself is moved to either $Success, $Error, or $Skipped depending on the processing result.
  - `$NewFiles`: Includes the files that are created using custom scripts.

2.2 Running Standard Import in Production

After the module has been set up, tested, and placed in a production environment, operators and administrators will run the module in production.

For information related to common production tasks for unattended modules, see *OpenText Intelligent Capture - Module Reference (ECPCORE-CMD)*.

Note: *OpenText Intelligent Capture - Module Reference (ECPCORE-CMD)* contains production topics that are common to many unattended modules and may not apply to this module.

Standard Import can be run either using the command line or as a service.

Running Standard Import Using the Command Line

If you want Standard Import to monitor specific import sources, you must launch your module from the command line and specify the profiles that the module must use.

For more information on using the command line, see *OpenText Intelligent Capture - Module Reference (ECPCORE-CMD)*.

Running Standard Import as a Service

You can run Standard Import in service mode with a specific email profile or file profile.

In the `CPIMPORT.config` file located in the `C:\ProgramData\EMC\InputAccel\CPIMPORT\Config` directory, specify the correct profile name as follows:`EmailProfileNames="<Profile name>"`

For more information on running modules in service mode, see *OpenText Intelligent Capture - Administration Guide (ECPCORE-AON)*.

2.2.1 Viewing Module Performance Information

Most modules display performance information during or after processing tasks. The location and content of this information varies by module.

To view modules performance information:

1. Start the module for production in the attended mode.
2. Begin the processing tasks.
3. Click **Information** or **Processing** in the navigation panel (depending on the module) to view information about the processing status.

   The **Processing Log** pane displays the trace log generated by Standard Import. The log includes details such as date and time of events, node numbers, batch name, task progress, warnings, and error messages.

The **Counters** pane displays the following performance-related information:

- **Type**:
  - **S** indicates that the value of an operation is an instant in time.
  - **A** indicates that the value is an accumulation.
- **Name**: The name of the operation.
- **Count**: The value of the operation.
- **AvgExecMs**: The average execution time for the operation.
- **Errors**: The number of times the operation failed.
- **CountPerHour**: The rate at which the operation executes.

4. To export information displayed in the **Processing Log** pane to a file, click **Export Log**.
5. Select a destination directory and file name for the log, and click **Save**.
6. To stop or pause the import process, click **Stop**.

2.3 Reference—Standard Import

The topics within this section contain reference information useful while using the application in setup or production.

2.3.1 IA Values

Intelligent Capture provides both common and module-specific IA Values. The *Common IA Values* described in *OpenText Intelligent Capture - Module Reference (ECPCORE-CMD)* are valid for all modules.

IA values defined in the Standard Import module are described in "Standard Import-specific IA Values" on page 12. The **Type** column identifies the data type for each IA value followed by its modifier (input and/or output). The **Trigger Level** column identifies the IA value level in the batch (from 0 to 7). Trigger level set to `'T'` (up to level 7) indicates the level at which the import step is triggered.

Table 2-1: Standard Import-specific IA Values

IA Value	Trigger Level	Type	Description
`OutputImage`	0-7	File, Output	The image data that is imported.

IA Value	Trigger Level	Type	Description
`OutputFileType`	0-7	String, Output	The file extension (without a period) of the imported image (`OutputImage`); for example, the value for TIFF images is: `tif`
`SourcePath`	0-6	String, Output	The source path for the image file, including the file name and its extension.
`DateModified`	0-6	Date, Output	The date/time the file was last modified.
`ClassificationId`	0	String, Output	The information extraction-generated ID that should be routed to the Collector to assist in learning to classify the page.
`PageDataXML`	0	String, Output	The advanced recognition-generated data that should be routed to the Collector to assist in learning to classify and extract the page.
`ImageResolution`	0	Long, Output	The resolution of the image.
`Source`	7	String, Output	A string that identifies the type of profile that created the batch. The valid values are: • `EMAIL` • `FILE` • `WS` – Web Service layer (for example, the Intelligent Capture REST Service)

IA Value	Trigger Level	Type	Description
`ServiceInstanceName`	7	String, Output	The name associated with the running instance of the service that created this batch. For example, for the Intelligent Capture REST Service, this is the name configured in `web.config` or *`<hostname>:<website>`*. This name is used for diagnostic purposes. It can be auto-generated or added as a Command Line parameter.
`InstanceId`	7	String, Output	A globally unique ID associated with the running instance of the service that created the batch. For example, in the case of the REST server, this is a specific instance id between start and stop of a website. This ID is used for diagnostic purposes. This value is passed as a Command Line parameter.
`ListId`	7	String, Output	The ID of the list of files that was used to create the batch, and the backup destination identifier that was used to name the success, error, skipped, extracted files, and new files subfolders.
`FilesImported`	7	Long, Output	The number of pages that were successfully imported in the batch.

IA Value	Trigger Level	Type	Description
`FileName`	0-6	String, Output	The file name without its extension and path or the name of the attachment without its extension.
`SubPath`	0-6	String, Output	The subdirectory path from the watch folder. The path tokens are separated by the forward slash. May be used to group items.
`UimDocumentType`	1	String, Input, Output	The name of the document data type.
`UimData`	1	String, Output	The data object for the document.
`EmailFrom`	2	String, Output	The *From* display name of an email. The display name can appear empty in some emails, for example if a sender did not set a display name in their contact profile or if the email client did not set it within the email message. This value might appear as follows: `John Smith`.
`EmailFromAddress`	2	String, Output	The *From* email address of the sending account from which the email message was physically drafted and sent. This value will always appear as an email address. For example, `john.smith@domain.com`.

IA Value	Trigger Level	Type	Description
`EmailSender`	2	String, Output	The sender of an email. In some instances, emails have senders that are different from the actual sending email account, for example, when a message is sent on behalf of another user. This value will always appear as an email address. For example, `jane.smith@domain.com`.
`EmailTo`	2	String, Output	A comma-delimited *To* list of recipients of an email.
`EmailCc`	2	String, Output	A comma-delimited *CC* list of recipients of an email.
`EmailSubject`	2	String, Output	The subject line of an email.
`EmailBodyText`	2	String, Output	The body of an email in plain text format. The value is empty if plain text body is not present.
`EmailBodyHtml`	2	String, Output	The body of an email in HTML format. The value is a dummy body if the HTML body is not present.
`EmailDate`	2	Date, Output	The date when the email is sent.
`EmailAttachmentNames`	2	String, Output	A comma separated list of the names of attachments in an email.
`EmailMessageId`	6	String, Output	The unique message ID of the email server.
`EmailServer`	6	String, Output	The host name and port of the email server.

IA Value	Trigger Level	Type	Description
`EmailProtocol`	6	String, Output	The name of the email protocol.
`EmailAccount`	6	String, Output	The email account that is used with the optional folder.
`ServerType`	7	String, Output	The type of server that created the batch. This information is used for diagnostic purposes.
`WatchDirectory`	7	String, Output	The full directory path representing the watch directory.
`ScriptTag`	7	String, Output	The configuration value of *`ScriptTag`* that is taken from the profile.
`StartDateTimeUtc`	T	Date, Output	The UTC date and time the task was started.
`TotalTimeMilliSec`	T	Long, Output	The duration of the task in milliseconds.
`Machine`	T	String, Output	The name of the machine where the task is processed.
`TaskExecutionStatus`	T	Long, Output	The status of the task. Non-zero values reflect only the last time that the task was executed (in the case of tasks that are triggered multiple times). The valid values are: • `0` (Never processed) • `1` (Successfully processed) • `−1` (Processed and failed)

IA Value	Trigger Level	Type	Description
ErrorText	T	String, Output	A non-localized error message if the task execution fails. Empty if not processed successfully. "Canceled" (English) if the task is canceled after execution.
Profile	7	String	Name of the Standard Import profile used in the batch.

Chapter 3

Web Services Module

This section includes the Intelligent Capture general import/export module: **ecpcore-cso**.

This guide is designed to present users with conceptual information about this application, as well as step-by-step instructions on how to use the application. The topics within this section cover basic overview information and introduce the features and functions of the application.

The specific capabilities of each client module are described in the *OpenText Intelligent Capture - Module Reference (ECPCORE-CMD)*.

3.1 Understanding Web Services

The Web Services subsystem consists of several components. For an overview of the components and typical configurations, see "Web Services Architecture" on page 22 The Web Services Coordinator Transactions Database described in this section is part of the Intelligent Capture Database. It is not a separate database instance. It is configured during installation.

The Web Services Input (WS Input) and Web Services Output (WS Output) modules can be used independently or together. The Web Services subsystem enables external systems to interact with Intelligent Capture workflows and enable Intelligent Capture to interact with the workflows of external systems. External systems can use only the specific capabilities of individual Intelligent Capture modules. For example, create a process with a Web Services Input step, a processing module step such as *OCR*, and a Web Services Output step. The Web Services Input step receives data and files from the external system. It creates a batch based on an associated process, and then initiates processing with the specific Intelligent Capture module. After processing completes, the Web Services Output step sends the results back to the external system.

The WS Input and WS Output modules share several common features:

- They are both unattended modules that run only as services. The modules have no interactive production mode.
- Both modules have an interactive setup mode. Use setup mode to configure mappings between IA Values and simple web call parameters (single values, structures, and arrays), and to set error handling options.
- They both have client-side scripting capabilities that enable mapping of more complex parameters. Client-side scripting is configured in setup mode. Both modules share a common `WebServicesScripting` namespace in the .NET scripting libraries.

Understanding Web Services Input

The WS Input module is an Intelligent Capture client module that functions as a web service provider. A step of the WS Input module can be configured at the beginning or in the middle of a process. When used at the beginning of a process, the WS Input module creates new batches as it receives web service requests from external systems. When used in the middle of a process, the module can insert data and files into an existing batch. The WS Input module provides mapping for simple parameters: single values, structures, and arrays. It provides client-side scripting capabilities to enable processing of more complex parameters.

The WS Input module operates under the control of the Web Services Coordinator and Web Services Hosting components. Before using the WS Input module, the Web Services Subsystem must be configured by using Intelligent Capture Administrator.

Understanding Web Services Output

The WS Output module is an Intelligent Capture client module that functions as a web service consumer. A WS Output step is configured at or near the end of a process. It enables the module to export data processed by other modules. The WS Output module enables extracting images, files, and metadata from an Intelligent Capture system to any web-service enabled, third-party system without writing a custom export module.

The WS Output module runs independently and does not rely on the other components in the Web Services Subsystem. Therefore, no configuration is necessary in Intelligent Capture Administrator.

Related Topics

"Web Services Architecture" on page 22

"Understanding Common Usage Scenarios" on page 32

"Setting Up Web Services Input" on page 44

"Setting Up Web Services Output" on page 46

"Mapping Web Call Parameters to IA Values" on page 47

"Reference—Web Services" on page 57

Understanding How Intelligent Capture Processes Batches Using Web Services

Intelligent Capture Web Services provides the standards and tools to build an *SOA* to process work over the Internet and enable the Intelligent Capture system to be a consumer or provider of web services. A Web Service is a self-describing, self-contained, modular software application that provides one or more business functions to other systems through a standard Internet connection. In Figure 3-1, the

customer site, the capture server, and the third-party export system are situated in different locations.

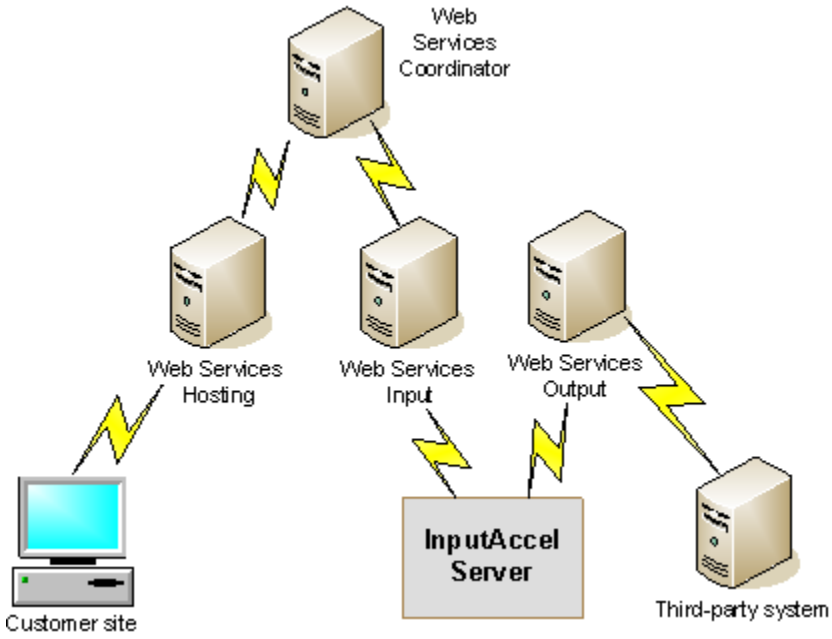

Figure 3-1: Batch processing using Web Services

The components in this example include:

- **Customer site**: Sends batch information with attached documents or a *URL* that references a document to the Web Services Hostings service based on a connection that is defined in Intelligent Capture Administrator. Web Services Input downloads the document file referenced by the URL. The documents are sent to the server as a stage file according to the parameters mapped in the Web Services Input module.
- **Web Services**: The following transactions occur:
  - The Web Services Hosting component acts as a Web Services Server and receives requests.
  - The Web Services Coordinator component manages Web Services Hostings. It also maintains the request/response data in the Intelligent Capture SQL Server database and provides routing between Web Services Hosting and Web Services Input.
  - The Web Services Input module receives documents or downloads the document file referenced by a URL. It extracts input IA Values and forwards them to the Web Services Coordinator component and then waits for output IA Values to send to the server. In Web Services Input setup mode, administrators map web service call parameters to IA Values to determine what information from incoming parameters should be stored in IA Values

and what information should be sent to outgoing parameters from IA Values. The Web Services Input module creates a batch and sends the IA Values and stage files, if any, to the server.

- **Intelligent Capture Server**: Runs a process that includes the Completion module, which indexes and validates data sent to it by the server. When Completion finishes indexing and validating, it returns the batch to the server. The server updates the IA Values and creates a stage file for this step.
- **Third-party system**: The server pushes the batch to the Web Services Output module, which routes the verified data to a third-party system over the Internet.

3.2 Web Services Architecture

A web service is a software system designed to support interoperable machine-to-machine interaction over a network. Typically, a web service is accessible on the Web (or intranet) through a *URL*. It is accessed by clients using *XML*-based protocols, such as *SOAP*, sent over accepted internet protocols, like *HTTP*. Clients can access a web service application through its interfaces and bindings, which are defined using XML artifacts, such as a *WSDL* file.

The Web Services subsystem includes four main components:

- Web Services Input (WS Input)
- Web Services Output (WS Output)
- Web Services Hosting
- Web Services Coordinator

The Web Services subsystem also includes a set of administrative pages, within Intelligent Capture Administrator. They enable managing relationships between and properties of Web Services components. Figure 3-2 demonstrates the typical interactions of these components in a larger system:

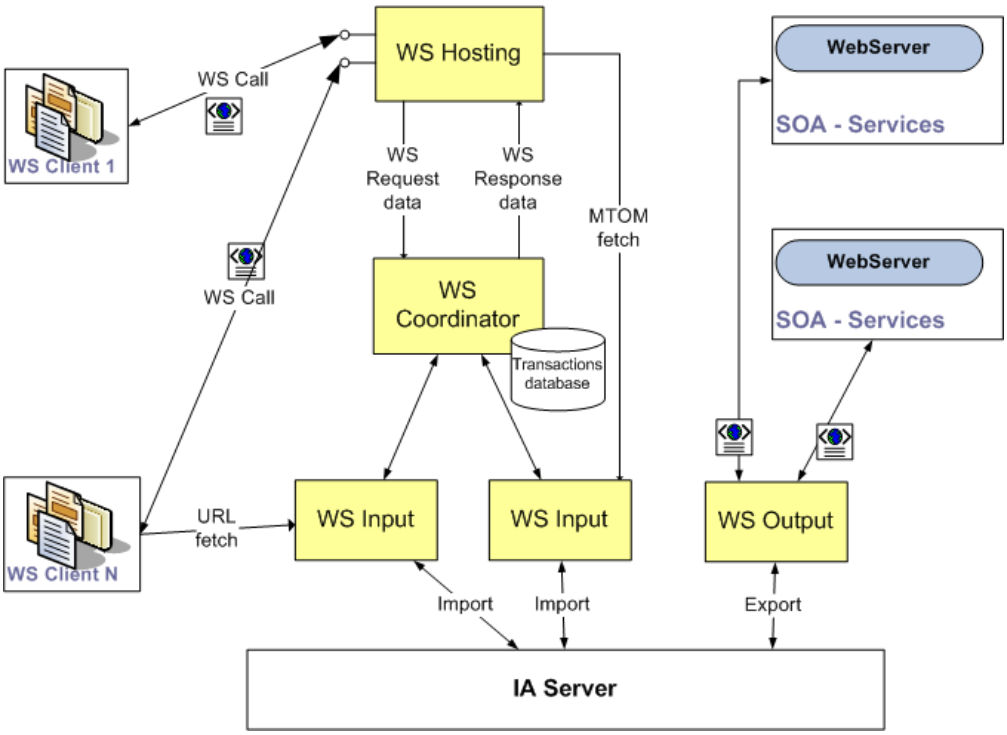

Figure 3-2: Web Services architecture

Related Topics

"Web Services Administration" on page 29

"Sample WSDL File" on page 59

3.2.1 Understanding Web Services Configuration

The Web Services subsystem includes a number of components that must be started and configured separately. These steps in the following procedure represent the most common tasks necessary to configure the components of the Web Services subsystem. Some of these steps do not apply to every implementation. The following procedure assumes that Intelligent Capture is installed, and the Intelligent Capture Server is running.

To set up the Web Services subsystem:

1. Start the **Intelligent Capture Web Services Coordinator** service.

   Note: Start only one (1) Web Services Coordinator service per system.

2. Start the **Intelligent Capture Web Services Hosting** service.

3. Start the **Intelligent Capture Web Services Input** service.
4. Start the **Intelligent Capture Web Services Output** service.
5. Configure a service. For instructions, see "Adding Web Services and Hostings" on page 43.
6. Configure a hosting. For instructions, see "Adding Web Services and Hostings" on page 43.
7. Register a service on a hosting. For instructions, see "Adding Web Services and Hostings" on page 43.
8. If the WS Input module is present as a step in the *IPP*, configure the WS Input module in setup mode. For instructions, see "Setting Up Web Services Input" on page 44.
9. If the WS Output module is present as a step in the IPP, configure the WS Output module in setup mode. For instructions, see "Setting Up Web Services Output" on page 46.
10. Make a valid web call to this registered service.

For a detailed explanation of some standard implementations of the Web Services subsystem in a larger network, see "Understanding Common Usage Scenarios" on page 32

Related Topics

"Web Services Input" on page 24

"Web Services Output" on page 27

"Web Services Coordinator" on page 28

"Web Services Hosting" on page 29

"Mapping Web Call Parameters to IA Values" on page 47

3.2.2 Web Services Input

The WS Input module is an unattended Intelligent Capture module that runs as a Windows service. WS Input acts as a provider of Intelligent Capture-exposed web services. It enables any *SOA* requester entity to access Service Oriented Intelligent Capture services. Web services can be called from either .NET or Java 2 Platform (J2EE) clients. The WS Input module can communicate with any web service client running Web Services Enhancements (WSE) 3.0 or Java API for XML-based Web Services (JAX-WS) 2.0. In Figure 3-3, the WS Input module consists of three components: the WS Input module itself, the Web Services Coordinator and Web Services Hosting. To begin processing, WS Input must receive a web call with parameters from the Web Services Coordinator.

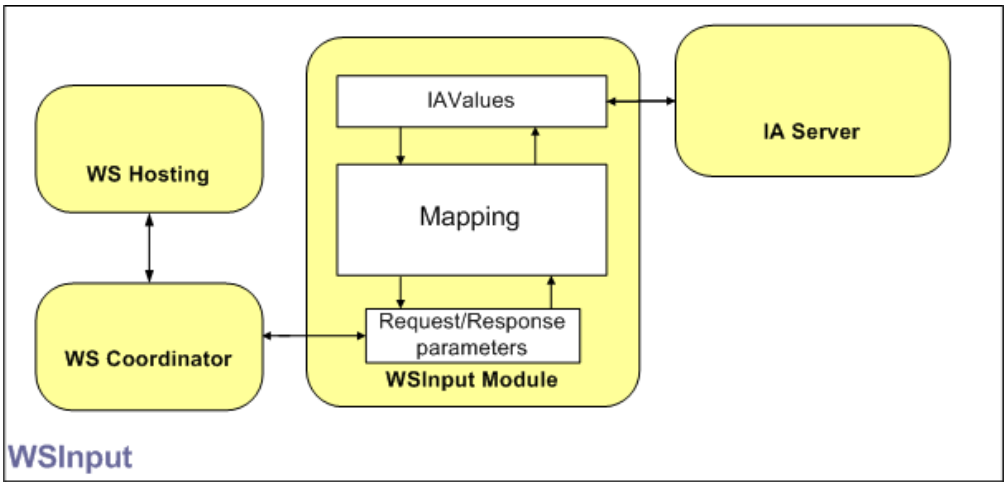

Figure 3-3: Web Services input

In setup mode, the WS Input module enables the visual mapping of simple input parameters to IA Values in the batch. This module also includes a scripting interface for mapping more complex parameters. Upon receiving a triggered task, WS Input uploads the received documents to an existing batch, or to a new batch created for these documents. For information on setting up the WS Input module, see "Setting Up Web Services Input" on page 44.

Processing Web Calls

Unlike many Intelligent Capture modules, which can process a task when triggered, the WS Input module must also receive a web call before processing a task. A triggered task and a web call sharing the same *`Service ID`*, *`Method ID`*, and *`Correlation ID`* form a "web call – task" pair. When the Web Services Coordinator finds a pair in its ready-task queue, it schedules this pair for processing in the first available WS Input module. The WS Input module begins processing upon receiving this "web call – task" pair, even if the batch priority is set to 0 for Offline Processing. Changing the priority of a batch while a task is processing does not affect the batch.

Note: The exception to this behavior occurs when the WS Input module creates a batch. In this case, the Web Services Coordinator only sends the "web call – process name" pair to the WS Input module for processing.

After processing a "web call – task" or "web call – process name" pair, the WS Input module sends the response data to the Web Services Coordinator. This data contains the output parameters for the web response.

The WS Input module handles binary data coming from web requests. However, it has no way of recognizing a file as an image as it is not intended to manipulate images or their thumbnails. For this reason, there can be discrepancies between the images and the thumbnails in the batch, especially after running a batch in another module.

Related Topics

"Create Batch Mode" on page 26

"Authenticating Web Service Input Calls" on page 39

"Mapping Web Call Parameters to IA Values" on page 47

"Web Services Enhancements (WSE) 3.0" on page 66

"IA Values" on page 57

3.2.2.1 Create Batch Mode

The Web Services subsystem enables automatic batch creation, to create a batch and insert images and data. To ensure that the WS Input module successfully receives an incoming request, creates a batch and imports pages, do the following steps:

- Place the WS Input module first in the process, and associate it with the proper *WSDL* file.
- Verify that the WS Input module is triggered at level 7 or level 0.

Note: A page is defined as a single-sided image. It is scanned or imported into an Intelligent Capture Server by modules such as ScanPlus, Standard Import (MDW and Email), WS Input, and Ricoh GlobalScan Plug-in. For every page scanned or imported in simplex mode, one page is counted. For every page scanned or imported in duplex mode, two pages are counted. Each single-sided page is counted once when it enters an Intelligent Capture system.

Each level 0 node created by the module decrements the server count by one. Since the module does not split multi-page documents into single-page documents, all multi-page documents are considered a level 0 node. The module can process a multi-page file like a *TIFF* or *PDF*. However, it does not split this file into separate pages while processing. It only counts the file as a single page for debiting the server page count license.

Related Topics

"Web Services Input" on page 24

"Sample WSDL File" on page 59

3.2.3 Web Services Output

The WS Output module is an unattended module that runs as a Windows Service. This module, which functions like a standard Export module, acts as a web services client to consume external web services. Unlike WS Input, WS Output does not interface with the Web Services Coordinator or Web Services Hosting services. In fact, this module does not include any external interface because it acts solely as an export module and communicates with an external system. If a process requires only outgoing web calls, then it is unnecessary to install the Web Services Coordinator or Web Services Hosting services. Figure 3-4 demonstrates the interaction of the WS Output module:

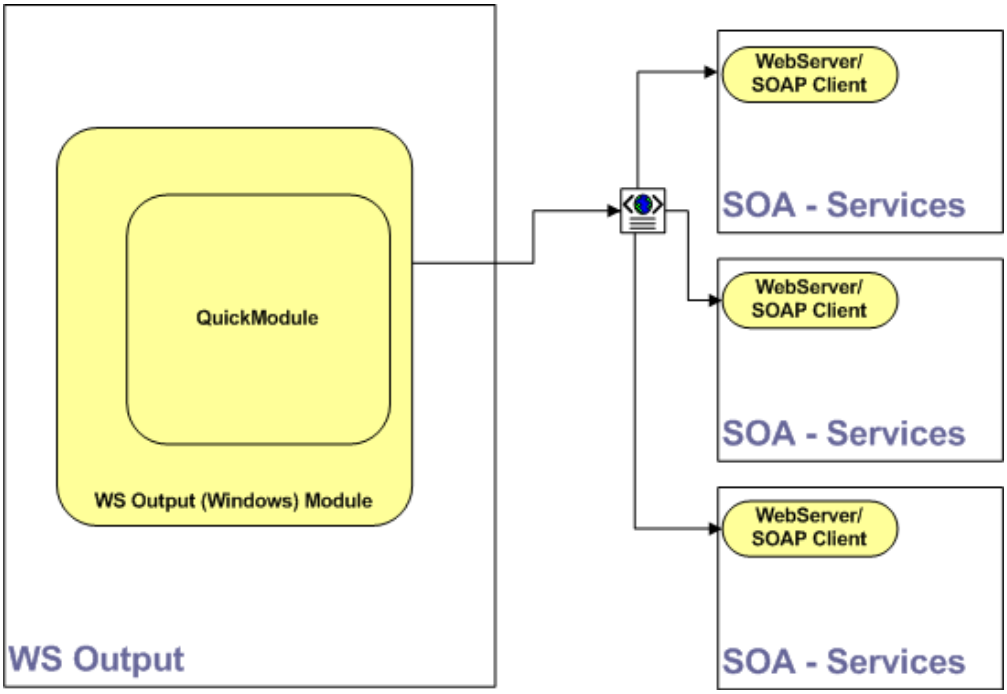

Figure 3-4: Web Services output

In setup mode, the WS Output module enables the visual mapping of IA Values in the batch to simple output parameters. Map more complex parameters, like arrays, using the scripting interface provided by this module. In addition to parameter mapping, the WS Output module also enables the configuration of *SSL* for secure connections. To provide security for these transfers, store authenticated SSL certificates on both sides of the connection. And, use "*HTTPS*" instead of "*HTTP*" in the destination *URL*. The "Setting Up Web Services Output" on page 46 section contains information on setting up the WS Output module.

Related Topics

"Authenticating Web Service Output Calls" on page 38

"Using SSL to Secure Communications" on page 41

"Mapping Web Call Parameters to IA Values" on page 47

"IA Values" on page 57

3.2.4 Web Services Coordinator

The Web Services Coordinator is a Windows service that is transparent to users. The Web Services Coordinator acts as a synchronization mechanism and establishes a correspondence between web calls and tasks.

The Web Services Coordinator uses a unique *`Correlation ID`* to provide a free-form interaction between Intelligent Capture and third-party systems. The Web Services Coordinator also enables you to isolate the hosting service from your intranet with a firewall.

The Web Services Coordinator performs load balancing on the WS Input module by synchronizing "web call – task" pairs. Synchronizing ensures that the WS Input module is only loaded with prepared tasks, in which both the trigger and web call have been received. When the WS Coordinator loses connection to a WS Input module, it reschedules the "web call – task" pair that the module is processing.

Trace Logs

Though disabled by default, you can enable tracing for each of the services in the Web Services subsystem by changing settings in the registry. Specifically, add the *`-trace`* command line parameter to the corresponding service in the Registry Editor, under **HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services**.

The Web Services Coordinator makes a trace log entry each time a WS Input module is started. It does not log anything when a WS Input module is stopped. Currently, there is no way to know that the Web Services Coordinator removed the WS Input module from the list of active modules. By default, the trace logs are stored in the following directory: `%APPDATA%\Emc\InputAccel\6.5\Logs`.

Other Considerations

A successful installation of the Web Services subsystem includes only one Web Services Coordinator service. Do not connect a single Web Services Hosting service to more than one WS Coordinator services. A sharing conflict would occur as each Web Services Coordinator would attempt to reconfigure the Web Services Hosting according to its own rules.

Related topics

"Working with Correlation IDs" on page 50

"Sample WSDL File" on page 59

3.2.5 Web Services Hosting

Web Services Hosting is a web service provider that acts as a web server for hosting web services for the WS Input module. It processes web calls addressed to the WS Input module on one or more specified ports. It does so by extracting web call input parameters as *XML*, then passing them to the Web Services Coordinator. The Web Services Hosting service then receives output parameters from the WS Coordinator and sends this response to the caller.

A successful installation of the Web Services subsystem includes only one Web Services Coordinator. Do not connect a single Web Services Hosting service to more than one Web Services Coordinator. A sharing conflict would occur as each Web Services Coordinator would attempt to reconfigure the Web Services Hosting according to its own rules.

When the Web Services Hosting service is started with the "-trace" command line parameter, the service logs critical events to the Event Log. It stores non-critical events in a log file in the following directory:

```
%ALLUSERSPROFILE%\Application Data\Emc\InputAccel
```

Note: A limitation in the HTTPListener component of the .NET Framework, prevents the Web Services Hosting service from detecting that the client closed one of its connections. In some cases, the Web Services Hosting service throws and successfully handles an exception. In most cases, the .NET Framework does not notify the Web Services Hosting service about closed connections.

For specific instructions on how to add and configure hostings, see *OpenText Intelligent Capture - Administration Guide (ECPCORE-AON)*.

Related Topic

"Adding Web Services and Hostings" on page 43

3.2.6 Web Services Administration

You can perform Intelligent Capture Server administrative tasks for the Web Services subsystem through Intelligent Capture Administrator. From this location, you can view, add, or remove web services and hostings, as well as view and change the properties of those services and hostings.

For more information on the *WS* administration pages in Intelligent CaptureAdministrator, see *OpenText Intelligent Capture - Administration Guide (ECPCORE-AON)*.

Related Topics

"Web Services Input" on page 24

"Web Services Output" on page 27

"Web Services Coordinator" on page 28

"Web Services Hosting" on page 29

"Adding Web Services and Hostings" on page 43

3.3 Setting Up Web Services

Most configuration information for Intelligent Capture client modules is defined during *module setup*. You can run a module in setup mode from Intelligent Capture Designer, from Intelligent Capture Administrator, or from a command prompt with appropriate arguments. For more information, see *OpenText Intelligent Capture - Module Reference (ECPCORE-CMD)*.

When launched in setup mode, a client module displays the **Setup** window in which you can specify the module's configuration settings.

3.3.1 Understanding Error Handling

To make processing more efficient, specify how a module must respond when an error occurs during production. Specific error handling functionality depends on whether the module is run as an attended or unattended process. Module steps in an *IPP* that are run as services or unattended client applications are considered unattended processes. There are two categories of error handling options. One option defines what to do when an error occurs during production. The other option specifies how to recover from an error.

When defining error handling options for a module step, consider how the step is defined in the IPP. For example, a step that has no associated error routine in the IPP keeps being requeued if it encounters an error in production. The task is resubmitted to the module for processing, errors out, and the cycle repeats in a loop. When error handling is not included for a step in the IPP, select **Automatically respond with the following options** on the **Error Settings** pane. Enable **Abort entire task** and **Set the batch priority to 0** to avoid this error retry cycle.

Available error handling options can vary between modules. Not all modules have error handling options available from the setup window. If the module setup window does not have an **Error** option on the navigation panel, then error handling must be defined in the IPP. Depending on the module used to create the process, see *OpenText Intelligent Capture - Designer Guide (ECPCORE-CPD)* or *OpenText Intelligent Capture - Scripting Guide (ECPCORE-PSC)* for help including an error handling routine in an IPP.

Unattended modules respond differently to error handling settings defined on the **Error Settings** pane, depending on whether they are run as an unattended client application or a service:

- When run as an unattended client application, the module recognizes all of the error handling settings defined during module setup.

- When run as a service, the module recognizes all settings except the **Prompt for an action** and **Stop receiving tasks after the current task** options. When the **Prompt for an action** option is set for a module run as a service, it is ignored and switched to **Automatically respond with the following options**. When the **Stop receiving tasks after the current task** option is set for a module run as a service, it is always ignored.

Note: Unattended modules (run as client application or service) cancel any task that was not properly configured (setup mode settings were never applied). The task is canceled with a retry reason and the batch priority is set to 0.

Not all Intelligent Capture modules can be run in unattended mode. For a list of modules that can be run unattended and detailed information on installing and configuring client modules, see *OpenText Intelligent Capture - Installation Guide (ECPCORE-IGD)*.

3.3.1.1 Setting Up Error Handling

This procedure applies to modules that have an **Error** option on the navigation panel of the module setup window and run as an application. If the module does not have an **Error** option, or if the module is run as a service, then error handling must be defined in the *IPP*. Depending on the module used to create the process, see *OpenText Intelligent Capture - Designer Guide (ECPCORE-CPD)*or *OpenText Intelligent Capture - Scripting Guide (ECPCORE-PSC)* for help including an error handling routine in an IPP.

Some error conditions cease to exist when subsequent attempts are made to process a task. For example, a temporarily locked file can cause an error that gets resolved by the time the module attempts to process the task again.

To specify error handling and recovery:

1. Run the WS Input or WS Output module for setup.
2. Select **Errors** from the navigation panel.
3. (Only for WS Input module) Select the **Abort entire the entire task** option in the **When an Error Occurs** section to abort the current task and returns an appropriate <*ExportResult*> IA Value. This IA value is evaluated by instructions in the IPP `Finish` event handler. Processing continues on the next available task in the queue.
4. (Only for WS Output module) In the **Recovery Options** section, determine how to recover from errors during production. Select the **Automatically retry** option and type or select a numeric value in the **Number of times to retry before reporting an error** field. The module retries the task the specified number of times before reporting an error. When this option and field are cleared, the module does not retry tasks that have errors.

3.3.2 Understanding Common Usage Scenarios

The topics in this section describe scenarios for integrating the modules of the Web Services subsystem into a larger network architecture.

3.3.2.1 Simple Scenario: Import

Figure 3-5 demonstrates a simple use case for the WS Input module:

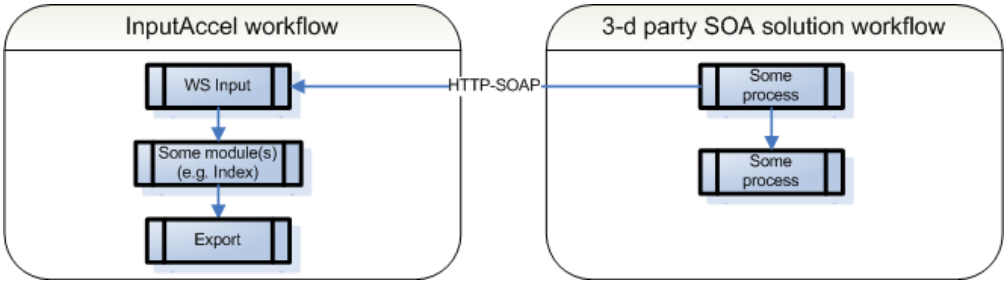

Figure 3-5: Import scenario

In this graphic, a third-party client, which is able to send *SOAP* messages, calls the WS Input module. The WS Input module creates a batch and delivers new batch information to other modules. Batch information is delivered with files attached or referenced through a *URL*. Similarly, the WS Input module can add files to an existing batch upon receiving the *`batch ID`* information through a Web Services call. The files can then be sent as an attachment or as a URL, which the WS Input module can fetch.

Note: The WS Input module handles binary data coming from web requests. However, it cannot recognize a file as an image and is not intended to manipulate images or their thumbnails. For this reason, there can be discrepancies between the images and the thumbnails in the batch, especially after running a batch in another module.

Related Topics

"Simple Scenario: Export" on page 33

"Simple Scenario: Calling a Module as a Service" on page 34

"Complex Scenario: Integrating Intelligent Capture with Other Systems through Web Services" on page 34

3.3.2.2 Simple Scenario: Export

Figure 3-6 demonstrates a simple use case for the WS Output module. In this scenario, the WS Output module initiates a call to a third-party system, enabling the export of data to an external system. TheWS Output module can be used to call any web service. Therefore, this module enables users to query documents from other *SOA* systems (likeDocumentum).

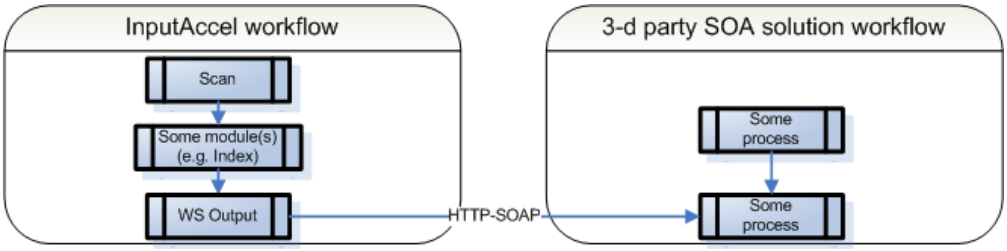

Figure 3-6: Export scenario

For example, the Brizzle Car Company builds and ships electric cars all over the world. In many countries, Brizzle Cars also manages a line of Service Centers where customers can bring their cars for repairs. Customers who visit a Service Center must fill out a Service Center Form detailing their car service history and the reason for the visit. The Brizzle Car Company wants to digitize these forms and import them into their Enterprise Content Management (*ECM*) system.

First, a process developer creates the workflow for digitizing these Service Center forms. The process developer then configures each module. The WS Output module must point to the *WSDL* of the Brizzle Car Company web service, and IA Values are mapped to web call parameters.

With the process tested, the Service Center Forms are ready for processing. Processing begins when the Scan operator creates a batch and scans the form. The completed batch is then delivered to the Index operator. The Index operator loads the module and chooses to **Run All Batches** to complete the Index fields for every form. The WS Output module runs as a service on the same machine as Index. It performs a web call for each Service Center Form to export this data to the Brizzle Car Company ECM system.

Related Topics

"Simple Scenario: Import" on page 32

"Simple Scenario: Calling a Module as a Service" on page 34

"Complex Scenario: Integrating Intelligent Capture with Other Systems through Web Services" on page 34

3.3.2.3 Simple Scenario: Calling a Module as a Service

Figure 3-7 demonstrates a simple use case showing both the WS Input and WS Output modules:

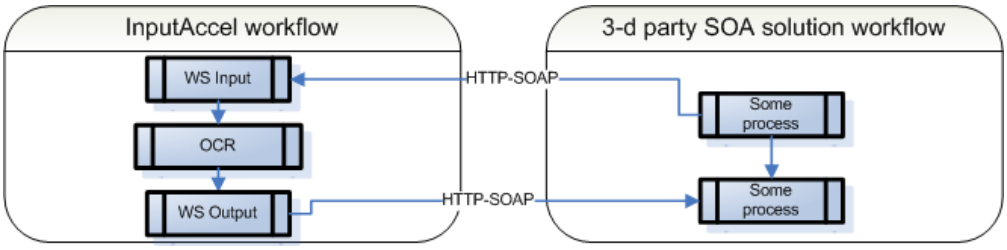

Figure 3-7: Module as a Service

In the graphic, the OCR module is exposed as a web service by using the WS Input and WS Output modules in the *IPP*. The third-party system can call the WS Input module, passing in the images (or the *URL* to images) from which data can be extracted. The WS Input module then creates a batch with these images, and sends this batch to the next step in the batch, the OCR module. The OCR module extracts the data from the images and stores this data in IA Values, before sending the batch to the WS Output module. Finally, the WS Output module makes an outgoing web service call. This call sends the extracted data from the IA Values back to the third-party system for further processing.

Related Topics

"Simple Scenario: Export" on page 33

"Simple Scenario: Import" on page 32

"Complex Scenario: Integrating Intelligent Capture with Other Systems through Web Services" on page 34

3.3.2.4 Complex Scenario: Integrating Intelligent Capture with Other Systems through Web Services

Figure 3-8 demonstrates how a complete *SOA* solution can be implemented using the Web Services subsystem of Intelligent Capture. ABC Bank is building an SOA solution for its credit department. After installing the Intelligent Capture Server and SQL Server, the administrator deploys and configures the Web Services infrastructure. Next, a platform developer at ABC Bank develops the following business process for bank credit requests:

1. Receive a fax with Credit Form from local office by fax.
2. Perform *OCR* for indexing by *ECM* system.
3. Extract the Social Security Number (*SSN*) from the form.

4. Ask Credit History Bureau for client credit history.
5. While the credit history processes its request, extract client signature from a form.
6. Save client signature in a Clients Database.
7. Receive client credit history from Credit History Bureau.
8. Save client credit history and form to ECM.
9. Analyze and decide whether to provide a bank credit.
10. Send a fax with decision to the local office.

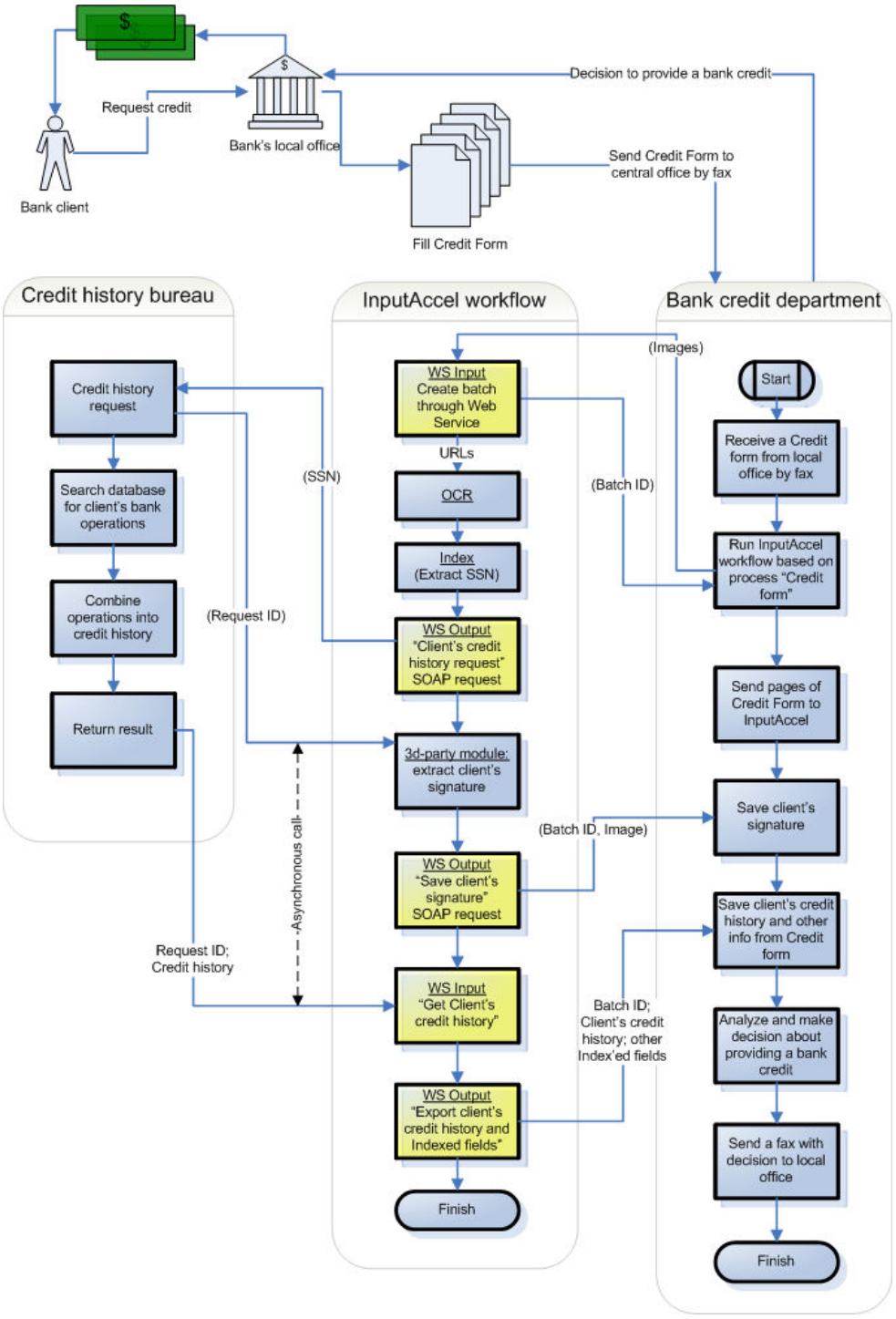

Figure 3-8: Integrating Intelligent Capture with Other Systems through Web Services

After analyzing these business process steps, the platform developer decides that Intelligent Capture must perform Steps 2–5 and 7. Next, the developer creates a process with the following steps:

1. **Import**: Uses the WS Input module to import Credit Form pages.
2. **OCR**: Uses the NuanceOCR module to perform OCR on the Credit Form.
3. **SSN**: Uses the Index module to extract the Social Security Number.
4. **CHRequest**: Uses the WS Output module to send a web service request to the Credit History Bureau.
5. **Signature**: Uses a custom module from ABC Bank to extract the client signature as an image.
6. **StoreSig**: Uses the WS Output module to store the client signature in the ABC Bank database.
7. **CHResponse**: Uses the WS Input module to receive credit history from the Credit History Bureau.
8. **Export**: Uses the WS Output module to export the client credit history and OCR-scanned form to ABC Bank ECM system.

After compiling and installing this process on the Intelligent Capture Server, the platform developer proceeds to configure the modules in the process.

In production mode, the process works smoothly without much user interaction. The process begins when the ABC Bank Fax Receiving service receives a Credit Form by fax. The Fax Receiving service sends the contents of the fax to the Web Service. As a result, a batch is created that the WS Input module can fill.

Steps 2 and 3 are performed automatically. Then, the WS Output module performs a web call to the Credit History Bureau and stores the <*RequestID*> in the appropriate IA Value. The process continues through Step 5. The process then triggers the WS Output module in Step 6 to perform a web call to ABC Bank web service. This service can then store the client signature in the bank signature database.

Next, the WS Input module in Step 7 alerts the Web Services Coordinator to wait for a web call using the same <*RequestID*> for identification. Upon receiving this web call and verifying the <*RequestID*>, the Web Services Coordinator pushes the corresponding task back to the WS Input module. The WS Input module then copies the appropriate credit history from the web call parameter to an IA Value.

When the last step is triggered, the WS Output module calls ABC Bank web service to export the data back to the ABC Bank ECM system. Data includes the SSN, Form Number, Credit History, and the OCR-scanned form. The ABC Bank ECM system can then process this data in accordance with its internal workflow. It produces a decision to either approve or decline the bank credit. Finally, the ABC Bank Fax Send web service delivers the decision to the bank local office.

Related Topics

"Web Services Input" on page 24

"Web Services Output" on page 27

"Simple Scenario: Export" on page 33

"Simple Scenario: Import" on page 32

"Simple Scenario: Calling a Module as a Service" on page 34

3.3.3 Securing Web Services Communications

This section includes three security scenarios available in the Web Services subsystem. It also includes a step-by-step procedure for creating and installing an *SSL* certificate on a service host. The scenarios listed here build on the ABC Bank scenario detailed previously in "Complex Scenario: Integrating Intelligent Capture with Other Systems through Web Services" on page 34.

3.3.3.1 Authenticating Web Service Output Calls

The Web Services subsystem provides secure communications through the Secure Sockets Layer (*SSL*) protocol. SSL is designed to provide secure, encrypted connections and server authentication. SSL uses certificates installed on a server with a private key and a client. The client can download this private key from the server when establishing a connection. The client must trust this certificate to authenticate the server, in one of two ways:

- The server certificate is installed into the client trusted certificates storage.
- A trusted vendor signs the server. The client has the vendor certificate installed in trusted storage. In this case, the client can download the server certificate and verify its signature. If a trusted vendor properly signs the certificate, then the client trusts the server certificate.

Communication between client and server is handled through an SSL connection, using *HTTPS* instead of *HTTP* in the *URL*. After the server is authenticated, a secure connection is established and the client is authenticated, using a standard user name and password. No additional encryption is required for credentials, because the SSL connection is already encrypted. This technique enables you to store the client-side username and password as parameters in a web call.

Related Topics

"Simple Scenario: Export" on page 33

"Simple Scenario: Calling a Module as a Service" on page 34

"Authenticating Web Service Input Calls" on page 39

"Authenticating Asynchronous Calls" on page 40

3.3.3.1.1 Conflicts when Authenticating Web Service Output Calls

Web Services Output uses TLS and SSL protocols to secure the HTTPS connection. By default Web Services Output uses auto detection to choose the protocol to use. Occasionally, in some environments, there may be a conflict on the protocol to use. When this happens, an operation timed out error occurs when Web Services Output attempts to call a web service. To fix this error, administrators must configure the `QuickModuleHost.exe.config` file and specify the protocol to use.

To resolve conflicts when using TSL or SSL protocols:

1. Open the `QuickModuleHost.exe.config` file from `Program Files\InputAccel\Client\binnt` (or Program Files (x86) for 64-bit systems) using Notepad.
2. Add a `SecurityProtocol` key to the **appSettings** tags and set the value to one of the following:
   - `Ssl3|Tls` (by default)
   - `Ssl3`
   - `Tls` (for TLS v1.0)
   - `Tls11` (for TLS v1.1)
   - `Tls12` (for TLS v1.2)

   For example:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
<appSettings>
<add key="SecurityProtocol" value="Ssl3" />
</appSettings>
</configuration>
```

3. Save the `QuickModuleHost.exe.config` file.

3.3.3.2 Authenticating Web Service Input Calls

This security scenario involves authenticating both the client and server for a *WS* Input call. This scenario uses the same authentication technique as in the previous scenario, except that a third-party system calls the WS Input module. The *SSL* certificate must be installed on the server that exposes the Web Services Hosting service. It is recommended that an application shipped with supported Windows operating systems, called `netsh.exe`, assign a certificate to a port. Once implemented, the *HTTP* driver receives SSL connections on this port and uses the installed certificate to authenticate the server and establish a secure connection.

A simple user name and password combination is then used to authenticate the third-party client system. Because different validation providers can be used on the client system, the Web Services subsystem has no predetermined authentication techniques for this task. However, the client-side scripting functionality of the WS Input module can access incoming parameters, perform validation, then process or

deny the call. The *Web Services Scripting API Reference* contains more information on the Web Services *API*.

Related Topics

"Simple Scenario: Import" on page 32

"Simple Scenario: Calling a Module as a Service" on page 34

"Authenticating Web Service Output Calls" on page 38

"Authenticating Asynchronous Calls" on page 40

3.3.3.3 Authenticating Asynchronous Calls

The final security scenario describes an asynchronous call, consisting of two separate outgoing, and incoming calls. The authentication of these calls can be managed separately for the WS Input and WS Output modules, as detailed in the preceding topics. However, the Web Services subsystem offers another way of authenticating incoming calls. As shown in "Complex Scenario: Integrating Intelligent Capture with Other Systems through Web Services" on page 34, the authentication process for the ABC Bank example includes the following steps:

1. The WS Output module establishes an *HTTPS* connection with the Credit History Bureau and authenticates this using *SSL*.
2. The WS Output module sends a request to the Credit History Bureau with credentials that enable the Credit History Bureau to authenticate the caller.
3. The Credit History Bureau validates the credentials. Now both the client and server have been authenticated.
4. The Credit History Bureau returns the *`Correlation ID`*.
5. The Credit History Bureau processes this request.
6. The Credit History Bureau establishes a secure HTTPS connection to the WS Hosting service, and authenticates it using SSL.
7. The Credit History Bureau sends a response (with the *`Correlation ID`* and credentials) that enables the WS Input module to authenticate the caller.
8. The Web Services Hosting service routes the response to the WS Coordinator service, which uses the *`Correlation ID`* to route the response to the appropriate batch.
9. The WS Input module validates the credentials. Now both the client and server are authenticated.
10. The WS Input module processes the response.

These procedures ensure that the modules of the Web Services subsystem properly route and receive web calls. Web calls are to and from the Credit History Bureau as well as between other Web Services modules. ABC Bank and the Credit History Bureau both want to have confidence that their systems interface to pass data

securely. They can achieve it by implementing SSL, using a *Correlation ID*, and standard user name/password validations.

Related Topics

"Complex Scenario: Integrating Intelligent Capture with Other Systems through Web Services" on page 34

"Authenticating Web Service Output Calls" on page 38

"Authenticating Web Service Input Calls" on page 39

"Working with Correlation IDs" on page 50

3.3.3.4 Using SSL to Secure Communications

The Web Services subsystem uses *SSL* for authentication and secure communication between the web service provider and consumer. This section provides an example on how to create a self-signed certificate, install the certificate, and bind the SSL certificate to a given *HTTP* port.

Notes

- This example demonstrates how to create a self-signed certificate for Windows Server 2008. A self-signed certificate is for a test environment only. For production environment, the SSL certificate must be obtained from an approved Certification Authority. IIS 6.0 Resource Kit Tools is required to create a self-signed certificate on Windows Server 2008. When installing this kit, choose the **Complete** installation option or verify that the **Custom** installation installs the Self SSL feature to generate the self-signed SSL certificate.
- To complete the following procedure successfully, generate a *UUID*. Many online sites offer free UUID generation, including: http://www.opensimulator.fr/staticpages/index.php/UUIDgenerator.

To create and install a self-instructed certificate on the service host:

1. Obtain an SSL certificate from an approved Certification Authority for your production environment. For test environments, you can create a self-signed certificate. Use these steps to create a self-signed certificate for Windows Server 2008:

   a. First, identify the ID of the site where to install the certificate. For example, if the name of the service host machine is <*PC1*>, then execute the following command from a command prompt:

   ```
   iisweb.vbs /query "PC1"
   ```

   This returns a site name of the following format: PC1 (W3SVC/12345). In this example, the relevant site ID is *12345*.

   b. Execute the following command from a command prompt:

```
selfssl.exe /N:CN=PC1 /V:10 /S:12345 /P:444
```

where:

`/N:CN=<Your Machine Name>` or the common name of the certificate

`/K:1024` represents the key length of the certificate

`/V:7` signifies the validity of the certificate in days

`/S:1` represents the ID of the site to which the certificate is installed

`/P:444` sets the SSL port

c. Type `mmc.exe` from the command prompt to invoke the Microsoft Management Console (MMC).

d. From the menu, select **File -> Add/Remove Snap-in...** to display the **Add/Remove Snap-in** window.

e. Click the **Add...** button. The **Add Standalone Snap-in** window displays.

f. Select **Certificates**, and click **Add**.

g. On the **Certificates Snap-in** window, select **Computer Account**, and then click **Next**.

h. On the **Select Computer** window, select **Local Computer**, and then click **Finish**.

i. Click **Close** on the **Add Standalone Snap-in** window.

j. Click **OK** on the **Add/Remove Snap-in** window to confirm the new Snap-in.

2. Install the certificate on the service host:

a. From the Console Root, open **Local Computer -> Personal -> Certificates** to display the newly added certificates.

b. Select both certificates, and then press **CTRL+C** to copy them to the Clipboard.

c. Open **Local Computer -> Trusted Root Certification Authorities -> Certificates**, and then press **CTRL+V** to paste these certificates into this directory.

3. Bind the SSL to a port using netsh.exe. The syntax of this command varies for different environments, but you must have your newly generated UUID. For more specific information regarding the exact syntax in your environment, see Microsoft TechNet Library (https://technet.microsoft.com/en-us/library). For example, at the command prompt, type:

```
httpcfg set ssl /i 255.255.255.0:444 /g"{ eba59598-a8c8-4b05-ba16-d734d8b3bbe8}"
```

Related Topics

"Complex Scenario: Integrating Intelligent Capture with Other Systems through Web Services" on page 34

"Authenticating Web Service Output Calls" on page 38

"Authenticating Web Service Input Calls" on page 39

3.3.4 Adding Web Services and Hostings

To process requests from a third-party web service client, first add and register the web service within the Web Services subsystem. To add a web service, define the service, map a *`Correlation ID`* for asynchronous processing, and then register the service within the Web Services subsystem. For more information, see *OpenText Intelligent Capture - Administration Guide (ECPCORE-AON)*.

Notes

- If the WS Input module uses this service to create batches (as the first module step in a process), do not map a *`Correlation ID`*. Only methods that do not have a *`Correlation ID`* mapped to them are present in the **Mapped method** list. If all methods are mapped, then the service is not present in the **Service for mapping** list.
- If the WS Input module uses this service to add data to an existing batch (in a position other than the first module step in a process), map a *`Correlation ID`*. Map at least one method to a *`Correlation ID`*. Only methods that have a *`Correlation ID`* mapped to them are present in the WS Input module's **Mapped method** list. If no methods are mapped, then the service is not present in the **Service for mapping** list.

Adding a web service hosting sets up an instance of a web server to handle requests from and responses to third-party web service clients. Part of adding a hosting is publishing one or more web services that have been registered within the Web Services subsystem. Add multiple web service hostings as needed (one per machine name). Multiple hostings can be necessary to handle a high request/response load or to isolate external (internet) requests and responses from internal (intranet) requests and responses.

Note: Do not add a hosting or change the hosting configuration while processing a web request. The Web Services Hosting service would reject the connection with the web client. It would also abort all connections affected by these changes.

To add a hosting, define the machine that acts as a web server and select the web services to publish on the hosting. Then, register the hosting within the Web Services subsystem. For more information, including the steps for adding and configuring a Web hosting, see *OpenText Intelligent Capture - Administration Guide (ECPCORE-AON)*.

Related Topics

"Understanding Web Services Configuration" on page 23

"Web Services Hosting" on page 29

"Mapping Web Call Parameters to IA Values" on page 47

"Working with Correlation IDs" on page 50

3.3.5 Setting Up Web Services Input

The WS Input module can be run in setup mode (for configuring parameters) or as a service. To perform parameter mapping, run the module in setup mode. With limited exceptions, the **Mapping** window described in Mapping Web Call Parameters to IA Values enables mapping of simple web call parameters to IA Values. For mapping more complex parameters, use the Client-Side Scripting functionality available on the **Scripting** pane.

Permissions for this Module

Configure the WS Input module to read and write data for the process, the batch, the module, and other basic security options. This configuration enables the module to execute tasks successfully in your environment. For example, if you intend to use naming schema tags, the module must have System.ProcessModify permissions; or, if the WS Input module is the first step in the process, the module must have System. SecurityRead permissions. The table next lists the permissions applicable to this module and their purpose in the system:

Table 3-1: Permissions for the WS Input module

Permission	Description
Server.Login	Log in to the Intelligent Capture Server.
Server.Read.Module.Data	Read module data.
Server.Write.Module.Data	Write module data.
Server.Create.Batch	Create or modify a new batch.
System.BatchModify	Write or delete batch data.
System.BatchRead	Read batch data.
System.ServerRead	Read non-module server data, such as registry values.
System.ProcessRead	Read process data.
System.ProcessModify	Change process data, for example, when using naming schema tags.
System.SecurityModify	Write *ACL* security data.

Configuring the WS Input Module

The following procedure describes how to configure the WS Input module for setup.

To configure the WS Input module for setup:

1. Run WS Input for setup.
2. In the **Service for mapping** combo box on the **WS Input** pane, choose the web service that receives the web call for this instance.

   Note: The *Correlation ID* can be set in Intelligent Capture Administrator. For more information on how to configure this value, see *OpenText Intelligent Capture - Administration Guide (ECPCORE-AON)*.

   - If the WS Input module uses this service to create batches, (as the first module step in a process), do not map a *Correlation ID*. Only methods that do not have a *Correlation ID* mapped to them are present in the **Mapped method** list. If all methods are mapped, then the service is not present in the **Service for mapping** list.
   - If the WS Input module uses this service to add data to an existing batch (in a position other than the first module step in a process), map a *Correlation ID*. Map at least one method to a *Correlation ID*. Only methods that have a *Correlation ID* mapped to them are present in the WS Input module's **Mapped method** list. If no methods are mapped, then the service is not present in the **Service for mapping** list.
3. If the WS Input module is the first module in the current process, the **Batch name schema** field is visible in this window. Enter a format for the automatic batch naming schema. Note that you can only set the **Batch Name Schema** when the WS Input module is the first step in the process.
4. If the WS Input module is not the first module in the current process, the **Asynchronous task timeout in seconds** selection box displays. Leave the default value of 86400 or change, if necessary.
5. In the **Mapped method** combo box, select the web method that receives the web call for this instance. For each method, map method parameters to IA Values. Select a mapped method from the combo box, and then click **Mapping**. The **Mapping** window displays.
6. In the **Mapping** window, link IA Values to the incoming parameters of web methods and link parameters of outgoing web methods to IA Values. The Mapping Web Call Parameters to IA Values section contains more detailed instructions on how to perform parameter mapping.
7. Click **OK** to save the mappings and return to the **WS Input** pane.
8. Click **OK** to save these changes.

Note: The WS Input module only supports the **Automatically abort entire task** error setup option on the **Error** pane. It does not support the **Automatically retry** feature.

Related Topics

"Web Services Input" on page 24

"Working with Correlation IDs" on page 50

"Mapping Web Call Parameters to IA Values" on page 47

"IA Values" on page 57

"Configuring Automatic Batch Creation" on page 52

"Understanding Web Services Configuration" on page 23

3.3.6 Setting Up Web Services Output

The WS Output module can be run in setup mode (for configuring parameters) or as a service. To perform parameter mapping, run the module in setup mode. With limited exceptions, the **Mapping** window described in Mapping Web Call Parameters to IA Values enables the mapping of simple web call parameters to IA Values. For mapping more complex parameters, use the Client-Side Scripting functionality available on the **Scripting** pane.

To configure WS Output for setup:

1. Run WS Output for setup.
2. In the **WSDL URL** field, enter the full path to the *WSDL*, in one of the following formats:
   - `c:\somepath\sample.wsdl`
   - `http://somesite/somedir/sample.wsdl`
   - `http://somesite/somedir/sample?wsdl`
   - `\\somecomputer\somedir\sample.wsdl`
3. Click the **Parse** button to download the WSDL and populate the **Service name** and **Method Name** combo boxes. Verify that the correct service is displayed.
4. To enable *MTOM* usage for sending files, select the **Use (MTOM) to send files** check box. The Sending Files as MTOM attachments section contains more information.
5. Leave the default value for the **Request timeout in seconds**, or change if necessary. This value represents the amount of time the module waits for a web response before processing fails.
6. To begin mapping parameters, select a method from the **Method Name** combo box, and click the **Mapping...** button.
7. In the **Mapping** window, link IA Values to the incoming parameters of web methods and link parameters of outgoing web methods to IA Values. The

Mapping Web Call Parameters to IA Values section contains a list of mapping rules and more detailed instructions on how to perform parameter mapping.

8. Click **OK** to save these mappings and return to the **WS Output** pane.
9. Click **OK** to save these changes.

Note: This module supports the **Recovery options** and the **Automatic retry** error setup options on the **Error** pane. All other error setup options are ignored.

Related Topics

"Web Services Output" on page 27

"Mapping Web Call Parameters to IA Values" on page 47

"IA Values" on page 57

"Sample WSDL File" on page 59

"Sample MTOM-encoded File Upload" on page 60

"Sending Files as MTOM Attachments" on page 52

"Understanding Web Services Configuration" on page 23

3.3.7 Mapping Web Call Parameters to IA Values

Map IA Values to web call parameters to ensure that these values remain intact as they pass from module to module and across the network. Mapping is allowed only for IA Values from the trigger level and higher, but not for any child nodes of the trigger node. For example, if a module is triggered at level 7, then only level 7 variables are available for mapping. If the WS Input module is triggered at level 0, then all the IA Values at all levels are mappable.

Note: You can leave web call parameters unmapped when, for example, the WS Input module is the first step in the process. In this case, the Client-Side Scripting interface handles the web call parameters.

The Mapping Window

The **Mapping** window is composed of three parts. On the left is the hierarchical list of IA Values for the current process. On the right is the hierarchical list of parameters for the corresponding web method (as obtained from the *WSDL* file). In the center of the window is the **Link zone**, for drawing mapped links between IA Values and web call parameters. You can perform mapping by dragging-and-dropping IA Values and parameters, or with the help of the keyboard shortcuts listed next.

The hierarchical list of IA Values for the current process is on the left side of the **Mapping** window. Each level node represents IA Values for a particular level, with

Level 7 as root, Level 6 as its child, continuing down to Level 0. Each level node includes a child node called `ID` that represents the `node ID`. At Level 7, this `node ID` is the `Batch ID`. Below this `node ID`, each Level node includes child nodes for every module instance in the process. Each module instance node contains child nodes representing IA Values declared in the module *MDF* on this level.

On the right side of the **Mapping** window, the hierarchical list of parameters for the corresponding web method (obtained from the specified WSDL) includes two main nodes: **In** and **Out**, representing request and response web method parameters, respectively. Each parameter in the list also includes its data type, as well as an icon that specifies whether this parameter is simple or complex. The meaning of these nodes is different for the WS Input and WS Output modules. For the WS Input module, the set of parameters marked **In** represent request parameters: this data is exported to the module. The set of parameters marked **Out** represent response parameters which are imported from the module. For the WS Output module, the set of parameters marked **In** represent request parameters: this data is exported from the module. The set of parameters marked **Out** represent response parameters which are imported to the module.

Simple and Complex Parameters

For each parameter and node in these hierarchical lists of the **Mapping** window, a small icon specifies whether this parameter is simple or complex. The icon for simple parameters (like *`client_name`* or *`priority`* in this graphic) resembles a single blue box. The icon for complex parameters (like *`oRescanRequest`*) resembles a stack of three blue boxes. Simple parameters can be mapped directly to an appropriate IA Value. To map a complex parameter, map its subparameters to appropriate IA Values, as shown in this example.

Mapping Rules

Mapping provides automatic type conversion to enable, for example, the mapping of integers to strings. When adding a mapping, keep in mind the following mapping rules:

- Setting the value of a web call parameter above the limit of its value type causes an error.
- Simple parameters can be mapped directly to an IA Value.
- To map a complex parameter, map its subparameters to IA Values.
- IA Values of type Date are supported in mapping. The following rules apply when mapping Date IA Values to and from *WSDL* DateTime types:
  - Mapping WSDL DateTime toDate in setup mode displays a warning about unsafe conversion because Date does not support time zones.
  - Conversion from WSDL DateTime toDate ignores time zone information in the DateTime value, and saves only the date and time without a time zone specification.

  - Conversion from Date to WSDLDateTime results in a date/time value with no time zone specified.
  - Conversion from WSDL DateTime to string preserves time zone information by converting the date/time data format into the format specified by the locale of the machine hosting the module.
  - Conversion from string to WSDL DateTime preserves the time zone information (if present).
  - Conversion from string to Date ignores the time zone (if present) and is valid only if the format of the time in the string matches the format specified by the regional settings of the machine hosting the module.

- Web-call parameters of type String can be mapped directly to IA Values of type File. This mapping can be useful if the String parameter contains a *URL* that must be fetched and stored in the mapped IA Value of type File. If the String does not contain a valid URL or path, and the string is not Base64 encoded, then the module errors. In this case, the value of the string is not written to the corresponding IA Value file. If the String is empty, then the WS Input module does not create a file or generate an error.

  Note: Most modules do not validate fixed string lengths. Data can be lost.

- If the WS Input module is not the first step in the process, map the web call parameter *`Correlation ID`* to an IA value. If the WS Input module is the first step in the process, the *`Correlation ID`* is not necessary. For more information on how to configure the *`Correlation ID`*, see *OpenText Intelligent Capture - Administration Guide (ECPCORE-AON)*.

This screen also supports the following shortcuts:

Table 3-2: Mapping Shortcut Keys

Shortcut	Conditions	Result
CTRL+L	When focus is on an IA Value element and a method parameter.	Creates a mapping link between the IA Value and the method parameter.
CTRL+U	When focus is on an element with a mapping link.	Removes the link for the selected element.
CTRL+R	Anytime	Removes all links.
CTRL+TAB	Anytime	Switches focus between the IA Values and Method's parameters hierarchies.

The process of mapping IA Values to web call parameters provides limited functionality. To perform additional actions on web calls, run the *WS* Input module in setup mode to access the client-side scripting functionality.

Note: If mapping is not configured using the user interface, then the *NoWSIMapping* parameter is added to the response object.

Related Topics

"Web Services Input" on page 24

"Working with Correlation IDs" on page 50

"Sample SOAP Request with Correlation ID in SOAP Header" on page 60

"IA Values" on page 57

3.3.7.1 Working with Correlation IDs

Some systems have many components sending and receiving data across a network. For these systems, it can be necessary to match an incoming response to a previous, outgoing request. Suppose that System A sends a message to System B, and a response is required. System A must be able to recognize and accept the correct response from System B when it arrives. To serve this purpose, a unique number, called the *Correlation ID*, is added to all request and response messages. System A then sends a message to System B, with some set of common parameters, plus a *Correlation ID*. Upon receiving this message, System B stores this ID, then returns this value with the response. System A can then correlate the request and response to ensure the accuracy of the data.

Note: The *Correlation ID* is a case-sensitive parameter that can take any form, including a unique number, a *GUID*, or any other flat value of any type. Intelligent Capture or any third-party system can generate this parameter.

In a web call, the *Correlation ID* can be stored in a parameter or in the *SOAP* header. When mapping a *Correlation ID* stored in a parameter, point to the parameter holding the ID. When mapping a *Correlation ID* stored in a SOAP header, point to the name of the header, because *WSDL* does not describe SOAP headers.

To handle an incoming web call successfully, point to the IA Value that stores the *Correlation ID*. If the WS Input module is the first module in the Intelligent Capture process, then this association is used to store the *Correlation ID* in an IA Value. Otherwise, this association is used to find a batch that correlates with the incoming call.

To handle an outgoing web call successfully, point to the IA Value that stores the *Correlation ID*. Then, this value is copied from the IA Value to the parameter or the header of the web call that stores the *Correlation ID*.

If the WS Input module is the first step in the process, it cannot interoperate with a Web Method that contains a defined *Correlation ID*. If the WS Input module represents any other step in your process, it can interoperate with a Web Method that contains a defined *Correlation ID*.

Each Web Method can be used with only one process where the WS Input module represents the first step. If you configure WS Input as the first step in a process, you can choose a method used in another process where the WS Input module is the first step. But in this case, the old process would no longer work with this web method.

Related Topics

"Web Services Input" on page 24

"Mapping Web Call Parameters to IA Values" on page 47

"Sample SOAP Request with Correlation ID in SOAP Header" on page 60

3.3.7.2 Sending Files Across the Network

The Web Services subsystem offers two mechanisms for sending files across the network when making a web call: sending a *URL* link to the file, or sending the file as an attachment.

Sending a URL link in the *SOAP* packet of a web call enables faster network transmission and faster processing in the Web Services subsystem. This method is appropriate if the files are hosted on a web client with an easily accessible web server and the web service can download them with a URL. This method is impossible if the files are hosted on a web client behind a *NAT*.

The solution to performing web calls from behind a NAT is to send files as attachments, rather than as URL references. This method is much slower and requires more system memory for processing. It significantly affects the performance of the Web Services Coordinator, which must store large amounts of data in the database.

Note: Sending large files as attachments can cause significant network congestion issues. Therefore, do not transmit large files this way. Instead, send all files through URL web parameters. However, if sending large files as attachments is the best solution on your network, increase your system memory to 1*GB*.

Related Topics

"Sample MTOM-encoded File Upload" on page 60

"Web Services Enhancements (WSE) 3.0" on page 66

"Sample SOAP Request with Correlation ID in SOAP Header" on page 60

"Mapping Web Call Parameters to IA Values" on page 47

3.3.7.2.1 Sending Files as MTOM Attachments

WSE 3.0 enables a client and a web service to communicate. They can use large amounts of data, such as an image file, and can interoperate smoothly with Web Services specifications. To send large amounts of data (over 4*MB*), *WSE* supports the *SOAP* Message Transmission Optimization Mechanism (*MTOM*) specification. When WSE 3.0 is configured to send or receive SOAP messages that comply with (MTOM), it transfers this data as part of the SOAP envelope. To send large amounts of data, specify that the web service takes a parameter or return type that is, or contains, a byte array. In this way, the data is encrypted (or digitally signed) without the use of Base64 encoding, reducing the size of the SOAP message.

Note: Sending large files as attachments can cause significant network congestion issues. For this reason, it is not recommended to send large files this way; instead sending all files through *URL* web parameters. However, if you determine that sending large files as attachments is the best solution on your network, increase your system memory to 1*GB*.

Related Topics

"Sample MTOM-encoded File Upload" on page 60

"Web Services Enhancements (WSE) 3.0" on page 66

"Sample SOAP Request with Correlation ID in SOAP Header" on page 60

3.3.7.3 Configuring Automatic Batch Creation

When used as a batch creation module, WS Input uses naming schemas to automate batch naming. Automatic batch naming is required because the module runs only as a service—no operator intervention is possible.

A batch naming schema is a naming pattern consisting of literal text, schema keys, and certain IA Values. Literal text is used in batch names exactly as entered during setup. Schema keys are placeholders that are replaced with the value they represent at the moment a new batch is created. IA Values are variables that also are replaced with the value they represent in the same way as schema keys. However, any IA Values used must contain valid data at the time the batch is created. The batch naming schema must produce a unique batch name each time the module creates a batch. If it produces a duplicate name or an empty string, the module does not create a batch and stops processing.

If a batch naming schema includes only the schema key to insert the date, the module creates only a single batch on a given date. The second batch it attempts to create on the same date has an identical name. Therefore, batch creation fails. If the schema includes schema keys to insert the date and time or the batch index, the resulting batch name is unique. In this example, the schema can be that time changes every second and the batch index is incremented with each new batch.

To configure automatic batch creation

1. Run the module for setup.
2. Select the **WS Input** option from the navigation panel.
3. In the **Batch name schema** field, type a combination of literal text and batch naming schema keys that produce unique batch names. The following table provides a description of all available batch naming schema keys supported by the module.

Table 3-3: Batch Naming Schema Keys

Schema key	Description
<@(Date)>	Outputs the current date in the form of *<YYYYMMDD>*, where: • *<YYYY>* is the four digit year • *<MM>* is the current month • *<DD>* is the current date To change the format of *<@(Date)>* and *<@(Now)>*, add a format string to the schema key as shown in the following examples: • *<@(Date,MM-DD-YYYY)>*: Outputs the date in the format of month-day-year (for example, 12-31-2013). • *<@(Date,YYYYY)>*: Outputs the date as a four-digit year followed by the integer day of the year (for example, 2013365). • *<@(Date,YYYY)>*: Outputs the date as a four-digit year (for example, 2013). • *<@(Date,YYY)>*: Outputs the date as a two-digit year followed by the integer day of the year (for example, 13365). • *<@(Date,YY)>*: Outputs the date in the format of a two-digit year (for example, 13). • *<@(Date,Y)>*: Outputs the date as an integer day of the year, 1–366 (for example, 365)

Schema key	Description
<@(Index<[,num]>)>	Returns a unique, sequential, increasing integer value within a single Intelligent Capture Server, beginning with 1. **Note:** `@(Index<[,num]>)` does not create a unique integer value among Intelligent Capture Servers in a ScaleServer group. Use the `@(Index<String>)` to create a unique, increasing integer value among Intelligent Capture Servers in a ScaleServer group. You may format the value to pad the number with leading zeros. For example: • `@(Index,OO)` produces, 01, 02, 03. • `@(Index,OOOO)` produces 0001, 0002, 0003. Padding does not limit the number of digits in the value. It only adds leading zeros to fill the specified number of digit placeholders. In the first example above, the number continues to increment, if necessary, past 99 to 100, 101, 102, etc. To generate a unique index number using the `@(Index)` key, the operator must have permissions to modify the process (**System.ProcessModify** permission) used to create the batch. If the operator does not have these permissions, the module will not be able to update the index value, which is stored in the process file.

<table>
<tr><th>Schema key</th><th>Description</th></tr>
<tr><td><@(Index<string>)></td><td>Returns a unique, increasing integer value among all Intelligent Capture Servers in a ScaleServer group. The value produced is a unique, increasing number, but not sequential by default. <code>String</code> represents a user-defined alphanumeric string containing no spaces (underscores are allowed).

Examples

You have <code>@(IndexABC)</code> as part of your automatic batch creation naming schema and you want to reset the key value to 1 each day using IATimer. You can use the key string <code>$module=IA__EDITABLEVALUES/BatchSchemaIndexABC</code> where <code>ABC</code> is a user-defined alphanumeric string containing no spaces (underscores are allowed).

Your company uses three scanner workstations to capture accounting documents. The accountants use a process called “Expense Reports” to capture documents, and each batch based on this process is named “Expense Reports”. You can configure the capture step for “Expense Reports” to use the schema key <code>@(IndexAcct1)</code> to differentiate batches by typing <code>Expense Reports @(IndexAcct1)</code> in theProcess schema list box from the Auto Batch Creation pane.

Each time a new batch is created using the process, the batch name contains an index number. For example:

<table>
<tr><th>Scanner station</th><th>Batch name</th></tr>
<tr><td>1</td><td>Expense Reports 1</td></tr>
<tr><td>1</td><td>Expense Reports 2</td></tr>
<tr><td>2</td><td>Expense Reports 3</td></tr>
<tr><td>1</td><td>Expense Reports 4</td></tr>
<tr><td>3</td><td>Expense Reports 5</td></tr>
<tr><td>2</td><td>Expense Reports 6</td></tr>
</table>

Format the index number using 0 as a placeholder. For example: <code>@(IndexAcct1,00)</code> produces, 01, 02, 03, etc. <code>@(IndexAcct1,0000)</code> produces 0020, 0021, 0022.</td></tr>
</table>

Schema key	Description
<@(Machine)>	Outputs the name of the machine running the WS Input module that created the batch.
<@(ModuleInstance)>	Outputs the Windows services name of the WS Input module that created the batch.
<@(Name)>	Outputs the name of the process being used to create the batch. This value is constant; therefore, this schema key by itself will not generate unique batch names.
<@(Now)>	Outputs the current date and time in the form of <YYYYMMDD-HHMMSS>, using the same rules and formatting overrides as <@(Date)> and <@(Time)>.
<@(Server)>	Outputs the machine name of the Intelligent Capture Server that owns the batch.
<@(Time)>	Outputs the current time in the form of <HHMMSS>, where: • <HH> is the current hour in a 24-hour clock • <MM> is the current minute • <SS> is the current second When auto-naming with <@(Time)>, the batch name uses the time when scanning starts, not the time when the batch is created. To change the format of <@(Time)> and <@(Now)>, add a format string to the schema key and rearrange the placeholders. For example: <@(Time,SS-MM-HH)> outputs the time using the format of seconds-minutes-hours.

3.4 Running Web Services in Production

After the module has been set up, tested, and placed in a production environment, operators and administrators will run the module in production.

For information related to common production tasks for unattended modules, see *OpenText Intelligent Capture - Module Reference Help (ECPCORE-H-CMD)*.

Note: *OpenText Intelligent Capture - Module Reference Help (ECPCORE-H-CMD)* contains production topics that are common to many unattended modules, which might not apply to this module.

3.5 Reference—Web Services

The topics within this section contain reference information useful while using the application in setup or production.

3.5.1 IA Values

The topics in this section describe IA values that can be used with this application.

3.5.1.1 Web Services Input IA Values

In addition to the standard QuickModule IA Values, the *MDF* for the WS Input module also contains the following IA values:

Table 3-4: WS Input IA Values

IA Value	Description
<*InputImage*> (File,Input, Level 0)	The input image of the task on the Intelligent Capture Server. *`InputImage`* may be any type of file processed by upstream modules. If this value is used in the *IPP*, it is a trigger. When all trigger values are non-zero, the module will process the current task. This value is not used if WS Input is the first step in a process.
<*OutputImage*> (File, Output, 0)	The output image of the task on the Intelligent Capture Server. *`OutputImage`* may be any type of file. The user can map some In parameter to this IA Value – in this case it will contain downloaded image; otherwise it will contain the *`InputImage`* image.

IA Value	Description
<*CorrelationID*> (String, Input, NoTrigger)	This value can be used for asynchronous calls for storing the *`Correlation ID`*. Placement for the *`Correlation ID`* is defined during module setup when mapping the *`Correlation ID`* to an IA Value. This IA Value is only a placeholder for the *`Correlation ID`* which may be used if there is not a more convenient IA Value for this parameter. This placeholder will be empty if you map the *`Correlation ID`* to some other IA Value.

3.5.1.2 Web Services Output IA Values

In addition to the standard QuickModule IA Values, the *MDF* for the WS Output module also contains the following IA values:

Table 3-5: WS Output module IA Values

IA Value	Description
<*InputImage*> (File, Input, 0)	The input image of the task on the Intelligent Capture Server. *`InputImage`* may be any type of file processed by upstream modules or exported through a web call. If this value is used in the *IPP*, it is a trigger. When all trigger values are non-zero, the module will process the current task.
<*OutputImage*> (File, Output, 0)	The output image of the task on the Intelligent Capture Server. *`OutputImage`* may be any type of file. The user can map some In parameter to this IA Value – in this case it will contain downloaded image; otherwise it will contain the *`InputImage`* image.

IA Value	Description
<ServiceURL> (String, Input, NoTrigger)	This value contains the *URL* of the web service to be called and, by default, is completed with the value from the *WSDL*. However, this value can also be changed in the IPP to call the service from another URL.
<LevelN_StartDateTimeUTC>	Start date and time. • **Attributes**: String, Output • **Value**: 0 - 7
<LevelN_EndDateTimeUTC>	End date and time. • **Attributes**: String, Output • **Value**: 0 - 7

3.5.2 Sample System Files

This section describes sample files that are useful when configuring the Web Services subsystem. For descriptions of the scripting samples installed with the application, see *OpenText Intelligent Capture - Scripting Guide (ECPCORE-PSC)*.

3.5.2.1 Sample WSDL File

The Web Services Definition Language (*WSDL*) is an *XML*-based language that provides a model for describing web services. The WSDL defines services as ports, by associating a network address with a reusable binding, and a collection of ports define a service. When a client program connects to a web service, it can read the WSDL to determine the functions available on this server. It can call any number of these functions with a *SOAP* message.

The following sample WSDL files are included:

- *`IARescanWS.WSDL`*: This sample WSDL file is a necessary component of the WS Input Rescan scripting sample installed with the Web Services subsystem. The *Web Services scripting sample: WS Input Rescan* section contains more information about the scripting sample.
- *`ExportImagesWithZones.WSDL`*: This sample WSDL file is a necessary component of the WS Output scripting sample installed with the Web Services subsystem. The *Web Services scripting sample: WS Output* section contains more information about the scripting sample.

3.5.2.2 Sample SOAP Request with Correlation ID in SOAP Header

SOAP is a platform and language independent protocol that enables the exchange of *XML*-based messages over a network, most commonly using *HTTP*.

In a web call, the *Correlation ID* can reside in a parameter or in a SOAP header. In the SOAP header, the *Correlation ID* is distinguished by a name specified when configuring Correlation Mapping settings in Intelligent Capture Administrator:

```
<?xml version="1.0" encoding="utf-8"?>
<SOAP-ENV:Envelope>
  <SOAP-ENV:Header>
    <CorrelationIdName>Correlation ID Value</CorrelationIdName>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <Method></Method>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Related Topics

"Working with Correlation IDs" on page 50

"Web Services Enhancements (WSE) 3.0" on page 66

3.5.2.3 Sample MTOM-encoded File Upload

The Message Transmission Optimization Mechanism (*MTOM*) specification provides an efficient method of sending large amounts of data in a *SOAP* message to and from Web Services. When *WSE* MTOM encodes a SOAP message, most of the message is transmitted as *XML* in textual form. The rest of the message are large blocks of binary data that are optimized by transmitting them as-is, without conversion to text. SOAP messages with binary data that is not digitally signed, provide a performance improvement over protocols that require converting the binary data to XML.

Note: The .NET Framework limits MTOM attachments to 4*MB*. Attachments larger than 4MB can significantly affect the performance of the Web Services subsystem. Some large files (larger than 512MB) can cause the module or the Intelligent Capture Server to stop responding. You can increase this 4-MB limitation on MTOM attachments, but this practice is strongly discouraged.

The following code sample shows how a binary file, called `patch_t.tif`, can be MTOM-encoded, and sent as text data within a SOAP message:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <UploadDocument xmlns="http://captivasoftware.com/eim/webservices/">
      <FileName>patch_t.tif</FileName>
      <buffer>SUkqAAgAAAAPAP4ABAABAAAAAAAAAAAABBAABAAAApAYAAAEBBAA...
        Actual MTOM file data==</buffer>
      <Bytes>4841</Bytes>
    </UploadDocument>
  </soap:Body>
</soap:Envelope>
```

Related Topics

"Sending Files as MTOM Attachments" on page 52

"Web Services Enhancements (WSE) 3.0" on page 66

"Sample SOAP Request with Correlation ID in SOAP Header" on page 60

3.5.2.4 Sample MDF Files

Although Web Services modules do not always process images, their corresponding Module Definition Files (*MDF*) contain IA Value definitions for <*InputImage*> and <*OutputImage*>. If nothing was mapped to <*OutputImage*>, the <*OutputImage*> value contains the file parameter mapped to this value or the <*InputImage*> value.

If a module has an input value of type File and this module is the first in the process, then this IA Value cannot be populated. Only include an <*InputImage*> IA Value as the first module in a process if the WS Input module is the first step. When using the WS Input or WS Output modules to perform mapping, map the incoming file to the <*OutputImage*> IA Value only. The <*InputImage*> IA Value is intended to link Web Services modules in an *IPP* when the module copies the image from <*InputImage*> to <*OutputImage*> before performing any mappings. To send a file to a third party system, map <*InputImage*> to the web call parameter in WS Output.

Sample WS Input MDF: wsinput.mdf

The following code represents a sample Module Definition File for the WS Input module:

```
'
'   wsinput.mdf –
Web Service Input Module Definition File'
'   Copyright © 2018 Open Text. All Rights Reserved.

Module wsinput(0) ' - - - - - - - - - - - - - - - - - - - - - - - - - - - -

' ** Input and OutputFiles
       InputImage    as    File,Input
       OutputImage   as    File,Output
End Module

Module wsinput(T) ' - - - - - - - - - - - - - - - - - - - - - - - - - - - -
' ** Statistics for each coding task
       StartDateTime    as    Date,Output             'Date and time this task was
started
       EndDateTime      as    Date,Output             'Date and time of day this task
was completed
       TotalTime        as    Long,Output,Prime       'Number of milliseconds it took to
complete the task
       CorrelationID    as    String,Input,NoTrigger  'Correlation ID for asynchronous
call
       ErrorNumber      as    Long,Output
       ErrorText        as    String,Output           'Error number and text
End Module
```

Sample WS Output MDF: wsoutput.mdf

The following code represents a sample Module Definition File for the WS Output module:

```
'  wsoutput.mdf – Web Service Output Module Definition File
'   Copyright © 2018 Open Text. All Rights Reserved.

Module wsoutput(0) ' - - - - - - - - - - - - - - - - - - - - - - - - - - - -

' ** Input and OutputFiles
         InputImage    as    File,Input
         OutputImage   as    File,Output
         ServiceURL    as    String,Input,NoTrigger
End Module

Module wsoutput(T) ' - - - - - - - - - - - - - - - - - - - - - - - - - - - -
' ** Statistics for each coding task
     StartDateTime    as     Date,Output       'Date and time this task was started
     EndDateTime      as     Date,Output       'Date and time of day this task was
completed
     TotalTime        as     Long,Output,Prime 'Number of milliseconds it took to
complete the task
     ErrorNumber      as     Long,Output
     ErrorText        as     String,Output     'Error number and text
End Module
```

Sample WS Input MDF: WSInput_rescan.mdf

The following code represents a sample Module Definition File for the WS Input module:

```
Imports Microsoft.VisualBasic
'
'   wsinput.mdf - Web Service Input Module Definition File
'   Copyright © 2018 Open Text. All Rights Reserved.

    Module wsinput(0) ' - - - - - - - - - - - - - - - - - - - - - - - - - - - -

' ** Input and OutputFiles
         InputImage           as File,Input
         OutputImage          as File,Output

' ** Standard QuickModule values
         Level0_EndTime       as String,Output              'End time of the node
processing
         Level0_EndDate       as String,Output              'End date of the node
processing
         Level0_StartDate     as String,Output              'Start date of the node
processing
         Level0_StartTime     as String,Output              'Start time of the node
processing
         Level0_EndDateTime   as Date,Output                'End date and time of the
node processing
         Level0_StartDateTime as Date,Output                'Start date and time of the
node processing
         Level0_TotalTimeSpan as Long,Output                'Duration of the node
processing
         Level0_TotalTime     as Long,Output                'Duration of the node
processing
         Level0_ErrorName     as String,Output              'Error name
         Level0_ErrorNumber   as Long,Output                'Error number
         Level0_ErrorText     as String,Output              'Error text
         Level0_Processed     as Long,Input,Output,NoTrigger 'Whether the node has been
processed
         needs_rescan         as String,Output
         page_no              as String,Output
         reason               as String,Output
         thumb_marks          as String,Output
End Module

Module wsinput(1)
         Level1_EndTime       as String,Output              'End time of the node
```

```
processing
         Level1_EndDate       as String,Output               'End date of the node
processing
         Level1_StartDate     as String,Output               'Start date of the node
processing
         Level1_StartTime     as String,Output               'Start time of the node
processing
         Level1_EndDateTime   as Date,Output                 'End date and time of the
node processing
         Level1_StartDateTime as Date,Output                 'Start date and time of the
node processing
         Level1_TotalTimeSpan as Long,Output                 'Duration of the node
processing
         Level1_TotalTime     as Long,Output                 'Duration of the node
processing
         Level1_ErrorName     as String,Output               'Error name
         Level1_ErrorNumber   as Long,Output                 'Error number
         Level1_ErrorText     as String,Output               'Error text
         Level1_Processed     as Long,Input,Output,NoTrigger 'Whether the node has been
processed
End Module

Module wsinput(2)
         Level2_EndTime       as String,Output               'End time of the node
processing
         Level2_EndDate       as String,Output               'End date of the node
processing
         Level2_StartDate     as String,Output               'Start date of the node
processing
         Level2_StartTime     as String,Output               'Start time of the node
processing
         Level2_EndDateTime   as Date,Output                 'End date and time of the
node processing
         Level2_StartDateTime as Date,Output                 'Start date and time of the
node processing
         Level2_TotalTimeSpan as Long,Output                 'Duration of the node
processing
         Level2_TotalTime     as Long,Output                 'Duration of the node
processing
         Level2_ErrorName     as String,Output               'Error name
         Level2_ErrorNumber   as Long,Output                 'Error number
         Level2_ErrorText     as String,Output               'Error text
         Level2_Processed     as Long,Input,Output,NoTrigger 'Whether the node has been
processed
End Module

Module wsinput(3)
         Level3_EndTime       as String,Output               'End time of the node
processing
         Level3_EndDate       as String,Output               'End date of the node
processing
         Level3_StartDate     as String,Output               'Start date of the node
processing
         Level3_StartTime     as String,Output               'Start time of the node
processing
         Level3_EndDateTime   as Date,Output                 'End date and time of the
node processing
         Level3_StartDateTime as Date,Output                 'Start date and time of the
node processing
         Level3_TotalTimeSpan as Long,Output                 'Duration of the node
processing
         Level3_TotalTime     as Long,Output                 'Duration of the node
processing
         Level3_ErrorName     as String,Output               'Error name
         Level3_ErrorNumber   as Long,Output                 'Error number
         Level3_ErrorText     as String,Output               'Error text
         Level3_Processed     as Long,Input,Output,NoTrigger 'Whether the node has been
processed.
End Module

Module wsinput(4)
```

```
         Level4_EndTime       as String,Output               'End time of the node
processing
         Level4_EndDate       as String,Output               'End date of the node
processing
         Level4_StartDate     as String,Output               'Start date of the node
processing
         Level4_StartTime     as String,Output               'Start time of the node
processing
         Level4_EndDateTime   as Date,Output                 'End date and time of the
node processing
         Level4_StartDateTime as Date,Output                 'Start date and time of the
node processing
         Level4_TotalTimeSpan as Long,Output                 'Duration of the node
processing
         Level4_TotalTime     as Long,Output                 'Duration of the node
processing
         Level4_ErrorName     as String,Output               'Error name
         Level4_ErrorNumber   as Long,Output                 'Error number
         Level4_ErrorText     as String,Output               'Error text
         Level4_Processed     as Long,Input,Output,NoTrigger 'Whether the node has been
processed.
End Module

Module wsinput(5)
         Level5_EndTime       as String,Output               'End time of the node
processing
         Level5_EndDate       as String,Output               'End date of the node
processing
         Level5_StartDate     as String,Output               'Start date of the node
processing
         Level5_StartTime     as String,Output               'Start time of the node
processing
         Level5_EndDateTime   as Date,Output                 'End date and time of the
node processing
         Level5_StartDateTime as Date,Output                 'Start date and time of the
node processing
         Level5_TotalTimeSpan as Long,Output                 'Duration of the node
processing
         Level5_TotalTime     as Long,Output                 'Duration of the node
processing
         Level5_ErrorName     as String,Output               'Error name
         Level5_ErrorNumber   as Long,Output                 'Error number
         Level5_ErrorText     as String,Output               'Error text
         Level5_Processed     as Long,Input,Output,NoTrigger 'Whether the node has been
processed
End Module

Module wsinput(6)
         Level6_EndTime       as String,Output               'End time of the node
processing
         Level6_EndDate       as String,Output               'End date of the node
processing
         Level6_StartDate     as String,Output               'Start date of the node
processing
         Level6_StartTime     as String,Output               'Start time of the node
processing
         Level6_EndDateTime   as Date,Output                 'End date and time of the
node processing
         Level6_StartDateTime as Date,Output                 'Start date and time of the
node processing
         Level6_TotalTimeSpan as Long,Output                 'Duration of the node
processing
         Level6_TotalTime     as Long,Output                 'Duration of the node
processing
         Level6_ErrorName     as String,Output               'Error name
         Level6_ErrorNumber   as Long,Output                 'Error number
         Level6_ErrorText     as String,Output               'Error text
         Level6_Processed     as Long,Input,Output,NoTrigger 'Whether the node has been
processed.
End Module
```

```
Module wsinput(7)
         Level7_EndTime       as String,Output               'End time of the node
processing
         Level7_EndDate       as String,Output               'End date of the node
processing.
         Level7_StartDate     as String,Output               'Start date of the node
processing
         Level7_StartTime     as String,Output               'Start time of the node
processing
         Level7_EndDateTime   as Date,Output                 'End date and time of the
node processing
         Level7_StartDateTime as Date,Output                 'Start date and time of the
node processing
         Level7_TotalTimeSpan as Long,Output                 'Duration of the node
processing
         Level7_TotalTime     as Long,Output                 'Duration of the node
processing
         Level7_ErrorName     as String,Output               'Error name
         Level7_ErrorNumber   as Long,Output                 'Error number
         Level7_ErrorText     as String,Output               'Error text
         Level7_Processed     as Long,Input,Output,NoTrigger 'Whether the node has been
processed
         client_name          as String,Output
         client_version       as String,Output
         ia_batch_name        as String,Output
         priority             as String,Output
         client_info_XML      as String,Output
End Module

Module wsinput(T) ' - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      'This value can be used for asynchronous calls for storing Correlation ID.
      'Note that actual placement for Correlation ID is defined during module setup
      '(mapping Correlation ID to IA Value). This IA Value is only a possible
      'placeholder for Correlation ID, you may use it if there is no other IA Value
      'that is more convenient to use as a Correlation ID placeholder. If you map
       Correlation ID to some other IA Value then this one will be empty.

         CorrelationID      as String,Input,NoTrigger      'Standard QuickModule values
         EndTime            as String,Output               'Task end time.
         EndDate            as String,Output               'Task end date
         StartDate          as String,Output               'Task start date
         StartTime          as String,Output               'Task start time
         EndDateTime        as Date,Output                 'Task end date and time
         StartDateTime      as Date,Output                 'Task start date and time
         TotalTimeSpan      as Long,Output                 'Task duration
         TotalTime          as Long,Output                 'Task duration
         ErrorName          as String,Output               'Error name
         ErrorNumber        as Long,Output                 'Error number
         ErrorText          as String,Output               'Error text
         EndDateTimeUTC     as String,Output               'Task end date and time in
UTC format
         StartDateTimeUTC   as String,Output               'Task start date and time in
UTC format
         TaskResult         as Long,Output                 'Task result
         ExportResult       as Long,Output                 'Export result; repeats the
TaskResult
End Module
```

3.5.2.5 Web Services Enhancements (WSE) 3.0

Web Services Enhancements for Microsoft .NET (WSE) is a supported add-on to Microsoft Visual Studio .NET and the Microsoft .NET Framework providing developers the latest advanced web services capabilities to keep pace with the evolving web services protocol specifications. WSE 3.0 enables you to build secure web services on a network. WSE is necessary on the Web Services Server, the Web Services Client, as well as any machine where Web Services Hosting and WS Output modules are installed. This configuration is necessary for successful operation of Web Services in your architecture. For more information about the WSE 3.0 release, see What's New in Web Services Enhancements (WSE) 3.0 (http://msdn.microsoft.com/en-us/library/ms977317.aspx).

By default, the size of a request to WSE 3.0 is limited to 4*MB*. This limit is specified in an application configuration value called maxMessageLength *(http://msdn2.microsoft.com/en-us/library/aa528811.aspx)*. The following example specifies *HTTP* runtime parameters, including the `maxMessageLength`, for a WSE 3.0 application:

Note: The WSE 3.0 library does not process a web response if the `ContentType` of response is different from the `ContentType` of the request. If there are differences, you can resolve it by changing the `ContentType` of the response. This value is located in the HTTP header. Note that the `ContentType` of request cannot be changed.

```
<configuration>
  <microsoft.web.services3>
    <messaging>
      <maxMessageLength value="1024" />
    </messaging>
  </microsoft.web.services3>
</configuration>
```

Configure this setting in the `WSHosting.exe.config` file (in the same folder as `WSHosting.exe`), or in the `machine.config` file for global .NET settings. It is not recommended to increase this value without serious reason. Processing large files through *MTOM* increases the workload on both the Web Services Coordinator and WS Hosting services. As a result, the system is more sensitive to *DoS* or *DDoS* attacks. It is recommended that you use *URL*-fetching instead.

Note: If this application configuration value is increased, be sure that the value of the incoming web call timeout is set adequately in Intelligent Capture Administrator. A limit of 4*MB* is reasonable because it matches the default HTTP request size in Microsoft IIS.

Related Topics

"Web Services Coordinator" on page 28

"Web Services Hosting" on page 29

Chapter 4

ODBC Export Module

This section includes the Intelligent Capture general import/export module: **ecpcore-cxd**.

The ODBC Export module uses Open Database Connectivity (*ODBC*) to transfer data between the Intelligent Capture system and any ODBC data sources running on the client machine.

Note: You must be logged in with administrator rights when you install ODBC Export and also when you first log in after installing.

ODBC Export includes the following features:

Connects to various data sources

The ODBC Export module takes data from a batch, such as IA Values, files, and images, and exports the information to various data sources. For a list of the supported data sources, see the *Intelligent Capture Release Notes* (available in My Support (https://support.opentext.com)). ODBC Export can also import information from a data source into IA Values.

Transfers data to and from tables using actions

The ODBC Export module allows the user to transfer information between any number of tables within a data source. The transfer includes the following actions that retrieve and insert data into database tables:

- **Insert**: Adds new rows to a specified table, then populates the new rows with data from the IA Values or files.
- **Update**: Finds all of the existing rows that match certain criteria, then modifies the rows with data from IA Values or files.
- **Fetch**: Retrieves data from the first row that matches certain criteria. It then inserts the data from the row into an IA Value. Or, in the case of a file, it inserts the data into the local file system or the Intelligent Capture Server.
- **Run SQL**: Executes a specified *SQL* statement, enabling access to the entire SQL language. This action can call stored procedures that insert data into tables, or the user can create a custom SQL statement.

4.1 Setting Up ODBC Export

Most configuration information for Intelligent Capture client modules is defined during *module setup*. You can run a module in setup mode from Intelligent Capture Designer, from Intelligent Capture Administrator, or from a command prompt with appropriate arguments. For more information, see *OpenText Intelligent Capture - Module Reference (ECPCORE-CMD)*.

When launched in setup mode, a client module displays the **Setup** window in which you can specify the module's configuration settings.

4.1.1 Configuring Data Sources

Configure your data sources using the Microsoft *ODBC* Data Sources control panel before connecting to ODBC Export.

4.1.2 Understanding Data Source Connections

After launching ODBC Export for setup, use the **Mappings** tab to create a connection to at least one *ODBC* data source to create a mapping. You can connect to two types of data sources:

- **File data sources**: Available to all machines and users.
- **Machine data sources**: Specific to the machine to which you are connected.

Note: You cannot use ODBC Export to configure a data source. Configure your data sources using Microsoft ODBC Data Sources control panel before connecting to ODBC Export.

4.1.2.1 Creating a File Data Source Connection

When you click the **Connect** button on the **Mappings** tab of the **ODBC Export Setup** window, the **Select Data Source** window displays, allowing you to specify a file or machine data source.

To create a connection to a file data source:

1. Run ODBC Export for setup. The **ODBC Export - Setup** window displays.
2. Select the **Mappings** tab. The **Data Source** area displays the available data sources. and then
3. Click **Properties** to display the properties for a data source.
4. Click **Connect**. The **Select Data Source** window displays.
    a. Select the **File Data Source** tab to select the file data source that describes the driver that you want to connect to. You can use any file data source that refers to an *ODBC* driver which is installed on your machine.

This tab also allows you to connect with a data source that has file *DSN*. A file-based data source, not necessarily user-dedicated nor local to a computer, can be shared among all users who have the same drivers installed.

b. Click **New**. The **Create New Data Source** window is displayed.

c. Select a driver from the list and click **Next**. The **Create New Data Source** window prompts you to name the data source connection.

d. Type a name for the connection and click **Next**. The **Create New Data Source** window displays the new name and connection information for the data source.

> **Note:** If you select a name that is associated with an existing connection, the existing configuration information for the connection is overwritten by the new configuration. If you copy settings from one process or batch to another, the complete connection string is copied. It overwrites any DSN connection information that was specified on the target system.

e. Review the information and click **Finish**. A standard ODBC data source login window displays. The window displayed depends upon the data source selected.

f. Type login and other requested information and click **OK** to close the window and return to the **Mappings** tab.

5. View the connection information for the selected data source in the **Data Source** area.

6. Configure mappings for the data source. For more information, see "Understanding Mappings" on page 73.

> **Note:** After connecting to a data source, create a mapping before clicking **Cancel** on the **Mapping tab**. If you click **Cancel** before creating a mapping, the ODBC Export module does not save the connection to the data source.

Related Topics

"Understanding Data Source Connections" on page 68

"Creating a Machine Data Source Connection" on page 70

"Opening an Existing Data Source Connection" on page 71

"Viewing Data Source Properties" on page 71

4.1.2.2 Creating a Machine Data Source Connection

To create a connection to a machine data source:

1. Run ODBC Export for setup. The **ODBC Export - Setup** window displays.
2. Click **Connect** in the **Mappings** tab. The **Select Data Source** window displays.
3. Select the **Machine Data Source** tab to connect with a data source that has a user *DSN* or a system DSN. Machine data sources are specific to the local machine and cannot be shared. User data sources are specific to a user on the local machine. All users or a system-wide service can use the system data sources.
4. Click **New**. The **Create New Data Source** window is displayed.
5. Select one of the following options:
   - **User Data Source (Applies to this machine only)**: Creates a data source specific to the workstation in use and is visible only to the user currently logged in to the workstation.
   - **System Data Source (Applies to this machine only)**: Creates a data source specific to the workstation in use but that is visible to anyone who logs on to this machine.
6. Click **Next**. The **Create New Data Source** window displays the drivers to which you can connect.
7. Select a driver from the list and click **Next**. The **Create New Data Source** window displays the new connection information:
   - If you select a name that is associated with an existing connection, the existing configuration information for the connection is overwritten by the new configuration.
   - If you copy settings from one process or batch to another, the complete connection string is copied. It overwrites any DSN connection information that was specified on the target system.
8. Review the information and click **Finish**. A standard *ODBC* data source login window displays. The windows here varies depending on the data source.
9. Type login information and click **OK**. The **Mappings** tab displays, displaying connection information for the selected data source.
10. Configure mappings for the data source. For more information, see "Understanding Mappings" on page 73.

Related Topics

"Understanding Data Source Connections" on page 68

"Creating a File Data Source Connection" on page 68

"Opening an Existing Data Source Connection" on page 71

"Viewing Data Source Properties" on page 71

4.1.2.3 Opening an Existing Data Source Connection

To open a previously created connection to a data source:

1. Run ODBC Export for setup. The **ODBC Export - Setup** window displays.
2. Click **Connect** in the **Mappings** tab. The **Select Data Source** window displays.
3. Select the tab that corresponds to the type of data source to which you want to connect:
   - Select the **File Data Source** tab to access a list of data sources available to all machines and users.
   - Select the **Machine Data Source** tab to access a list of data sources specific to the machine to which you are connected.
4. Select the data source that you want to connect to and then click **OK**. A standard *ODBC* data source login window displays. The login window here varies depending on the data source.
5. Enter login information and click **OK**. The **Mappings** tab displays connection information for the selected data source.
6. Configure mappings for the data source. For information about setting up mappings, see "Understanding Mappings" on page 73.

Related Topics

"Understanding Data Source Connections" on page 68

"Creating a File Data Source Connection" on page 68

"Creating a Machine Data Source Connection" on page 70

"Viewing Data Source Properties" on page 71

4.1.2.4 Viewing Data Source Properties

The **Data Source Properties** window contains the connection information. The **Data Source Properties** window displays the following static connection information:

- **Data source**: Contains the name of the selected data source.
- **Server**: Contains the Server for this data source.
- **Driver**: Contains the driver used for this data source.

You can modify the information in the following fields:

- **DSN**: Contains the data source name, including the path. You can modify the connection string in the *DSN* field. For example, you can share DSN connection information between different machines. Use a driver, such as Microsoft Access *ODBC*, that uses fixed paths to reference the information.
- ***Password***: If a password has been assigned to the data source, type the password.

  Note: To increase security, ODBC Export parses password information from the DSN string and masks it with asterisks in the **Password** field. To avoid parsing issues, avoid the characters `[]{}(),;?*=!@`. Some ODBC drivers do not handle these characters. In this case, the drivers returns invalid information to the DSN string. By changing the properties such as the server name and drive, and then clicking **New Connection**, you can create a connection with any DSN string. This connection allows options that some ODBC connection windows do not provide, such as an Access DSN string free from fixed paths. When you modify the DSN information, you cannot save it to the existing connection; create one.
- *New Connection*: If you have modified properties on this window, click **New Connection** to establish the connection to the data source.

Note: If the connection between your machine database is lost during setup, exit from the ODBC Export module and then restart the module to establish the connection.

4.1.2.5 Modifying an Existing Data Source Connection

The **Data Source Properties** window contains the connection information.

To modify the properties for a data source:

1. Run ODBC Export for setup. The **ODBC Export - Setup** window displays.
2. Select the data source on the **Mappings** tab and then select **Properties**.
3. In the **DSN** field, modify the connection string that connects ODBC Export to a data source. Do this modification to share *DSN* connection information between different machines with a driver that uses fixed paths, such as Microsoft Access ODBC.
4. In the **Password** field, type a password if needed.
5. Click **New Connection** to establish a connection to the data source based on the assigned properties.
6. Click **OK** to accept the changes and close the **ODBC Export - Setup** window, or select a different tab to continue setting up the module.

Related Topics

"Understanding Data Source Connections" on page 68

"Creating a File Data Source Connection" on page 68

"Creating a Machine Data Source Connection" on page 70

"Opening an Existing Data Source Connection" on page 71

4.1.3 Understanding Mappings

ODBC Export stores configuration information in mappings. A mapping is a set of guidelines that specifies how the module transfers data. When creating a mapping, configure the following elements:

- **Level**: Indicate the level from which the module exports data.
- **Data source**: Specify the data sources to use to export or import data.
- **Action**: Specify a single action per mapping that the module executes to transfer data between ODBC Export and a data source table.
- **SQL**: If you cannot perform the data transfer procedure you want using the available actions, you can write a custom *SQL* statement within ODBC Export setup. For more information, see "Understanding SQL Statements" on page 77.
- ***Field mappings***: Create field mappings that indicate which individual columns and rows of a table ODBC Export maps to which IA Values or files. For more information, see "Understanding SQL Statements" on page 77.

A single ODBC Export step can include any number of mappings. For instructions, see "Setting Mapping Parameters" on page 73.

When ODBC Export receives a task during production, it runs the mappings one at a time, in the specified order. The module loops through every child node of the task node at the level specified in the mapping and executes the operation specified by the action.

4.1.3.1 Setting Mapping Parameters

This topic describes the **Mapping** area of the **Mappings** tab.

To configure a mapping:

1. Connect to one or more data sources using Understanding data source connections as a guide.
2. Run ODBC Export for setup. The **ODBC Export - Setup** window displays.
3. Select the **Mappings** tab. The options are:
   - **New**: Creates a mapping. Click to display the **Edit Mapping** window.
   - **Edit**: Modifies an existing mapping. Click to display the **Edit Mapping** window.
   - **Delete**: Deletes an existing mapping.

- **Restore**: Recovers a previously deleted mapping. With several mappings, they are restored to the **Mapping** list in the opposite order in which they were deleted. This means that the first deleted mapping is restored to the bottom of the list.

4. To create a mapping, click **New**. The **Edit Mapping** window displays.

   a. From the **Data source** list, select a data source.

   b. From the **Export from Level** list, select a level that is less than or equal to the trigger level for the step. The default is 0.

   c. In the **Action** area, select the appropriate function to perform. Depending on the data source to which you are connecting, some actions are not available.

      - **Insert**: Adds new rows to the table specified in your *SQL* statement. These rows are then populated with data from the IA Values or files specified in your field mapping.
      - **Update**: Overwrites the data in the rows of the table specified in your SQL statement. These rows are updated with data from the IA Values or files specified in your field mapping. When using the **Update** action, use a `WHERE` clause to specify which rows to update, otherwise the module updates the entire table.
      - **Fetch**: Retrieves data from the first row in your table that meets the criteria specified in the SQL statement. The data retrieved is inserted into the IA Value specified in the field mapping. With file data, it is written to the local file system or the Intelligent Capture Server. When using the **Fetch** action, note that ODBC Export:
        - Can only fetch data from an existing file directory. It cannot create a directory.
        - Must have permissions to write to the selected directory.
        - Will not prompt you before overwriting a file.
        - Cannot create IA Values or validate whether the IA Values typed in a file name are valid.
        - May return any single row that matches your specifications. Therefore, if you want to retrieve the same type of data from multiple rows, create multiple **Fetch** mappings. The module returns any row if you do not use an `ORDER BY` statement, because SQL returns rows in random order.
      - **Run SQL**: Executes the specified SQL statement. When using the **Run SQL** action, the module cannot retrieve return values from stored procedures.
      - **Error if more than one row found**: Returns an error if it detects multiple rows that match the specifications of a single row selected during setup. Available for **Update** and **Fetch** actions.

- **Error if no rows found**: Returns an error if it cannot find the row that was specified during setup. Available for **Update** and **Fetch** actions and recommended for **Fetch**.
- **Insert if no rows found**: Inserts data into a new row if it cannot find the row specified during setup. Available for **Update** when **Error if no rows found** is cleared.

d. In the **SQL Statement** field, type a SQL statement or click the **Select Table** option for assistance. For instructions on how to create SQL statements, see "Creating SQL Statements" on page 79 . The field names associated with the specified table in the **SQL Statement** field display in the grid, where you can map IA Values, default values, and formatting.

e. In the grid at the bottom of the window, select a field for which to create a field mapping. A field can be a row or column from the data source table.

Note: The **Field Name** and **Data Type** columns in the grid are read-only. Use these columns to view which fields are available to map and the corresponding data types for those fields.

f. In the **File** column, select one of the following options:

- **Yes**: Treats the value in the **Mapped to** column as a file.
- **No**: Treats the value in the **Mapped to** column as a string.

g. In the **Mapped to** column, enter the IA Value or file path you want to export or import. Or, click the **Browse** button and select an IA Value from the **Choose Value** window. ODBC Export inserts IA Values using the `@("ValueName")` format.

Note: When you use the **Fetch** action and set the **File** column to **No**, the value in the **Mapped to** field must be formatted as `@(Instance.Value)`. This format indicates the IA Value where to insert data in the database. When you select **Yes** in the **File** column, the ODBC Export module interprets the string as a file name where to write the data.

h. In the **Default** column, type the IA Value or file path to use when the value entered in the **Mapped to** field returns an empty string. As with the **Mapped to** field, you can type an IA Value directly into the field or click **Browse** to choose an IA Value from the **Choose Value** window.

i. In the **Format** column, select a format to apply to the data in the **Mapped to** and **Default** fields. You can also type a format in this column using Visual Basic formatting.

Note: Date fields usually require formatting. Most date fields in databases store both a date and a time. For example, if you map a time-only IA Value to a date field, the time can be incorrectly interpreted as a Julian date. Format date and time values using the **Format** field and use *VBA* functions in your *IPP* to ensure that dates and times are formatted and stored correctly.

The ODBC Export module exports and formats the date as <*yyyy-mm-dd*>. If you use VBA functions to format the date, verify that the date format is exporting correctly. It is possible that the results, when exporting a day value from 1 through 12, is <*yyyy-dd-mm*>. If the day value is between 13 to the end of the month, the results are <*yyyy-mm-dd*>.

j. In the **Empty is NULL** column, select **Yes** to export a `NULL` when the field is empty. Select **No** to export an empty value.

Note: To export a `NULL` value, the corresponding database field must support `NULL` values.

k. Repeat the previous steps for each field mapping you want to create. Use one of the following options to finish.

- **Refresh**: Executes a SQL statement and implements any modifications that you have made.
- **OK**: Saves the new mapping or changes to the existing mapping.

Note: To save changes to a mapping, click **OK** on both the **Edit Mapping** window and the **Mappings** tab. An asterisk (*) in the title bar of the active window indicates that unsaved changes exist for the mapping.

- **Cancel**: Discards the entire mapping if it is new, or changes to the existing mapping.

l. To save the mapping, click **OK**. The **Mappings** tab displays.

5. Change the sort order in the **Mapping** list using the arrow buttons. ODBC Export uses the sort order of the **Mapping list** to determine the order in which mappings are run during production. The list order determines the export order.

Related Topics

"Understanding Mappings" on page 73

"Understanding SQL Statements" on page 77

4.1.3.2 Understanding SQL Statements

ODBC Export executes *SQL* statements through the data source. You can create a SQL statement using two methods in the **Edit Mapping** window during ODBC Export setup:

- Select the **Run SQL** action to create your own statement.
- Select the **Insert**, **Update**, or **Fetch** action to have ODBC Export automatically create the corresponding SQL statement for you. The statement you create must specify a table.

You can type parameters into any SQL statement in any place where a value exists by inserting a `?` where the value is. For example:

```
select col1, col2, from example where col3=?
```

These parameters are mapped to IA Values, just like other fields. The field name for each is parameter is `?#n`, where <*n*> is the parameter position (i.e., the first parameter is called `?#1`, the second is called `?#2`, and so on).

The connected data source executes the SQL statement you create in ODBC Export. The data source evaluates the SQL statement and chooses which fields and parameters to set.

ODBC Export displays these fields and parameters in a grid. Use the options in this grid to create field mappings. A field mapping connects individual columns and rows in a database to IA Values or files; data is transferred between the specified sources.

The module remembers all of the settings for the field mappings you associate with a SQL statement. The module remembers settings it is the data source and not the module that executes the SQL statement. ODBC Export remembers the original settings for the fields even if you modify the SQL statement. You can modify the statement to no longer use some fields in the mapping and then use them again.

During setup, ODBC Export prompts the data source to evaluate the SQL statement at every key press. If the data source detects an error in the statement, ODBC Export displays the error. It does not allow you to save settings until the error is corrected.

Note: Writing SQL statements in ODBC Export is subject to some restrictions. For more information, see "Understanding SQL Statement Guidelines and Restrictions" on page 78.

Related Topics

"Creating SQL Statements" on page 79

"Understanding Mappings" on page 73

4.1.3.3 Understanding SQL Statement Guidelines and Restrictions

When configuring a mapping, create a *SQL* statement that, at a minimum, specifies the tables to which you want to export or import data. To enter a SQL statement, you can either type it in the **SQL Statement** field of the **Edit Mapping** window or use the **Select Table** window to help construct a SQL statement.

As you create your statement, please note the following:

- You must be connected to a data source. You must be connected to a data source to be able to create or modify a SQL statement. If you are not connected, you can only view SQL statements and their corresponding settings.
- You can use parameter markers in the place of values. Your SQL statement can contain parameter markers, such as a `?` in place of values. For example, the following `INSERT` statement uses `?` to display the IA Values used in the batch and insert the IA Values into columns 1-3:

  Insert into example `(col1, col2, col3) Values (?, ?, ?)`
- You cannot use SQL statements to fetch data. Do not use the **Run SQL** action to write SQL statements that fetch data from a database. To retrieve data, use the **Fetch** action instead.
- You can use the Run SQL action to call some stored procedures. Use the Run SQL action to write SQL statements. These statements insert values into a database using stored procedures, assuming that the data source supports it. The syntax for calling a stored procedure is:

  `{call procedure-name[([parameter][,[parameter]]...)]}`

  For example, `{call MyProc(?,?,?)}`
- You cannot include duplicate columns in your SQL statement. If you include two columns with the same name (case ignored) in your SQL statement, the program fails. Most databases do not allow you to create a table with duplicate names.
- Insert null values by omission. You cannot directly insert or update a null value using ODBC Export because the Intelligent Capture Server does not recognize null values. A null value is an unassigned value contrary to an empty value, which has no characters. To insert a null value into a database using the ODBC Export module, omit it in the `INSERT` statement. For example, with a table (`table1`) containing three columns (`col1`,`col2`,`col3`), create a null value for `col3` using the following SQL statement:

  Insert into table1 `(col1, col2) values (?, ?)`

  Note: When you run the mapping, the module ignores `col3` and fails to select it, forcing the value to remain unassigned.
- Carefully test your SQL statement. ODBC Export cannot determine which parameters the data source accepts. The module allows you to select parameters during setup even if they do not work during production. Test your SQL statement and corresponding settings carefully before using them in a production environment.

- If an error is detected, you cannot save changes. If the data source detects an error in your SQL statement, the module does not let you save changes until you have fixed the error. The module displays any SQL statement errors generated by the data source at the bottom of the **Edit Mappings**window. Not all databases can display error messages.
- If fields are left unmapped, wildcard mappings results in null values. `SELECT` statements that include a wildcard character (*) are stored as null values in database fields that are unmapped.

Related Topics

"Understanding SQL Statements" on page 77

"Creating SQL Statements" on page 79

4.1.3.4 Creating SQL Statements

In the **Select Table** window, you select a table and then ODBC Export populates the **SQL Statement** field with a `SELECT` statement. You can then customize the statement to meet your needs.

To create a *SQL* statement using the Select Table window:

1. Run ODBC Export for setup. The **ODBC Export - Setup** window displays.
2. To create a mapping, select the data source you want from the **Data Source** list, and then click **New**. To modify an existing mapping, select the mapping you want from the **Mapping** list, and then select **Edit**. In both cases, the **Edit Mapping** window displays.
3. From the **Edit Mapping** window, click the **Select Table** button. The **Select Table** window displays.
4. Type information in the **Schema**, **Table**, and **Catalog** fields to filter the tables that are displayed. For example, to display tables that belong to a specific schema, type that schema name in the **Schema** field.

   By default, ODBC Export displays all of the tables available from the **Data source** field in the **Edit Mapping** window. If no criteria are specified, the list contains all tables from the **Data source** list in the **Edit Mapping** window.
5. Click **Refresh** to view the filtered tables.
6. Select the tables to use for exporting or importing data. At the bottom of the window, the ODBC Export module displays the SQL statement to use for connecting to the table.
7. Click **OK**. The **Edit Mapping** window displays, showing the SQL statement and corresponding arguments selected.

Related Topics

"Understanding Mappings" on page 73

"Understanding SQL Statements" on page 77

"Understanding SQL Statement Guidelines and Restrictions" on page 78

4.1.4 Specifying When to Commit Transactions to the Data Source

Use options on the **Transactions** tab to specify when ODBC Export writes (or commits) data transactions to the data source. A transaction is a complete set of export operations written to a data source at one time.

Note: All **Fetch** actions are executed immediately whatever the settings on the **Transactions** tab.

To specify when ODBC Export commits transactions:

1. Run ODBC Export for setup. The **ODBC Export - Setup** window displays.
2. Select the **Transactions** tab.
3. Select one of the following options:
   - **After each Node**: Configures ODBC Export to commit transactions after it finishes processing each node. If the driver you are using supports transactions, you can select this option to minimize memory requirements.
   - **After each Mapping is finished (default)**: Configures ODBC Export to commit transactions after it executes each mapping.
   - **After all Mappings are finished**: Configures ODBC Export to commit transactions after it executes all mappings. Out of memory can occur with large batches.
4. Click **OK** to accept the changes and close the **ODBC Export - Setup** window, or select a different tab to continue setting up the module.

Related Topics

"Understanding Data Source Connections" on page 68

"Creating a File Data Source Connection" on page 68

"Creating a Machine Data Source Connection" on page 70

"Opening an Existing Data Source Connection" on page 71

"Viewing Data Source Properties" on page 71

"Specifying Connection Times and Conditions" on page 81

4.1.5 Specifying Connection Times and Conditions

Use the **Module** tab to specify how long and under what conditions you want ODBC Export to maintain a connection to a data source. Since connecting to a data source can potentially be time consuming, ODBC Export enables you to select when to disconnect.

Caution

The options described here are global. If you change connection duration settings on the **Module** tab, Intelligent Capture changes the settings of all batches that use ODBC Export.

To specify the duration of ODBC Export connections to data sources:

1. Run ODBC Export for setup. The **ODBC Export - Setup** window displays.
2. Select the **Module** tab.
3. Select one of the following options under **During run mode, keep Connections open:**
   - **As long as Module is running**: Never disconnect from the data source. Selecting this option ties up some connections, but can reduce processing time by reducing the time it takes to connect to data sources.
   - **If it was used in the last Task**: Wait to see if the next task uses the current connection before closing it. If you select this option, the module remains connected to the data source between tasks. It only closes the connection when it determines that the next task does not need it.
   - **Only during the current Task**: Disconnect from the data source immediately after a task is finished. If the ODBC Export module requires the same connection for consecutive tasks, the module still closes the connection after each task is finished. It then reopens the same connection.
4. Click **OK** to accept the changes and close the **ODBC Export - Setup** window, or select a different tab to continue setting up the module.

Related Topics

"Understanding Data Source Connections" on page 68

"Specifying When to Commit Transactions to the Data Source" on page 80

4.2 Running ODBC Export in Production

After the module has been set up, tested, and placed in a production environment, operators and administrators will run the module in production.

For information related to common production tasks for unattended modules, see *OpenText Intelligent Capture - Module Reference (ECPCORE-CMD)*

Note: The *Running Modules in Production Mode* section contains production topics that are common to many unattended modules and may not apply to this module.

4.2.1 Changing the View in Production Mode

In production mode, use the **View** menu and **View** tab of the control panel to modify how images display.

Table 4-1: Defining View Options During Production

Element	Description
Refresh	Updates the tree so that it reflects any changes to the tree structure.
Next Page	Displays the next image if more than one image is loaded.
Previous Page	Displays the previous image if more than one image is loaded.
Go to Page	Enables you to select which page to view if more than one image is loaded.
Fit Window	Scales an image to fit into the **Image** pane. If you change the window size, then the image size scales accordingly.
Fit Width	Scales an image to fit the width of the **Image** pane. If you change the window size, then the image width changes accordingly.
Scale 1 to 1	Displays one pixel of the image for each pixel in the Image pane. If you change the window size, then the image scale does not change.
Zoom In	Zooms in on the image by the designated zoom factor (default 10).
Zoom Out	Zooms away from the image by the designated zoom factor (default 10).
Change Zoom Factor	Changes the default zoom factor (10) to the number you specify.

Element	Description
Orientation	Changes the orientation of images. Generally only applicable when running the module in **Open Batch** mode because this is the only mode that enables an operator to manipulate pages before exporting. Select one of the following options from the **Orientation** menu: • **0 Degrees (Portrait) Default orientation** • **90 Degrees (Landscape)** • **180 Degrees** • **270 Degrees** Applies to current page or all pages in the tree pane depending on **Apply Settings To All Pages** check box. **Note:** Changing the view orientation does not affect the orientation of the output image unless the **Use Current Orientation** check box was selected during module step.
Scale to Gray	Displays images smaller than a 1:1 image size by representing the average density of missing pixels with a shade of gray, making images more readable. If you do not select this option, then the module hides some pixels when an image is reduced. For example, when an image is displayed at 25% of its actual size, only one pixel out of 16 is visible. The remaining pixels are disregarded and the resulting image may be unreadable.
Invert	Inverts the image in the **Image** pane, replacing white pixels with black and black pixels with white.
Brightness and Contrast sliders	For controlling the display of the image.
Apply Settings to All Pages	Applies viewing options to all loaded images.
Control Panel	Displays or hides the control panel.
Status Bar	Displays or hides the status bar at the bottom of the window.

Element	Description
Tree Preferences	Modifies the tree appearance: • Background color, lines, all level name text or a single selected text, and all thumbnail image text or a single selected thumbnail image text. • Font of the tree text. • Pairs the fronts with the backs of images if you scanned two sided pages.
Save Preferences Now	Saves the changes you made to the tree preferences now.
Save Preferences On Exit	Saves the changes you made to the tree preferences.

4.3 Reference—ODBC Export

The topics within this section contain reference information useful while using the application in setup or production.

4.3.1 Windows

The topics within this section provide descriptions of windows accessible from the application. The topics list the user interface element name and include a brief description of actions available from the window.

4.3.1.1 Data Source Properties

To display the properties for a data source, select the data source you want on the Mappings tab and then select **Properties**. The **Data Source Properties** window displays showing connection information, such as the data source name, server, and driver.

Table 4-2: Data Source Properties

Element	Description
Data Source	The name of the selected data source.
Server	The server for this data source.
Driver	The driver used for this data source.
DSN	Data source name, including the path. You can modify the connection string in the *DSN* field, when you need to share DSN connection information between different machines and are using a driver, such as a Microsoft Access *ODBC* driver, which uses fixed paths to reference the information.

Element	Description
Password	If a password has been assigned to the data source, type the password.
New Connection	If you have modified properties on this window, click **New Connection** to establish the connection to the data source based on your changes.

Note: To increase security, ODBC Export parses password information out of the DSN string and masks it with asterisks in the Password field. The module may have trouble parsing the password if it contains the characters `[]{}(),;?*=!@` since certain ODBC drivers cannot handle these characters and will return invalid information to the DSN string. By changing the properties such as the server name and drive, and then clicking Click **New Connection**, you can create a connection with any DSN string. This allows options that some ODBC connection windows do not provide, such as an Access DSN string free from fixed paths. When you make modifications to DSN information, you cannot save it to the existing connection; you must create one.

If the connection between your machine database is lost during setup, exit from ODBC Export and then restart the module to reestablish the connection.

4.3.1.2 Edit Mapping

Use the **Edit Mappings** window to configure new or existing mappings. To display the **Edit Mappings** window: From the **Mappings** tab, either select a data source and then click **New**, or select an existing mapping and then click **Edit**.The **Edit Mappings** window settings include:

Table 4-3: Edit Mappings Window

Element	Description
Data source	Displays the **Data source** list.
Export from Level:	Select the level from which you want to export data. The default is 0. **Note:** You must select a level that is less than or equal to the trigger level for the step.
SQL Statement:	Manually type in an **SQL statement** or select the **Select Table** option for assistance. The "Creating SQL Statements" on page 79 section contains instructions. The field names associated with the specified table in the **SQL statement** field display in the grid, where you can map IA Values, default values, and formatting.

Element	Description
Select Table...	Displays the **Select Table** window where you can construct *SQL* statements. The "Creating SQL Statements" on page 79 section contains instructions.

Element	Description
Action	• **Insert** - Adds new rows to the table specified in your SQL statement. These rows are then populated with data from the IA Values or files specified in your field mapping. • **Update** - Overwrites the data in the rows of the table specified in your SQL statement. These rows are updated with data from the IA Values or files specified in your field mapping. When using the **Update** action, you should use a `WHERE` clause to specify which rows you want to update, otherwise the module may update the entire table. • **Fetch** - Retrieves data from the first row in your table that meets the criteria specified in the SQL statement. The data retrieved is inserted into the *IA* Value specified in the field mapping, or in the case of file data, written to the local file system or the Intelligent Capture Server. When using the **Fetch** action, note that ODBC Export: • Can only fetch data from a file directory that already exists; it cannot create a directory. • Must have permissions to write to the selected directory. • Will not prompt you before overwriting a file. • Cannot create IA Values or validate whether the IA Values typed in a file name are valid. • May return any single row that matches your specifications. Therefore, if you want to retrieve the same type of data from multiple rows, create multiple Fetch mappings. The module may return any row if you do not use an `ORDER BY` statement, because SQL returns rows in random order. **Run SQL** - Executes the specified SQL statement. When using the **Run SQL** action, the module cannot retrieve return values from stored procedures. **Error if more than one row found** (Available for **Update** and **Fetch** only) - Returns an error if it detects multiple

Element	Description
	rows that match the specifications of a single row selected during setup. **Error if no rows found** (Available for Update and Fetch only) - Returns an error if it cannot find the row that was specified during setup (recommended for **Fetch**). **Error if no rows found** (Available for Update and Fetch only) - Inserts data into a new row if it cannot find the row specified during setup (only enabled for **Update**). Only available when **Error if no rows found** is cleared.

Element	Description
Field grid	• **Field Name** - Read-only. View which fields are available to map. • **Data Type** - Read-only. View the corresponding data types for the available fields. • **File?** - Select **Yes** or **No**. • **Yes** - Treats the value in the **Mapped to** column as a file. When the value is formatted as `@(InstanceName.FileName)`, ODBC Export writes to the node file of that name. • **No** - Treats the value in the **Mapped to** column as a string. ODBC Export relies on the *ODBC* driver to convert that value to an appropriate data type. **Mapped to** Enter the IA Value or file path you want to export or import. Or, select the **Browse** button and select an IA Value from the Choose Value window. ODBC Export inserts IA Values using the `@("ValueName")` format. **Note:** When you use the **Fetch** action and set the **File?** column to **No**, the value in the **Mapped to** field must be formatted as `@(Instance.Value)`. This indicates the *IA* Value where data will be inserted in the database. When you select **Yes** in the **File?** column, the ODBC Export module interprets the string as a file name where it should write the data. • **Default** - Enter the *IA* Value or file path that you want the module to use when the value entered in the **Mapped to** field returns an empty string. As with the **Mapped to** field, you can type an IA Value directly into the field or click **Browse** to choose an IA Value from a list. • **Format** - Select a format to apply to the data in the **Mapped to** and **Default** fields. You can also manually type a format in this column using standard Visual Basic formatting. Date fields usually require formatting. Most date fields in databases store both a date and a time. For example, if you map a time-only *IA* Value to a date field, the time may be incorrectly interpreted as a

Element	Description
	Julian date. You may need to format date and time values using the **Format** field and also use *VBA* functions in your *IPP* to ensure that dates and times are formatted and stored correctly. The ODBC Export module exports and formats the date as <*yyyy-mm-dd*>. If you use VBA functions to format the date, you should verify that the date format is exporting correctly. It is possible that your results, when exporting a day value between 1 and 12, will be <*yyyy-dd-mm*>. If the day value is between 13 to the end of the month, your results will be <*yyyy-mm-dd*>. • **Empty is NULL** - Click **Yes** if you want to export a `NULL` when the field is empty. Click **No** if you want to export an empty value. If you want to export a `NULL` value, the corresponding database field must support `NULL` values.
Refresh	Executes your SQL statement and implements any modifications that you have made.
OK	Saves the new mapping or changes to the existing mapping.
Cancel	Discards the entire mapping if it is new, or changes to the existing mapping.

Note: To save changes to your mapping, you must click **OK** on both the **Edit Mappings** window and the **Mappings** tab. An asterisk (*) in the title bar of the active window indicates that unsaved changes exist for the mapping.

Related Topics

"Understanding Mappings" on page 73

"Setting Mapping Parameters" on page 73

"Understanding SQL Statements" on page 77

4.3.1.3 ODBC Export

The available options and components of the **ODBC Export** window are:

Table 4-4: ODBC Export Production Window

Element	Description
File menu	The **File** menu provides options for running open batches, setup, printing and exiting.
View menu	The **View** menu provides options for manipulating the image.
Run menu	The **Run** menu provides options for running batches.
Help menu	The **Help** menu is for connecting to the online help.
Control panel	The **Control Panel** contains the **View** tab and elements found in the **View** and **Run** menus that guide the production process. You can move the control panel by clicking on the upper edge and dragging it. To re-dock the control panel, press **ESC**.
Tree pane	Displays a graphical representation of the hierarchical groupings of pages, documents and folders, using a thumbnail image for each batch page and a folder icon for each higher level node in the tree.
Image pane	Displays the currently selected **Tree pane** page.

4.3.1.3.1 File Menu

In production mode, the **File** menu contains the following options:

Table 4-5: File menu

Element	Description
Open Batch	Displays the **Open Batch** window.
Close Batch	Closes the current batch.
Set Up Instance	Allows batch level setup while running the module in production mode. The new processing instructions only affect the current batch.
Export	Exports the batch.
Print	Prints the currently selected document.

Element	Description
Exit	Exits the program.

4.3.1.3.2 Run Menu

In production mode, the **Run** menu contains the following options:

Table 4-6: Run Menu

Element	Description
All Batches	Processes batches in Run All Batches mode. The module receives any queued tasks from any open batch as the tasks become available from the Intelligent Capture Server.
Single Batch	Processes a single queued batch selected from the **Open Batch** window.
Stop	Stops batch processing.

4.3.1.3.3 View Menu

This table describes options available from **ODBC Export** window **View** menu and the **View** tab of the control panel.

Table 4-7: View Menu

Menu option	Keyboard shortcut	Description
Refresh	**F5**	Updates the tree so that it reflects any changes to the tree structure.
Next Page	**CTRL+N**	Displays the next image if more than one image is loaded.
Previous Page	**CTRL+P**	Displays the previous image if more than one image is loaded.
Go to Page	**CTRL+G**	Enables you to select which page to view if more than one image is loaded.
Fit to Window		Scales an image to fit into the **Image** pane. If you change the window size, then the image size scales accordingly.

Menu option	Keyboard shortcut	Description
Fit to Width		Scales an image to fit the width of the Image pane. If you change the window size, then the image width changes accordingly.
Scale 1 to 1		Displays one pixel of the image for each pixel in the Image pane. If you change the window size, then the image scale does not change.
Zoom In	**CTRL+I**	Zooms in on the image by the designated zoom factor (default 10).
Zoom Out	**CTRL+O**	Zooms away from the image by the designated zoom factor (default 10).
Change Zoom Factor		Changes the default zoom factor (10) to the number you specify.

Menu option	Keyboard shortcut	Description
Orientation		Changes the orientation of images. Generally only applicable when running the module in Open Batch mode because this is the only mode that enables an operator to manipulate pages before exporting. Select one of the following options from the **Orientation** menu: • 0 Degrees (Portrait) Default orientation. • 90 Degrees (Landscape) • 180 Degrees • 270 Degrees Applies to current page or all pages in the tree pane depending on **Apply Settings To All Pages** check box. **Note:** Changing the view orientation does not affect the orientation of the output image unless the **Use Current Orientation** check box was selected during module step or Open Batch setup. The *Specifying file content settings* section contains information on this control.

Menu option	Keyboard shortcut	Description
Scale to Gray	**CTRL+L**	Displays images smaller than a 1:1 image size by representing the average density of missing pixels with a shade of gray, making images more readable. If you do not select this option, then the module hides some pixels when an image is reduced. For example, when an image is displayed at 25% of its actual size, only one pixel out of 16 is visible. The remaining pixels are disregarded and the resulting image may be unreadable.
Invert		Inverts the image in the **Image** pane, replacing white pixels with black and black pixels with white.
Apply Settings to All Pages		Applies viewing options to all loaded images.
Control Panel		Displays or hides the control panel on the left side of the window. .
Status Bar		Displays or hides the status bar at the bottom of the window.
Tree Preferences		Makes several changes to the tree appearance, including: • Changes the color of the background, lines, all level name text or a single selected text, and all thumbnail image text or a single selected thumbnail image text. • Changes the font of the tree text. • Pairs the fronts with the backs of images if you scanned two sided pages.
Save Preferences Now		Immediately saves the changes you made to the tree preferences.
Save Preferences on Exit		Saves the changes you made to the tree preferences.

4.3.1.3.4 Help Menu

In production mode, the **Help** menu contains the following options:

Table 4-8: Help Menu

Element	Description
ODBC Export Help	Opens the module online help.
Master Online Help Index	Opens the Intelligent Capture online help.
About ODBC Export	Shows the **About** window containing information about the installed version of the module.

4.3.1.3.5 Control Panel

The **Control Panel** contains the **View** tab and elements found in the **View** and **Run** menus that guide the production process. You can undock the Control Panel by clicking and dragging the upper edge of the panel. To re-dock the Control Panel, press **ESC** or move it to the edge of the production window.

Table 4-9: Control Panel

Element	Description
Run All Batches	Run All Batches mode is a Wait for Task mode (a mode in which you can only process Batches that are queued on the Intelligent Capture Server for the module). In Run All Batches mode, the module receives any queued tasks from any open batch as the tasks become available from the Intelligent Capture Server.
Run Single Batch	Run Single Batch mode is a Wait for Task mode. In Run Single Batch mode, you choose a single queued batch to process.
Open Batch	In Open Batch mode, you can process any batch at any time, regardless of whether the batch is queued on the Intelligent Capture Server for the module or includes the module in its process. When you run in Open Batch mode, the Intelligent Capture Server ignores the standard `Prepare` and `Finish` event handlers in your *IPP*.
Stop	Appears during production. Stops batch processing.
Previous	Displays the previous image if more than one image is loaded.

Element	Description
Next	Displays the next image if more than one image is loaded.
View tab	The **View** tab provides many of the same operations available on the **View** menu.

4.3.1.3.6 View Tab

This table explains the options available from the **ODBC Export** window **View** menu and the **View** tab of the production mode control panel.

Table 4-10: View Tab

Menu option	Keyboard shortcut	Description
Previous	**CTRL+P**	Displays the previous image if more than one image is loaded.
Next	**CTRL+N**	Displays the next image if more than one image is loaded.
Go to Page	**CTRL+G**	Enables you to select which image to view if more than one image is loaded.
Fit to Window		Scales an image to fit into the window. If you change the window size, the image size scales accordingly.
Fit to Width		Scales an image to fit the width of the window. If you change the window size, the image width changes accordingly.
1 to 1		Displays one pixel of the image for each pixel in the window. If you change the window size, the image scale does not change.
Zoom In	**CTRL+I**	Zooms in on the image by the designated zoom factor (default 10). To change the zoom factor, select the **Change Zoom Factor** option.

Menu option	Keyboard shortcut	Description
Zoom Out	**CTRL+O**	Zooms away from the image by the designated zoom factor (default 10). To change the zoom factor, select the**Change Zoom Factor** option.
Change Zoom Factor		Changes the default zoom factor (10) to the number you specify. The zoom factor is the incremental value that an image's size changes when you select Zoom In or Zoom Out from the **View** menu. **Note:** Changing the zoom factor does not automatically change the image size.
Orientation		Displays the image in the image window at one of the following orientations: • 0 degrees (portrait position) • 90 degrees (landscape position) • 180 degrees • 270 degrees
Scale to Gray	**CTRL+L**	Displays images smaller than a 1:1 image size by representing the average density of missing pixels with a shade of gray, making images more readable. If you do not select this option, the module hides some pixels when an image is reduced. For example, when an image is displayed at 25% of its actual size, only one pixel out of four is visible. The remaining pixels are disregarded and the resulting image may be unreadable.
Invert	**CTRL+V**	Inverts the image, replacing white pixels with black and black pixels with white.
Apply Settings to All Pages		Applies viewing options to all loaded images.

4.3.1.4 ODBC Export Setup

ODBC Export Setup window includes the following tabs:

Table 4-11: ODBC Export Setup Window

Element	Description
"Mappings Tab" on page 99	Gives you control for creating, viewing, modifying or deleting mappings.
"Transactions Tab" on page 100	Configures ODBC Export commit transactions.
"Errors Tab" on page 101	Specifies how to handle production errors.
"Module Tab" on page 102	Controls module behavior during production.

4.3.1.4.1 Mappings Tab

The **Mappings** tab settings include:

Table 4-12: Mappings Tab

Element	Description
Data Source	Displays the available data sources you can map to.
Connect...	Click to create a data source connection or open an existing one; the **Select Data Source** window displays.
Properties...	To display the properties for a data source, select the data source you want on the **Mappings** tab and then click **Properties**. The **Data Source Properties** window displays, giving connection information such as the data source name, server, and driver.
Mapping	Displays the **Mapping** list. Mappings are executed in the order in which they appear in the **Mapping** list. Use the up and down arrow buttons to change the export order of the mappings.
New	To create a mapping, select the data source you want from the **Data Source** list, and then click **New** to display the **Edit Mapping** window.
Edit	To modify an existing mapping, select the mapping you want from the **Mapping** list, and then click **Edit** to display the **Edit Mapping** window.

Element	Description
Delete	To delete an existing mapping, select the mapping you want from the **Mapping** list, and then click **Delete**.
Restore	To recover a previously deleted mapping, click **Restore**. ODBC Export adds the previously deleted mapping to the bottom of the **Mapping** list. **Note:** When you restore more than one mapping, they are added back to the **Mapping** list in the opposite order in which they were deleted (i.e., the first one deleted is restored to the very bottom of the list).

Note: To save changes to your mapping, you must click **OK** on both the **Edit Mapping** window and the **Mappings** tab. An asterisk (*) in the title bar of the active window indicates that unsaved changes exist for the mapping.

Related Topics

"Understanding Mappings" on page 73

"Understanding SQL Statements" on page 77

4.3.1.4.2 Transactions Tab

The **Transactions** tab settings include:

Table 4-13: Transactions Tab

Element	Description
After each node	Configures ODBC Export to commit transactions after it finishes processing each node. If the driver you are using supports transactions, you may want to select this option to minimize memory requirements.
After each mapping is finished (default)	Configures ODBC Export to commit transactions after it executes each mapping.
After all mappings are finished	Configures ODBC Export to commit transactions after it executes all mappings. If you are processing large batches, you may run out of memory if you select this option.

Note: All Fetch actions are executed immediately and are not affected by settings on the **Transactions** tab.

Related Topics

"Understanding Data Source Connections" on page 68

"Specifying When to Commit Transactions to the Data Source" on page 80

"Specifying Connection Times and Conditions" on page 81

"Understanding Mappings" on page 73

4.3.1.4.3 Errors Tab

The **Errors** tab settings include:

Table 4-14: Errors Tab

Element	Description
If a Mapping has already been successfully run on a node:	The skip and redo options correspond to operations performed for mappings if a node was retriggered. • **Skip: Run/Do not run a mapping for a Node**: Does not run successfully executed mappings for a node. Will skip the mappings that were successfully executed and only re-execute the failed mappings for a node. • **Redo: run Node again**: Re-exports the nodes, replacing the previously exported information with new information. This means it will re-execute all the mappings, including the ones that were successfully executed.
When an error occurs, automatically retry	• **N times, if applicable** - Specify the number of times you want the ODBC Export module to automatically retry an operation that fails before returning an error code to the Intelligent Capture Server. **Note:** This option specifies how many times the module, not the Intelligent Capture Server, retries the task. If you enter 3 here, set the `RetriesLeft` IA Value in your *IPP* to 3 as well. ODBC Export may potentially try to process the same task up to nine times before abandoning all retry attempts.

Element	Description
If retrying fails or does not apply:	• **Abort current Task and keep accepting Tasks** - Abort the failed task and accept a new task. When the task is aborted, the `ExportResult` IA Value is populated with a value indicating which error occurred. You can configure your IPP to write this error value to the Windows Event Log. • **Abort current Task, then stop accepting Tasks** - Abort the failed task and do not accept a new task. When the task is aborted, the `ExportResult` *`IA`* Value is populated with a value indicating which error occurred. You can configure your IPP to take the appropriate action based on the error value. • **Continue current Task and keep accepting Tasks** - Log an error and allow the operator to continue processing the failed task and additional tasks thereafter. • **Continue current Task, then stop accepting Tasks** - Log an error and allow the operator to continue processing the failed task. The module stops accepting tasks after it finishes the task which encountered an error. • **Prompt user with these options** - Allow the operator to select how the error is handled, prompting the operator with the options: Abort, Retry, and Continue. This is the only error handling option that halts processing.
Log errors to the InputAccel Server	Select this check to log errors to the **Windows Event** log on the computer running the Intelligent Capture Server.

4.3.1.4.4 Module Tab

The **Module** tab settings include:

Table 4-15: Module Tab

Element	Description
During Run Mode, Keep Connections Open:	• **As long as Module is running** Never disconnect from the data source. Selecting this option ties up some connections, but can reduce processing time by reducing the time it takes to connect to data sources. • **If it was used in the last task** Wait to see if the next task uses the current connection before closing it. If you select this option, the module will remain connected to the data source between tasks, and only close the connection when it determines that the next task does not need it. • **Only during the current task** Disconnect from the data source immediately after a task is finished. If you select this option and ODBC Export requires the same connection for consecutive tasks, the module will still close the connection after each task is finished, and then re-open the same connection.

Caution

The options described here are global. If you change connection duration settings on the **Module** tab, Intelligent Capture will change the settings of all batches that use ODBC Export.

Related Topics

"Specifying Connection Times and Conditions" on page 81

"Understanding Data Source Connections" on page 68

"Specifying When to Commit Transactions to the Data Source" on page 80

4.3.1.5 Select Data Source

When you click the **Connect** button on the **Mappings** tab of the **ODBC Export Setup** window, the **Select Data Source** window displays, allowing you to specify a file or machine data source.

Table 4-16: Select Data Source Window

Element	Description
File Data Source tab	The **File Data Source** tab enables you to select the file data source that describes the driver that you want to connect to. You can use any file data source that refers to an *ODBC* drivers which is installed on your machine. This tab also allows you to connect with a data source that has file data source names (*DSN*). A file-based data source, not necessarily user-dedicated nor local to a computer, can be shared among all users who have the same drivers installed.
Machine Data Source tab	The **Machine Data Source** tab enables you to connect with a data source that has a user data source name (DSN) or a system DSN. Machine data sources are specific to this machine and cannot be shared. A machine data source is specific to this machine and cannot be shared. User data sources are specific to a user on this machine. System data sources can be used by all users on this machine, or by a system-wide service.

4.3.1.5.1 File Data Source Tab

Lets you connect with a data source that has file data source names (*DSN*). A file-based data source, not necessarily user-dedicated nor local to a computer, can be shared among all users who have the same drivers installed.

Table 4-17: File Data Source Tab

Element	Description
File Data Source	Displays all file DSNs and subdirectories of the directory indicated in the **Look in** box. Double-clicking a DSN connects to the data source.

Element	Description
Look in	Indicates the directory for which the subdirectories and file DSNs are listed in the window below. Clicking the down arrow alongside this text box displays the entire directory structure.
Up One Level	Replaces the directory indicated in the **Look in** box with the directory located one level up.
DSN Name	Displays the file DSN name, which is either selected in the **File Data Sources** list or entered manually.
New	Click to add a new file data source. In the **Create New Data Source** window that is displayed, choose the driver for which you are adding a file DSN and click **Next** to specify the name or location of the new file DSN. Click **Next** again to view a summary of the new information. Click **Finish** to display the driver-specific setup window.
OK	Closes the **Select Data Source** window and connects to the file data source that is indicated in the **DSN Name** text box. You do not have to click **OK** to accept changes to the **File Data Sources** list. Changes to the list are accepted when you click **OK** in the **Data Source Setup** window.
Cancel	Closes the **Select Data Source** window without connecting to the file data source and without undoing changes made via other window controls.

4.3.1.5.2 Machine Data Source Tab

enables you to connect with a data source that has a user data source name (*DSN*) or a system DSN. Machine data sources are specific to this machine and cannot be shared.

Table 4-18: Machine Data Source Tab

Element	Description
Machine Data Source	Lists all user and system DSNs, including the name and type of each DSN. Double-clicking a DSN connects to the data source.

Element	Description
New	Click to add a new machine data source. In the **Create New Data Source** window that displays, choose the driver for which you are adding a user or system DSN and click **Next** to specify the name or location of the new DSN. Click **Next** again to view a summary of the new information. Click **Finish** to display the driver-specific setup window.
OK	Closes the **Select Data Source** window and connects to the selected data source. Clicking **OK** does not accept changes to the **Machine Data Sources** list. Changes to the list are accepted when you click **OK** in the**Data Source Setup** window.
Cancel	Closes the **Select Data Source** window without connecting to the file data source and without undoing changes made via other window controls.

4.3.1.6 Select Table

The **Select Table** window enables you to construct *SQL* statements. This window helps you select which table you want to use, and then creates a `SELECT` statement for that table. The ODBC Export module populates the **SQL Statement** field with the `SELECT` statement, which you can use as is or further customize to meet your needs.

The **Select Table** window displays when you click the **Select Table** button in the **Edit Mapping** window.

Table 4-19: Select Table Window

Element	Description
Schema	Type a schema name to limit the table list.
Table	Type a table name to limit the table list.
Catalog	Type a catalog name to limit the table list.
Table list	Lists all the tables that correspond to the specified **Schema**, **Table**, and **Catalog** criteria. If no criteria are specified, the list contains all tables from the Data source listed in the **Edit Mapping** window.
SQL	Displays the SQL statement associated with the selected table.
Refresh	Click the **Refresh** button to change the display of the listed tables after you specify new **Schema**, **Table**, and **Catalog** criteria.

Related Topics

"Understanding Mappings" on page 73

"Understanding SQL Statements" on page 77

4.3.2 IA Values

The topics in this section describe IA values that can be used with this application and list the IA Values defined in the `iaxodbc2.mdf`.

4.3.2.1 Input IA Values

Input IA Values include input file variables.

Table 4-20: Input IA Values

IA Value	Description
<InputImage>	File name of the input image on the Intelligent Capture Server. The file that corresponds with *<InputImage>* is a binary *TIFF* file. This is the only trigger variable for the module, which means that after it is set to a non-zero value, the Intelligent Capture Server sends tasks to a running ODBC Export machine. • **Attributes**: File, Input, Trigger

4.3.2.2 Output IA Values

Output IA Values include file and processing variables.

Table 4-21: Output IA Values

IA Value	Description
<Level<n>_Action>	An integer specifying the output result. Valid values are: • 0 = Node was not processed • 1 = File exists and was skipped without prompting • 2 = File exists and was overwritten without prompting • 3 = File exists and error occurred • 4 = New file created • 4097 = File exists, operator was prompted and selected skip • 4098 = File exists, operator was prompted and selected overwrite • 4099 = File exists, operator was prompted and selected to log an error Attributes are: • **Attributes**: Long, Output • **Level<*n*>**: *<n>* = 0-7 • **Example**:*<Level3_Action>* as Long, Output
<Level<n>_FullPath>	The fully-resolved path name after evaluating all IA Values and converting them to strings. • **Attributes**: String, Output • **Level<*n*>**: *<n>* = 0-7 • **Example**: *<Level3_FullPath>* as String, Output

Chapter 5

Standard Export Module

This section includes the Intelligent Capture general import/export module: **ecpcore-cxb**.

This guide is designed to present users with conceptual information about this application, as well as step-by-step instructions on how to use the application. The topics within this section cover basic overview information and introduce the features and functions of the application.

The specific capabilities of each client module are described in *OpenText Intelligent Capture - Module Reference (ECPCORE-CMD)*.

5.1 Understanding the Standard Export Module

The Standard Export module performs data export from a batch to the specified destination. The data can be exported in any of the supported file formats.

Specifically, Standard Export can perform the following tasks and functions:

- **Export to a variety of file formats**: Standard Export can transform batch data to CSV format, free text format, XML format, data file format, and email format (HTML/Text), CMIS, and ApplicationXtender.
- **XSL transformation**: XML formatted export combined with a customer-defined XSL template can transform batch data to other export formats.
- **Export to a variety of destinations**: Standard Export supports batch data export to the following destinations:
  - **File system**: any generated file can be stored to the specified file system folder (local or network-located).
  - **Email**: server: an email message can be sent to the specified SMTP mail server.
  - **Batch**: formatted files, such as CSV, XML, and text files, can be stored to an IA value of type File in the batch.
  - CMIS-compliant server
  - ApplicationXtender server
  - OpenText Content Server
  - ODBC content
- **Profile-based export**: Standard Export gains much of its flexibility through its profile-based nature. The module defines the profile processing logic. The profile defines the batch processing scenario and the export commands that will be

applied to the batch. An export command defines the file format, the file content, the export destination, and other export rules. A customer can use a specific export profile for each Standard Export-based step in a process.

- **Filter-based batch processing**: The profile defines filters that select batch nodes for data export on basis of a filtering condition. The selected nodes are handled with a set of export commands defined for that filter.
- **Configurable file content**: Each export command, except data file, includes the file content definition. This definition specifies the file structure, the placeholders for the batch data, and other file parameters.
- **Configurable data type formats**: The format of some data types, such as date/time and number, is customer-defined. A customer can select any predefined data format, or they can define their own format in the formatted string.
- **Calculated expressions**: Standard Export can resolve IA values and supported functions in the expressions at runtime. Format expressions can be used to define file names, file paths, email header parameters, email attachment names, and file content.

5.2 Setting Up Standard Export

Most configuration information for Intelligent Capture client modules is defined during *module setup*. You can run a module in setup mode from Intelligent Capture Designer, from Intelligent Capture Administrator, or from a command prompt with appropriate arguments. For details, see *OpenText Intelligent Capture - Module Reference (ECPCORE-CMD)*.

When launched in setup mode, a client module displays the **Setup** window in which you can specify the module's configuration settings.

5.2.1 Setting Up Standard Export

Standard Export setup is an optional step in creating a capture process. You launch Standard Export in setup mode when you need to link an export profile to a 'standard export' step of a process.

The step's setup is not required if an export profile links that step in the module's MDF settings, which occurs is the following cases:

- If the process is designed in CaptureFlow Designer, the step's `Profile` IA value is assigned an export profile name on the CaptureFlow diagram.
- If the process includes scripting, the script that executes prior to the 'standard export' step unconditionally assigns the step's `Profile` IA value with an export profile name.

At runtime, the MDF setting takes a priority over the export profile name specified during setup.

To set up Standard Export:

1. Run the Standard Export module for setup as described in *OpenText Intelligent Capture - Module Reference (ECPCORE-CMD)*.
2. In the Standard Export module's setup window, expand the list of available export profiles and pick up one profile to set up your step.

   Note: To make an export profile available for Standard Export setup, you must create it using the Export Profiles editor of Intelligent Capture Designer and upload to the same server that stores the process that you set up. To learn more about creating export profiles, see *OpenText Intelligent Capture - Designer Guide (ECPCORE-CPD)*.

   Make sure that you select a profile that was specially created for your process and uses the same set of IA values. The *minimum task level* of the selected profile must not exceed the trigger level of the step that you set up.
3. Click **OK** to save the changes and exit the setup window.

5.3 Running Standard Export in Production

After the module has been set up, tested, and placed in a production environment, operators and administrators will run the module in production. For information related to common production tasks for unattended modules, see *OpenText Intelligent Capture - Module Reference (ECPCORE-CMD)*.

Note: The *Running Modules in Production Mode* section contains production topics that are common to many unattended modules and may not apply to this module.

5.4 Reference—Standard Export

The topics within this section contain reference information useful while using the application in setup or production.

5.4.1 IA Values

The topics in this section describe IA values that can be used with this application.

Intelligent Capture provides both common and module-specific IA Values. The Common IA Values described in *OpenText Intelligent Capture - Module Reference (ECPCORE-CMD)* are valid for all modules.

IA values defined in the Standard Export module are described in the following table. The **Type** column identifies the data type for each IA value followed by its modifier (input and/or output). The **Trigger Level** column identifies the IA value level in the batch (from 0 to 7). Trigger level set to 'T' indicates the level at which the export step will be triggered.

Table 5-1: Standard Export-specific IA Values

IA Value	Trigger Level	Type	Description
`Profile`	T	String, Input	The export profile name. If not empty, the specified profile is used for batch data export by default. If empty, the profile specified in setup mode is used.
`NumCreated`	T	Long , Output	The number of new files and email messages that were created without errors.
`NumOverwritten`	T	Long, Output	The number of duplicate files that were successfully overwritten.
`NumSkipped`	T	Long, Output	The number of duplicate files that were skipped.
`NumError`	T	Long, Output	The number of files or emails not created (skipped) due to an error (including `OverwriteBehavior` set to 'Error' and `ErrorBehavior` set to 'skip'). The process can check this value to determine if re-export is needed.
`NumUpdated`	T	Long, Output	The number of existing objects that have been updated as follows: • The properties of an existing Folder are updated. • An existing Document version is created.

Glossary

ACL

Access Control List

API

Application Programming Interface

DDoS

Distributed Denial of Service

DoS

Denial of Service

DSN

Data Source Name

ECM

Enterprise Content Management

GB

Gigabyte

GUID

Global Unique Identifier

HTTP

Hypertext Transfer Protocol

HTTPS

Hypertext Transfer Protocol Secure

IPP

Integrated ProcessFlow Project

MB

megabyte

MDF

Module Definition File

MTOM

Message Transmission Optimization Mechanism

NAT

Network Address Translation

OCR

Optical Character Recognition

ODBC

Open Database Connectivity

PDF

Portable Document Format

SOA

Service Oriented Architecture

SOAP

Service-Oriented Access Protocol

SQL

Structured Query Language

SSL

Secure Sockets Layer

SSN

Social Security Number

TIFF

Tagged Image File Format

URL

Uniform Resource Locator

UUID

Universally Unique Identifier

VBA

Microsoft Visual Basic for Applications

WS

Web Services

WSDL

Web Services Description Language

WSE

Web Services Enhancements

XML

Extensible Markup Language