Document Extraction Service

The Document Extraction service can be used to split individual documents out of a batch file to make each one a separate document. It can also be used to initiate EDI enveloping and outbound document processing. This service also enables you to batch multiple XML documents for processing.

The following table provides an overview of the Document Extraction service:

System name	DocumentExtractionService
Graphical Process Modeler (GPM) categories	All Services, Translation
Description	The Document Extraction service takes a file that contains one or more individual document, and attempts to find the map (from a specified list of maps) that produces output. The service extracts each individual document and puts it into the business process context with the name DOC-SPLIT-*N* where *N* is a 1-based number. Additionally, you have the option of having the extracted documents batched, EDI encoded, and EDI enveloped. The Document Extraction service can also extract XML sub-documents out of an XML compound document. Additionally, for the CHIPS and Fedwire standards, the Document Extraction service can be used to obtain values for the Application Sender ID and Application Receiver ID. These values can be passed to the EDI Encoder service to find an envelope for a document that needs to be extracted, encoded, and enveloped. Also, the Document Extraction service contains XML document extraction parameters that are supported when extracting XML documents. The values for these service parameters are XPath statements used by this service to locate the Application Sender ID and Application Receiver ID values within the XML document.
Business usage	The Document Extraction service allows you to concatenate many files into one, making tracking of the data more manageable.
Usage example	Your application generates purchase orders in a batch file, once a day. The application is scheduled to retrieve this file each day. When it is received, the first step in the business process has the Document Extraction service parse the file and produce an order document for each trading partner. Each document is passed on to the translator for conversion, and sent to the appropriate trading partner.
Preconfigured?	An instance of this service is created upon installation but is not configured, nor is any configuration required other than specifying values for the parameters when used within a business process.
Requires third party files?	No
Platform availability	All supported application platforms
Related services	EDI Encoder, EDI Envelope, For Each

Application requirements	None
Initiates business processes?	No
Invocation	By a business process
Business process context considerations	No
Returned status values	◆ Success – Extraction (and optional encoding and enveloping) were successful. ◆ Error – Errors were encountered during extraction, encoding, enveloping. Consult the status report in the business process context.
Restrictions	No

Requirements

To use the Document Extraction service, you should have advanced knowledge about translation maps and extended rules.

The Document Extraction service can only be used to extract documents from a batch file if the documents in that file are all in the same format. For example, they must all be in application (positional) format, or they must all be in EDI format.

Caution: An attempt to use this service to handle multiple data formats will produce unpredictable results and a potential loss of data.

How the Document Extraction Service Works

The Document Extraction service uses one or more translation maps to perform extraction using:

- ✦ Extended rules to find the start and end of a single document
- ✦ The Update standard rule to set sender ID, application sender ID, receiver ID, application receiver ID, and AccepterLookupAlias values from the document

The Document Extraction service provides the option to batch together similar documents during this extraction. If this option is specified, all documents extracted that have the same sender ID, receiver ID, and AccepterLookupAlias will be batched into a single document.

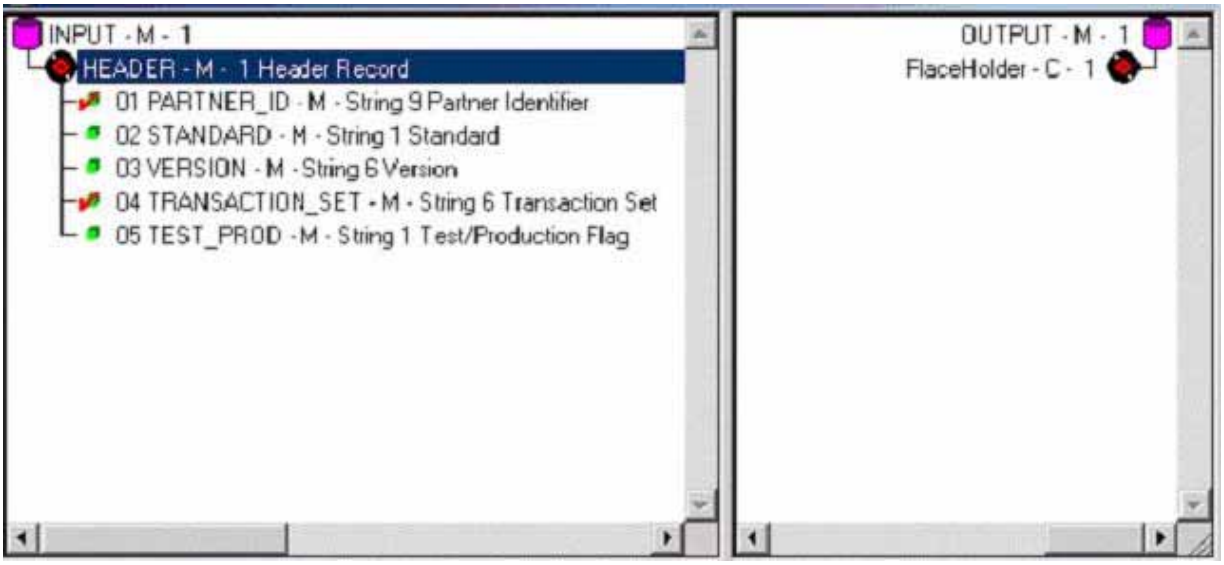
Additionally, the Document Extraction service contains XML document extraction parameters that are supported when extracting XML documents. The values for these service parameters (XMLAppSenderIDPath and XMLAppReceiverIDPath) are XPath statements used by the Document Extraction service to locate the Application Sender ID and Application Receiver ID values within the XML document.

Caution: You must write all XPath expressions (for example, Sender ID, Receiver ID, ALA, and so forth) in accordance with the extracted subdocument. See *Example of XML Code* on page 9 for an example of a correct XPath expression.

How the Document Extraction Service Uses Translation Maps

Translation maps define how a single document looks and where to find the sender ID, receiver ID, and accepter lookup alias values. Defining how a document looks is really the same as defining where a document starts and ends.

The following sample map is used to find a document that starts with a HDR record and ends with a SUM record:



This particular map defines only the first record of the file it is attempting to match, and it does not link any field from the input side of the map to the output side. The data is extracted using an extended rule on the first field of this header record. The function of the extended rule is to read and write records until it finds the end of the document. For this example, all records up to and including the SUM record are read and written.

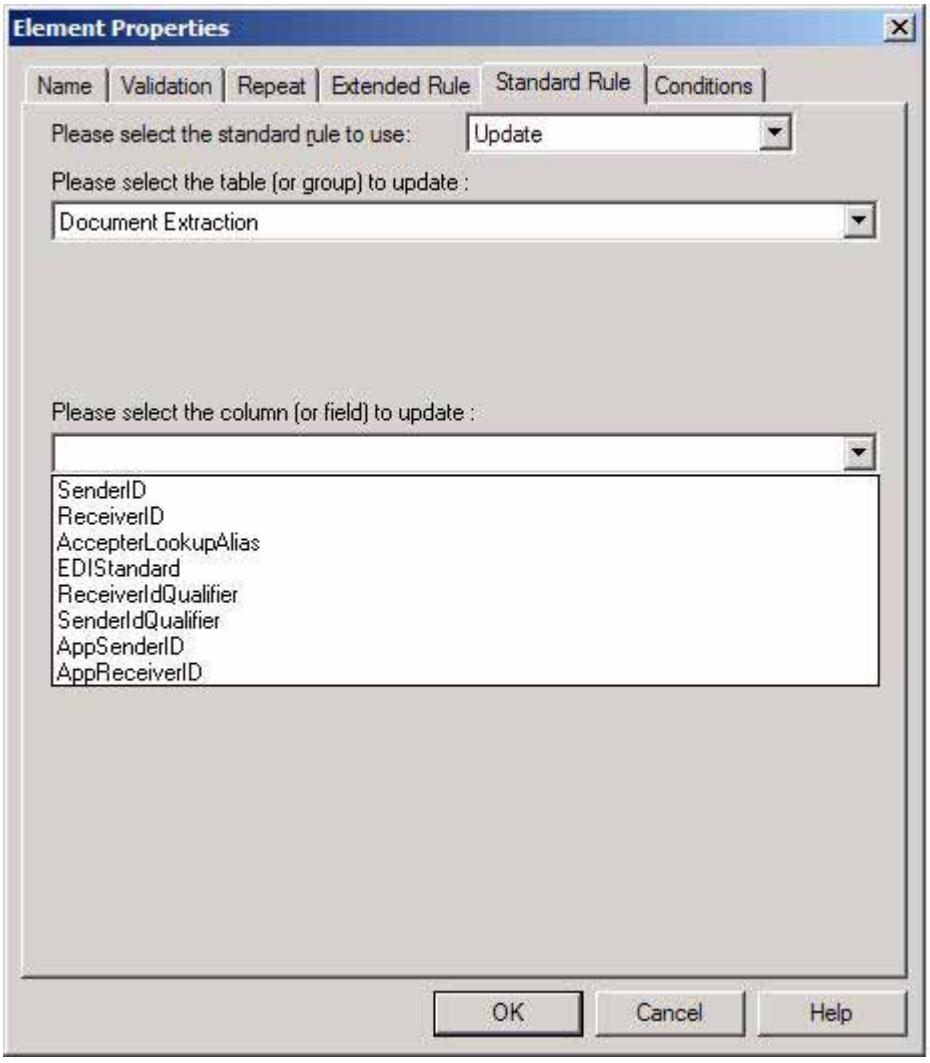
The following example shows the extended rule that is defined on the PARTNER_ID field:

```
//***** HEADER -> PARTNER_ID *****//

string[250] buffer;
string[3] match;
integer match_len;

// set these next two variables as desired
match = "SUM";  // the tag of the last record in the document
match_len = 3;  // the length of the tag

// read the block we're on and write it
readblock(buffer);
writeblock(buffer);

// keep reading and writing records until the end of the document
while readblock(buffer) do
begin
```

```
writeblock(buffer);
if left(buffer, match_len) = match then
   begin
      break;
   end
```

After specifying the start and end of the document, you can specify the sender ID, application sender ID, receiver ID, application receiver ID, and accepter lookup alias. The Document Extraction service relies on the translator to set these values using the Update standard rule. In the previous example, the receiver ID and accepter lookup alias are defined in the HDR record, and the sender ID is not used.

The following figure shows the standard rule that should be set up on the PARTNER_ID field:



It sets the receiver ID value to the value in that field in the data. The Document Extraction service accesses this value and puts it into process data. This will be explained in further detail later.

Similarly, the field that is to be used as the accepter lookup alias should have an Update standard rule defined with the Document Extraction table and AccepterLookupAlias column. In this example, the

AccepterLookupAlias value is populated by the data in the TRANSACTION_SET field. Similar logic also applies for the sender ID.

This example is fairly straightforward, where the metadata to be extracted (receiver ID, and accepter lookup alias) is in the first record. If this information is not in the first record of a document, you must include more than one record in the map. In this case, the extended rule on a field in the first record reads and writes records until it locates the record that contains the metadata. Then this record should contain a field with an extended rule that reads and writes blocks until the end of the document is reached.

What Happens When the Service Runs

When a document is processed by the Document Extraction service, zero or more extracted documents are produced. The documents are named with the convention DOC-SPLIT-1, DOC-SPLIT-2, … DOC-SPLIT-*n*. Values for sender ID, application sender ID, receiver ID, application receiver ID, and accepter lookup alias are placed into process data as child elements of the document.

In the following example of process data after the Document Extraction service runs, two documents are extracted from the primary document:

```
<?xml version="1.0" encoding="UTF-8"?>
<ProcessData>
  <PrimaryDocument SCIObjectID="server1:328145:f197c7bb55:-7250"/>
  <DOC-SPLIT-1 SCIObjectID="server1:328145:f197c7bb55:-7247">
    <ReceiverID>PETTEST1</ReceiverID>
    <AccepterLookupAlias>810</AccepterLookupAlias>
  </DOC-SPLIT-1>
  <DOC-SPLIT-2 SCIObjectID="server1:328145:f197c7bb55:-7246">
    <ReceiverID>PETTEST5</ReceiverID>
    <AccepterLookupAlias>850</AccepterLookupAlias>
  </DOC-SPLIT-2>
</ProcessData>
```

Any data found in the primary document that cannot be matched against any of the specified maps is placed in a document called *unrecognized*. The service status report describes what was processed by the Document Extraction service. The report includes the number of documents extracted and the number of each type if batch mode is set to Yes.

Note: When the Document Extraction parameter PDToProcessData is set to No, the DOC-SPLIT information will be placed in an array named SplitDocs. Use the For Each Document service to process the SplitDocs array. Each iteration through the For Each Document service will update process data with the current DOC_SPLIT and remove the previous split.

Implementing the Document Extraction Service

To implement the Document Extraction service, complete the following tasks:

1. Create the translation maps necessary to define how a single document looks and where to send the sender ID, application sender ID, receiver ID, application receiver ID, and accepter lookup alias values.
2. Create a Document Extraction service configuration. See *Creating a Service Configuration*.
3. Configure the service. See *Configuring the Document Extraction Service*.
4. Create a business process that includes the Document Extraction service and enable it.
5. Test and run the business process and the adapter.
6. To batch XML documents, set the **XMLRootTagForBatches** property to a non-null value. See *Batching Multiple XML Documents* on page 10 for more information.

Note: If the XMLRootTagForBatches property is null, the Document Extraction service generates malformed XML (a document without a root tag) if the batch contains more than one sub-document.

Configuring the Document Extraction Service

To configure the Document Extraction service, you must specify field settings in the Graphical Process Modeler (GPM). For general information about service configurations, see *Creating a Service Configuration* on page 15.

The following table describes the fields used to configure the Document Extraction service in the GPM:

Field	Description
Config	Name of the service configuration.
BatchLikeDocuments	Whether to combine documents extracted by the service into a single document, or split each document out individually. Valid values are Yes and No. ◆ If Yes is specified, documents that are extracted by the service are combined into a single document if they have the same sender ID, receiver ID, and accepter lookup alias. This parameter also allows you to batch XML documents. ◆ If No is specified, each document is split out individually, regardless of sender ID, receiver ID, and accepter lookup alias values.
DocExtractMapList	List of map names defined in the Map Editor that should be used to extract documents from a single batch file. Valid values are the names of the maps created. Make sure that all maps in this list are active in application. Map names in the list are separated by a space. **Note:** This parameter is mutually exclusive with the XMLInput parameter. If XMLInput is set to Yes, this parameter will be ignored.
DOCUMENT_NAME_PREFIX	Allows the user to specify a name prefix other than **DOC-SPLIT-**. When specified, the document extraction service will label the document with the DOCUMENT_NAME_PREFIX specified by the user.

Field	Description
EDIEncodeDocument	Whether each document extracted should be immediately EDI-encoded. Valid values are Yes and No. Must be set to Yes if EDIEnvelopeDocument is set to Yes. Otherwise, the service will generate an error and halt the business process. **Note:** If this parameter and the XMLInput parameter are set to Yes, the XMLSenderIDPath, XMLReceiverIDPath, and XMLAccepterLookupAliasPath parameters must be set.
EDIEnvelopeDocument	Whether each document extracted and encoded should be immediately EDI-enveloped. Valid values are Yes and No. If this parameter is set to Yes, the EDIEncodeDocument parameter must also be set to Yes. Otherwise, the service will generate an error and halt the business process.
ErrorBP	Business process to execute when unrecognized data is received.
ErrorOnUnrecognizedData	Whether to generate an error when unrecognized data is received. Valid values are Yes and No.
HALT_ON_TRANS_ERROR	Whether to stop processing transactions when a translation error occurs. Valid values are: ◆ Yes – Stop processing transactions when a translation error occurs. This is the default in Immediate Enveloping mode. ◆ No – Continue processing transactions if a translation error occurs. This is the default in Deferred Enveloping mode.
PDToProcessData	Specifies whether split documents will be stored in process data or in the business process context. Valid values are: ◆ Yes – Include each split document in process data (default). ◆ No – Do not include each split document on process data; instead, store the information for each document in an array named SplitDocs. Retrieve documents from SplitDocs one at a time using the For Each Document service. The PDToProcessData parameter can be used to improve performance by reducing the overhead associated with persisting large process data information. The major performance improvement is realized at the current step (Document Extraction) and each subsequent step by persisting a much smaller process data which does not contain multiple split documents. By using the For Each Document service in conjunction with PDToProcessData = No, only the current document is on process data. this avoids repetitive writing of non-current documents. Once documents are in the SplitDocs array, they can be retrieved only by using the For Each Document service. Setting this parameter to No, combined with setting EDIEncodeDocument and EdiEnvelopeDocument to Yes, may yield an improvement by reducing the amount of persisted process data information. The benefit will depend on the complexity of the business process. In this situation, you would not use the For Each Document service, because the documents are passed to invoked business processes.

Field	Description
UseInputEdiDelimiters	Whether the delimiters read from the syntax record for an EDI map are used when generating the extracted documents. Valid values: ◆ Yes – Extract the document using the same delimiters that were used to read the input. ◆ No – Do not use the delimiters. **Note:** When using the Document Extraction service to split documents in a delimited EDI file, only one set of delimiters is supported. The input file cannot contain multiple documents that each use a different set of delimiters. Therefore, if you have multiple documents using different delimiters, you must first send the data through the EDI Deenvelope service (in Document mode) to split out the individual documents that use differing delimiters. When you invoke the EDI Deenvelope service, the Mode service parameter must be set to Document. This instructs the EDI Deenvelope service not to bootstrap a business process after it finishes splitting the input file. You must also update the customer_overrides.properties file to include the START and END tag of the documents to be extracted, as well as the delimiters (or the location of delimiters) to use.
XMLAccepterLookupAliasPath	XPath-like string indicating the location of the XMLAccepterLookupAliasPath element tag. Required if XMLInput and EDIEncoding are set to Yes. Valid value: SubDocument/AccepterLookupAlias
XMLEDIEnvelopeStandard	Standard to be used by this instance of the Document Extraction service. For example: X12, EDIFACT, TRADACOMS, or CII. Optional.
XMLInput	Specifies whether XML input will be received. Required if extracting XML documents. Valid values: ◆ Yes – XML documents can be extracted. ◆ No – XML documents will not be used as input. **Note:** If this parameter is set to Yes, at least the XMLRootTag parameter must be set. This parameter is mutually exclusive with the DocExtractMapList parameter. If this parameter is set to Yes, the DocExtractMapList parameter will be ignored.
XMLReceiverIDPath	XPath-like string indicating the location of the XMLReceiverIDPath element tag. Required if XMLInput and EDIEncoding are set to Yes. Valid value: SubDocument/Receiver
XMLReceiverIDQualifierPath	XPath-like string indicating the location of the XMLReceiverIDQualifierPath element tag. Optional. Valid value: SubDocument/ReceiverQual

Field	Description
XMLRootTagForBatches	Root tag for the XML document. Required if the **BatchLikeDocuments** and **XMLInput** parameters are set to **Yes**. Valid value is the DocumentRootTag. If this parameter is null, the batched XML documents are generated. However, the end document will be malformed if it contains more than one sub-document. If you define the XMLRootTagForBatches parameter, the Document Extraction service generates a valid XML document with this defined value as the root tag of the document.
XMLRootTag	Root tag for the subdocument. Required if XMLInput is set to Yes. Valid value: SubDocument
XMLSenderIDPath	XPath-like string indicating the location of the XMLSenderIDPath element tag. Required if XMLInput and EDIEncoding are set to Yes. Valid value: SubDocument/Sender
XMLAppSenderIDPath	XPath-like string indicating the location of the XMLAppSenderIDPath element tag. Optional. Valid value: SubDocument/AppSenderID
XMLSenderIDQualifierPath	XPath-like string indicating the location of the XMLSenderIDQualifierPath element tag. Optional. Valid value: SubDocument/SenderQual

Example of XML Code

An example of XML code that could be used with the Document Extraction service when extracting XML sub-documents from an XML compound document is shown below:

Note: You must write all XPath expressions (for example, Sender ID, Receiver ID, ALA, and so forth) in accordance with the extracted subdocument. For the code example provided below, the following XPath for Sender ID is: /SubDocument/Sender.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
  <CompoundDocument>
    <SubDocument>
      <!-- subdocument # 1 -->
      <Sender>DOCEXTRACTTEST</Sender>
      <SenderQual>ZZ</SenderQual>
      <Receiver>PETTEST1</Receiver>
      <ReceiverQual>AA</ReceiverQual>
      <AccepterLookupAlias>810</AccepterLookupAlias>
      <Manifest>
      ...
      </Manifest>
    </SubDocument>
    <SubDocument>
      <!-- subdocument # 2 -->
      ...
    </SubDocument>
  ...
</CompoundDocument>
```

Batching Multiple XML Documents

The application enables you to specify a root tag if you are batching multiple XML documents, which allows you to generate valid XML output. There are two ways to perform XML batching, depending on whether or not the Document Extraction service instance is configured to do EDI Enveloping as part of the extraction. If the service instance is configured to do EDI enveloping, you do not have to use a root tag even if you are batching XML documents, because the service puts all of the documents that would go into a batch in the same call to the EDI Envelope service (which produces the same effect as batching without putting all of the documents in the same file with a common root tag). If the service instance is not configured to do EDI enveloping, the only way to batch and create valid XML output is to provide a root tag to use. Therefore, if you are doing EDI enveloping, the root tag can be NULL. If you are not doing EDI enveloping, the root tag can also be NULL, but will not generate a valid XML output document.

Please see the following table for a further explanation of when you should specify a root tag:

Value Specified for the BatchLikeDocuments Parameter (flag)	Value Specified for the EDIEnvelopeDocument Parameter (flag)	Value Specified for the XMLRootTagForBatches Parameter (string)	Document Extract Service Result
Yes	No	Present (non-null)	Valid XML output
Yes	No	NULL	Invalid XML output
Yes	Yes	Present (non-null)	Documents are EDI processed as usual
Yes	Yes	NULL	Documents are EDI processed as usual