Easily integrate unstructured and semi-structured data into SAP NetWeaver Process Integration (formerly XI) using the Conversion Agent

by Prasad Illapani

Prasad Illapani
PI RIG Specialist,
SAP Labs LLC

Prasad Illapani is a Platinum RIG Specialist with the SAP NetWeaver Integration team at SAP Labs LLC in Redmond, Washington. He focuses on both the development and systems areas of SAP Process Integration (PI). He is a regular speaker at WIS, TechEd, and SDN conferences and works closely with SAP Product Definition /Management, Development, and Field Services teams in the PI area. He also teaches PI 3.0 training classes and workshops for partners, customers, and consultants. Prior to joining SAP, he worked as an SAP technical consulant with an SAP partner company.

(full bio appears on page 102)

SAP NetWeaver Process Integration (PI) — formerly called SAP Exchange Infrastructure (XI)[1] — offers SAP customers an unprecedented level of flexibility in connecting their SAP and non-SAP systems. Out-of-the-box, PI can communicate via RFC and IDocs, and it includes a flexible plug-in-style adapter framework that enables customers to extend PI to send and receive information in other formats, including EDI, RosettaNet, and dialects such as ACORD from the auto industry. You simply purchase and install one of the dozens of adapters offered by SAP and SAP partners, and the adapter translates the specific format to PI-specific SOAP XML format at runtime.

But adapters are not your only option for converting data to a PI-understandable format. This article explores a powerful (often less expensive) alternative called the Conversion Agent, which dynamically converts unstructured messages from Microsoft Word, Excel, PowerPoint, PDF, plain text, etc., and semi-structured formats such as HL7, SWIFT, HIPA, ANSI X12, and COBOL to PI-understandable SOAP XML, so that you can easily integrate the information you need into your back-end systems. You can also reverse the process and use the Conversion Agent to transform XML into these unstructured or semi-structured formats, so that you can quickly and easily share information with your business partners. The Conversion Agent is offered by SAP in partnership with Itemfield, Inc.[2] for use with SAP NetWeaver ’04 and higher and consists of a set of libraries that are deployed on the PI server’s SAP J2EE Engine. PI’s Adapter Engine accesses these libraries during runtime to convert messages to XML, and it is gaining in popularity among PI implementers because of its simplicity and ease of use.

[1] SAP renamed Exchange Infrastructure (XI) to Process Integration (PI), because this name is more suggestive of what the software component “does.” Version 7.0 was the last version of this product branded as “Exchange Infrastructure.”

[2] Informatica acquired Itemfield in the fourth quarter of 2006.

Note!

The Conversion Agent is free for development and test landscapes. A license fee is required for production use (see SAP Note 894815 for details). You can download the Conversion Agent from the SAP Service Marketplace at http://service.sap.com.

This article provides an introduction to the Conversion Agent for technical developers who focus on integration scenarios. It discusses some of the key features of the tool, prerequisites for using it, the installation process, and configuration of the data conversion components and the sender/receiver communication channels that convert the data into PI-specific SOAP-XML. I take you on a tour of the tool's capabilities and show you how easy it is to convert the previously mentioned formats into XML format through a step-by-step example. The example walks through a complete end-to-end business scenario of converting a PDF file to XML and shows you how to integrate the Conversion Agent tool with PI.

Note!

This article is based on the session "Building Transformations Using the SAP Conversion Agent by Itemfield" (EPI250) presented at SAP TechEd '06. It also assumes working knowledge of PI, and therefore does not cover the configuration of PI design objects in the example. For additional information on these areas, please refer to the SAP Help Portal (http://help.sap.com).[3]

[3] For a detailed explanation of the PI configuration steps, see http://help.sap.com, and also the *SAP Professional Journal* articles "A Beginner's Guide to Implementing SAP Exchange Infrastructure (SAP XI) — Paving the Way to Seamless Integration" (March/April 2005), "A Beginner's Guide to Implementing SAP Exchange Infrastructure (SAP XI) — Designing and Configuring an SAP XI Integration" (May/June 2005), and "Mastering SAP NetWeaver Exchange Infrastructure 7.0: Successfully integrate SAP applications into a non-SAP, non-XML world" (March/April 2007).

Overview of the Conversion Agent tool

The Conversion Agent tool consists of the following key components:

- The **Conversion Agent Studio**, which is installed on the developer's PC, is an Eclipse-based development tool you use to develop data conversions and perform various types of data transformations. You then deploy the defined conversion component as a service to be executed in the Conversion Agent Engine.

Note!

There is an optional set of Conversion Agent libraries that contain predefined conversion components and schemas for various standard formats, such as the XML-based healthcare industry standard format HL7 (Health Level 7). If used, these libraries should be installed in the development environment in which you are running the Conversion Agent Studio. In the example later in the article, we use a predefined HL7 library and schema to create a conversion component.

- The **Conversion Agent Engine**, which is installed on the SAP J2EE Engine running PI, receives the deployed data conversion component as a service from the Conversion Agent Studio, and then executes the actual conversion behind the scenes.

Figure 1[4] shows how the Conversion Agent tool fits into the PI architecture. As you can see, PI is divided into two phases. In the design and

[4] This figure is adapted from the *SAP Professional Journal* article series "A Beginner's Guide to Implementing SAP Exchange Infrastructure (SAP XI) — Paving the Way to Seamless Integration" (March/April 2005) and "A Beginner's Guide to Implementing SAP Exchange Infrastructure (SAP XI) — Designing and Configuring an SAP XI Integration" (May/June 2005).

Note!

The Conversion Agent Engine offers various options for running Conversion Agent services, including the Conversion Agent Process Module, the Conversion Agent Java API, and additional interfaces. This article focuses on using the Conversion Agent Process Module, which is the mechanism that integrates the Conversion Agent with PI.

configuration phase, you use the Integration Builder toolset, which consists of the Integration Repository for developing the processes, interfaces, and mappings of an integration scenario, and the Integration Directory for configuring the channels for sending and receiving the messages containing the data to be communicated. In the runtime phase, the Integration Server processes the message exchange and provides connectivity to back-end systems through ABAP and Java-specific adapters. You will see how these PI components come into play when I take you through the example later in the article. You manage and monitor the runtime using the Runtime Workbench, which I will also show you how to use to monitor the message exchange. The System Landscape Directory (SLD) is a system-wide, centralized repository PI uses to access information on systems and solutions deployed in the landscape.[5]

The Conversion Agent Engine integrates into PI through the Conversion Agent Process Module, which connects to the Java-based PI Adapter Engine. A data transformation project is deployed from the Conversion Agent Studio to the Conversion Agent Engine as a service, where the conversion is performed. The sender and receiver communication

[5] Several additional SAP applications use the SLD to access system landscape information, including SAP Solution Manager, Supplier Relationship Management (SRM), and Auto-ID Infrastructure (AII). For a detailed introduction to the SLD, see the article "A system administrator's practical guide to SAP System Landscape Directory (SLD)" (*SAP Professional Journal*, November/December 2006).

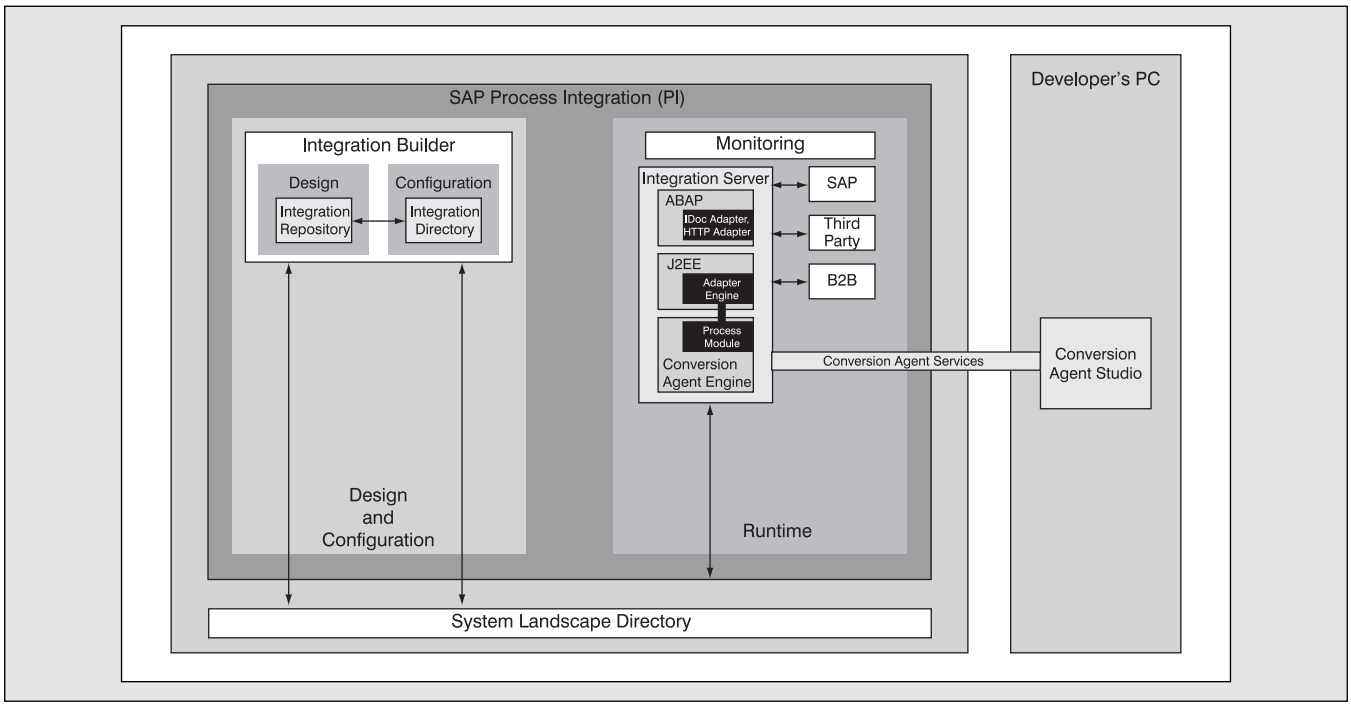


Figure 1 The Conversion Agent in an SAP Process Integration (PI) landscape

channels (configured as sender and receiver adapters on the PI Integration Server) then access the Conversion Agent Process Module to transform the source format to XML format, which is then converted by the Adapter Engine to a PI-compatible SOAP XML message.

Now that you have an understanding of the key components of the tool and how the tool fits into the PI architecture, let's take a look at how to get started by installing it.

Downloading and installing the Conversion Agent

Before we take a look at the Conversion Agent tool itself, we need to download and install the correct software.

> ***Note!***
>
> To install the Conversion Agent, XI 3.0 or higher — usage type PI as of SAP NetWeaver 2004s — must be installed, configured, and running. The Conversion Agent runs on Windows, Linux, AIX, HP-UX, and Solaris operating systems.

You can download the Conversion Agent from the SAP Service Marketplace Software Distribution Center at http://service.sap.com/swdc. Navigate to Download → Support Packages and Patches → Entry by Application Group → SAP NetWeaver → SAP NETWEAVER:

- For SAP NetWeaver '04, select SAP NETWEAVER 04 → Entry by Component → Process Integration (PI/XI) → CONVERSION AGENT 1.0.
- For SAP NetWeaver 2004s, select SAP NETWEAVER 2004s → Entry by Component → Process Integration (PI/XI) → CONVERSION AGENT 7.0.

Downloading and installing the file consists of three steps:

1. Extract the download file and deploy the contents.
2. Install and test the Conversion Agent Engine.
3. Install the Conversion Agent Studio.

Let's take a closer look at each of these steps.

> ***Note!***
>
> The following steps are based on the installation of version 1.0 of the Conversion Agent on a Windows 32-bit system with either XI 3.0 or 7.0 installed and configured.

Step 1: Extract the download file and deploy the contents

Select and download the appropriate software based on the operating system platform and release. For example, for Win32, SAPXI3.0-SPS18, the file is SAPNWCM19_0-20001444.sca.[6] Extract the file to a temp folder, such as C:/temp, and deploy the following files on the PI Integration Server using the Software Deployment Manager (SDM), as shown in **Figure 2**:

- CM_JavaAPI.sda (the Conversion Agent Java API)
- CM_RemoteSupportInterface.sda (an interface for viewing event logs in a Web browser rather than in the Conversion Agent Studio)
- CM_TransformModule.sda (the Conversion Agent Process Module)
- ConversionAgent.sda (the Conversion Agent Studio with the required Eclipse toolset, the

[6] In earlier versions, the Conversion Agent was called the Content Master. This name, along with the abbreviation CM, is still used in some file names and documentation.

transformation libraries, and the Conversion Agent Engine)

> ***Note!***
>
> The specifics of the deployment and the installation of the Conversion Agent Engine and Conversion Agent Studio will depend upon your particular organization and needs. See the documentation available at the SAP Help Portal at http://help.sap.com for detailed steps that you can adapt to your own environment.

Step 2: Install and test the Conversion Agent Engine

After you have installed the Conversion Agent Engine, you need to make sure that it is running correctly. Follow these steps:

1. Copy the entire contents of the Conversion Agent installation directory C:\Program Files\SAP\ConversionAgent\setupTests\TestCME to the ServiceDB subdirectory.
2. Open a command prompt and enter the following command:

```
CM_console TestCME
```

You will receive the following success message in response:

```
<Result> Test Succeeded </Result>
```

If you don't see a success message, you must recheck the installation process to make sure that the files are installed successfully.

Step 3: Install the Conversion Agent Studio

In this last step, you need to install the Conversion Agent Studio on your PC (along with the libraries, if you choose to use them, which we do for the example).

As mentioned earlier, the Conversion Agent Studio is the Eclipse-based IDE that you use to design, configure, and deploy Conversion Agent projects, which store the files used to execute data transformations for semi-structured and unstructured data formats. You can create the following four types of transformation components:

- **Parsers** convert source documents in unstructured or semi-structured formats to XML format. For example, if you wanted to integrate information from a PDF document into your system, you could convert it into XML, which is what we'll do for the example business scenario later in the article.

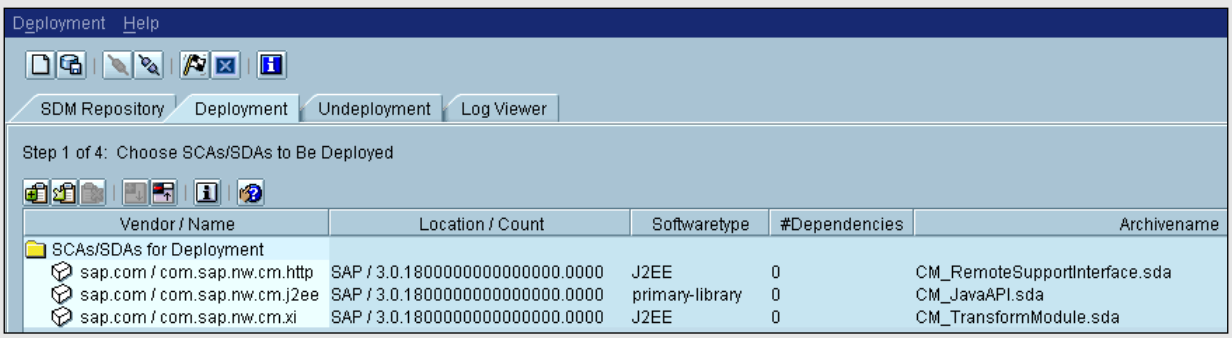


Figure 2 Deploying files with the SDM tool

- **Serializers** convert XML documents to output documents, which can be in any format. A serializer is the opposite of a parser. In the example, we use a serializer to further convert the XML that results from the PDF conversion into an HL7 format that is compatible with a back-end Health Information System (HIS).
- **Transformers** modify data. The input and output documents can be in any format. For example, you could modify text extracted from a source document by changing it to all uppercase. You can also use a transformer within another transformation type to act on only a portion of a document instead of the entire document. For instance, in the example we will use a transformer within a parser to modify some of the converted text.
- **Mappers** convert XML documents to a different XML structure or schema. For example, you could take a document that lists employee names and their departments together on a single line and convert it into a document that lists the names and departments on separate lines.

Within each of the four key transformation components, you can also optionally use the following components to further customize the data conversion:

- **Anchors** define how the Conversion Agent Engine should search for data within a document, what data it should extract from the document, and where it should store the extracted data in the output. In the example, we use anchors to extract specific text from a PDF file for conversion.
- **Actions** perform operations on the data in a document, such as concatenating strings, computing sums, or querying a database for additional data.
- **Formats** define the overall format of documents, such as the delimiters and processors associated with the selected format, which the Conversion Agent then uses to interpret the documents. Some supported formats include binary, HTML, RTF, and text. There is also a custom format available, which is a generic format that can be used for any type of document. With the custom format, which is the type that we use for the example, you must define delimiters and processors yourself.
- **Document processors** are components that convert the format of a complete document to another format for processing. You can use a document processor as a preprocessor, which converts the format of a source document prior to parsing. In the example, we convert the source PDF document to UTF-8 (Unicode Transformation Format based on 8-bit representation) format, which is much easier to parse than the binary PDF format.

Figure 3 shows the main areas of the Conversion Agent Studio workbench in which you design and implement data transformations. Let's take a closer look at the different areas of the workbench:

- The pane at the upper left displays the Conversion Agent Explorer, which shows the various stored conversion projects and files. You can open up files displayed in the Explorer by simply double-clicking on the file name. Double-clicking on an XML file, for example, opens the file in its own Internet Explorer window. Double-clicking on a TGP file, which is a data transformation script file, opens the file in the IntelliScript editor.
- The pane in the upper middle — or "script view" — contains the IntelliScript editor, which displays the data transformation script in a hierarchical tree-like structure. Using this editor, you can define parsers, serializers, transformers, mappers, and their nested components.
- The pane at the upper right — or "source view" — shows the source document associated with the open data transformation script. Using the tabs at the bottom of the pane, you can switch between Source, which displays the text of the source

> ***Note!***
>
> The script view and the source view are the two main panes that you work with in the Conversion Agent Studio when creating a parser. The visualization and the interaction between the two panes simplifies the parsing process through an intuitive drag and drop feature.

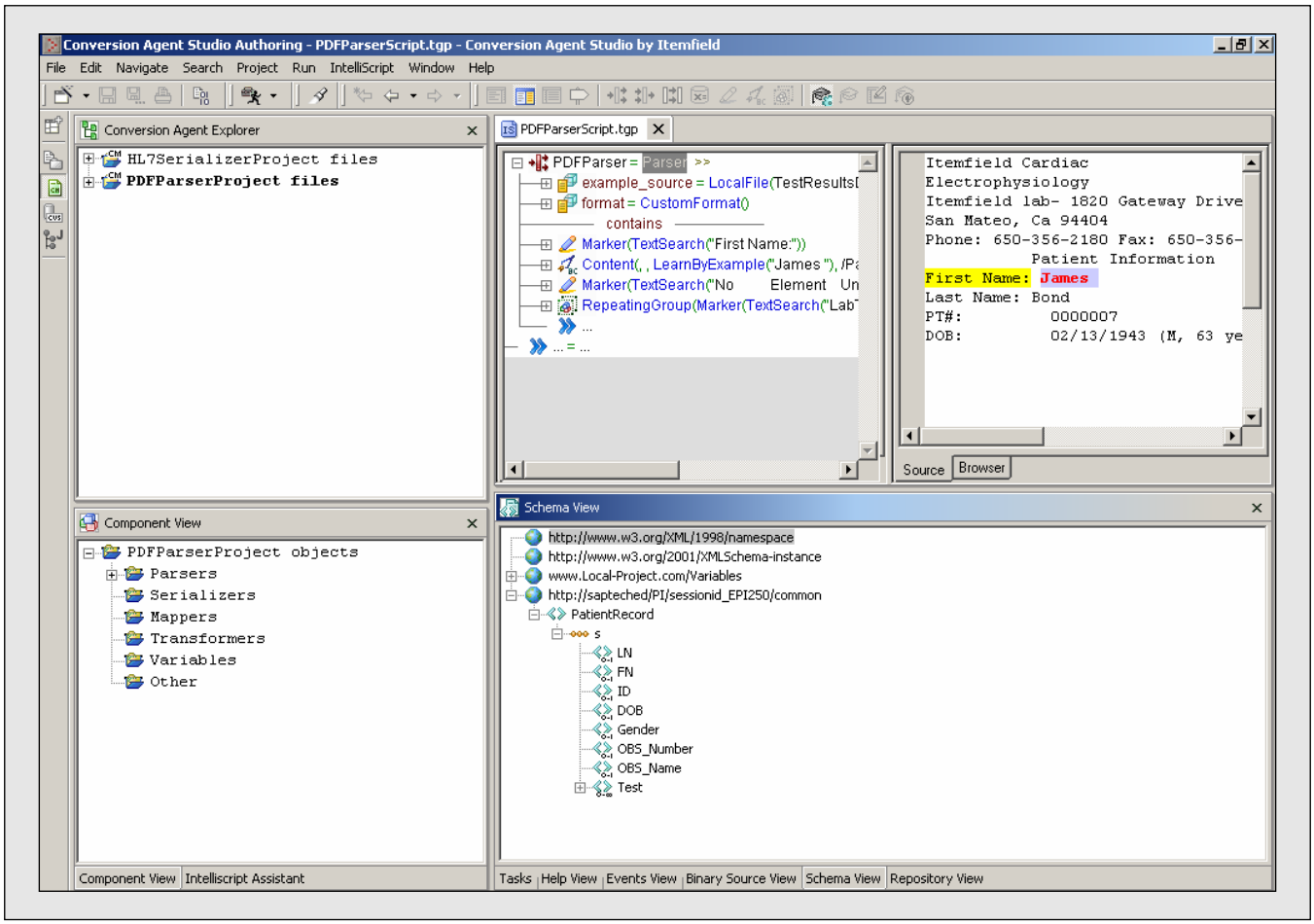


Figure 3 The Conversion Agent Studio development environment

document, and Browse, which displays the source document in its original, formatted form in an Internet Explorer display.

- The pane at the lower left includes two views — Component and IntelliScript. The Component view displays all of the top-level components (parsers, serializers, transformers, mappers, etc.) that are defined in the project. The IntelliScript Assistant view has the same options as the IntelliScript editor, but in a dialog-box format. The IntelliScript Assistant is active only for a few types of Conversion Agent components, such as certain types of anchors.

- The pane at the lower right contains various views, which you can switch between using the tabs at the bottom of the pane. Tasks summarizes syntax errors in a transformation or schema, the Help View provides interactive help when you are editing a script, and the Events View provides an interactive log to help with debugging.[7] The Binary Source view displays the binary code of the source document, the Repository view shows deployed transformations running as services in the Conversion Agent Engine, and the Schema View, which is shown in the figure, displays the name-spaces and data holders (XML elements, XML attributes, and variables) that are defined for the

[7] You can also optionally view the events log in a Web browser using a remote support interface (this is the CM_RemoteSupport Interface.sda file included with the download). See the documentation available at http://help.sap.com for details on how to use this interface.

project in its associated XSD schema. The Schema View also supports drag and drop mapping of source content to schema fields to define content anchors.

> ***Note!***
>
> Each of the workbench panes is resizable, so that you can customize the working environment to fit your needs.

Now that the Conversion Agent installation is complete, and you are familiar with the development environment you'll be working with, let's look at how this all works in an example.

Introduction to the business example: Converting a PDF document to XML

Consider the example scenario shown in **Figure 4**, where we have a PDF document containing data we want to extract and send to one or more back-end systems. In this case, we have a laboratory form containing medical test results that we want to extract and ultimately send to a back-end Health Information System (HIS). To do this, we will define two conversion components — a parser to convert the document to XML, and a serializer to convert the XML to an HL7 flat file.

> ***Note!***
>
> I could have chosen any type of unstructured document (Microsoft Excel spreadsheets or plain text files), or business scenario (time entry data, finance data, inspection lot test results, or plant maintenance) to demonstrate a transformation. The objective here is to demonstrate the technology through a simple example. You can apply what you learn here to any other scenario; you'll just need to slightly modify the steps.

Figure 4 summarizes the conversion process:

- The sender communication channel (i.e., the sender adapter defined in the PI Integration Server) polls the file using the defined communication channel parameters.
- The parser executes the transformation module parameters to convert the PDF document to XML format.

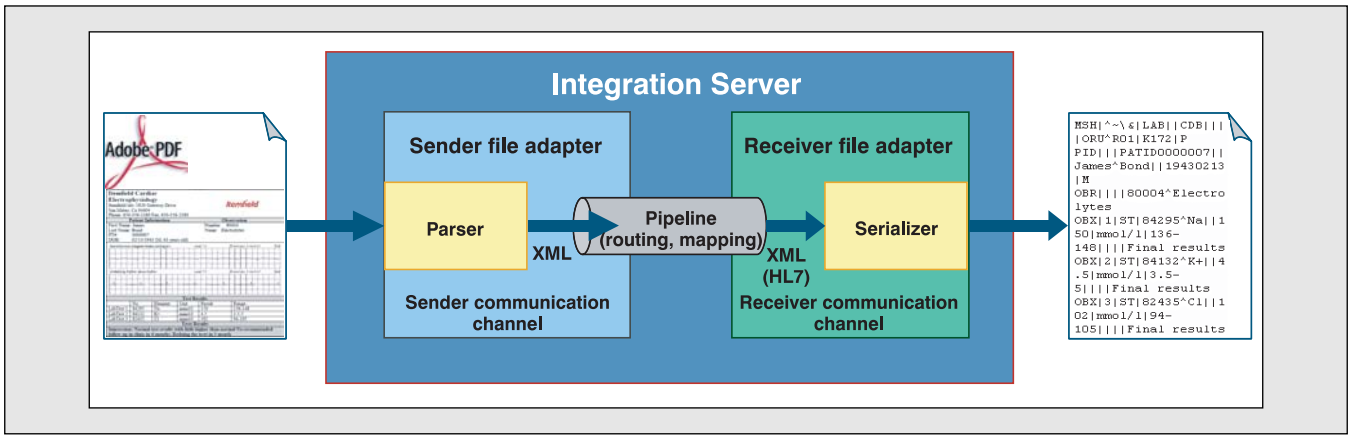


Figure 4 Conversion process overview

- The resulting XML message is called through the Integration Server pipeline by the receiver communication channel (i.e., the receiver adapter defined in the PI Integration Server), and is further processed by a serializer that maps it to the HL7 format.

Now that you have an idea of what's involved in the entire process, let's take a look at the following individual steps in more detail:

1. Create a parser project.
2. Create an HL7 library project.
3. Deploy the parser and HL7 library projects.
4. Configure the sender and receiver communication channels.
5. Test the business scenario.
6. Monitor Conversion Agent messages in the Integration Server.

Over the course of the remainder of this article, we will walk through each of these steps in detail.

Note!

The following steps assume that most of the necessary PI configuration work has already been done in the Integration Repository of the PI Integration Server, which is where all the design objects for the business scenario are created and activated, including configuration of the required integration processes, interfaces (both outbound and inbound), and mappings, along with some additional configuration in the Integration Directory, including creating business services for both the sender and receiver, interface determinations, and receiver agreement objects. The only PI configuration that has not yet been completed is the sender and receiver communication channels, which require information on the data conversion projects we'll be creating. I show you how to configure these in Step 4, which begins on page 95.

Step 1: Create the parser project

Parsers convert unstructured data — e.g., from Microsoft Word, PDF, plain text, etc. — to XML data that can be used in a back-end system. Because we want to extract data from a PDF document, our first step is to create a parser project. The PDF document, shown in **Figure 5** on the next page, consists of patient information, with header data fields and test results data embedded in the document body. We want our parser to extract the header data and the test results data from the document and transform the data into a predefined XML format specified by a previously defined XSD schema.

Note!

XSD schema definitions are beyond the scope of this article. For more details on creating schemas, see the SAP Help Portal at http://help.sap.com.

Creating the parser project consists of the following tasks:

- Create a new parser project.
- Identify the header data to extract.
- Identify the body data to extract.

Create a new parser project

The first task is to create a new parser project within the Conversion Agent Studio:

1. Start the Conversion Agent Studio development environment. From the Windows Start menu, choose SAP Conversion Agent → Conversion Agent Studio for Eclipse.
2. From the Window menu, choose Open Perspective → Conversion Agent Studio Authoring.
3. From the File menu, choose New → Project to

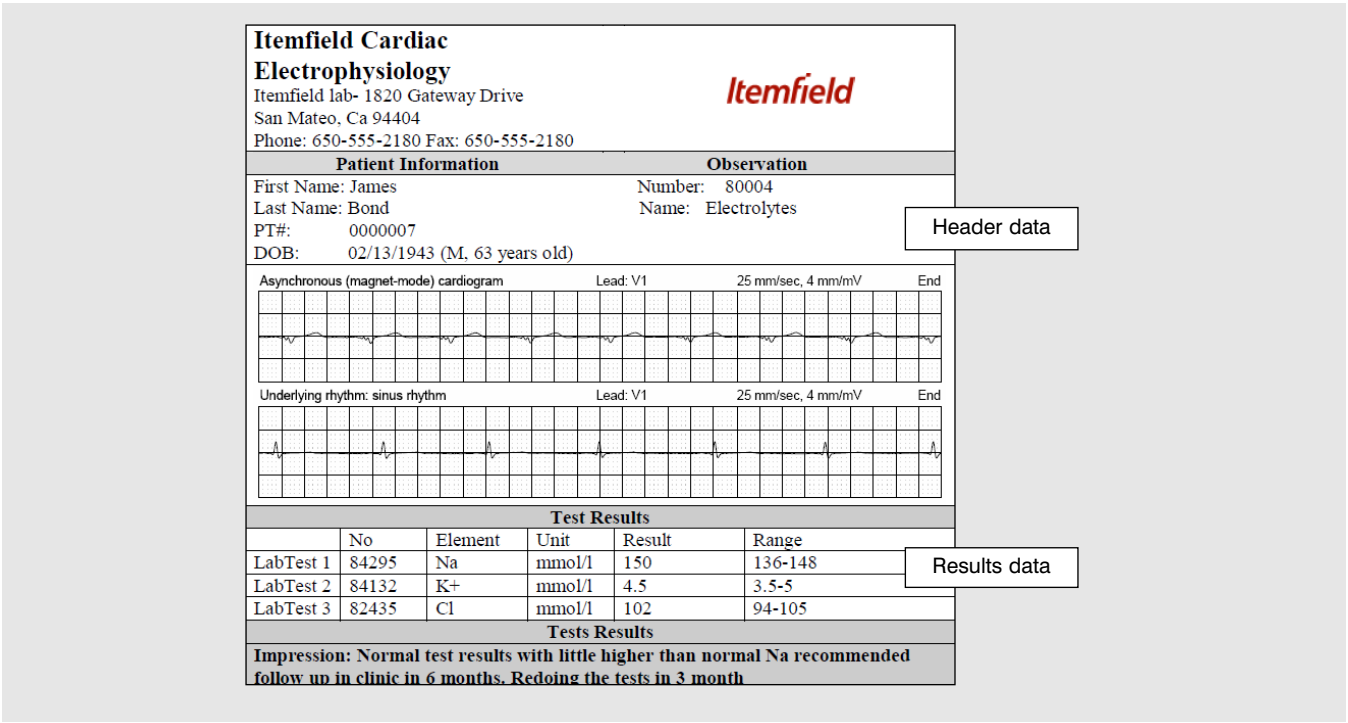


Figure 5 Structure of the source PDF from which we'll extract the header and test result data

start the new project wizard. You will see the first screen of the wizard, which is shown in **Figure 6**. Choose Conversion Agent and then choose Parser Project. Click on Next.

4. In the next screen, enter names for the project, parser, and data conversion script — PDFParserProject, PDFParser, and PDFParserScript, respectively — and click on Next.

5. Now we need to load the target schema — that is, the XML structure to which we want to map the incoming PDF document. Click on the Browse button, navigate to the folder containing the target schema (e.g., C:\CAProject), and select the previously defined schema (TestResults_schema.xsd in the example), as shown in **Figure 7**. Click on Next.

6. In the next screen, we need to specify the source data that will be used in the transformation. Because the source data is in a specific file, select the File radio button, as shown in **Figure 8**, and click on Next.

7. Next, we select the source file containing the data

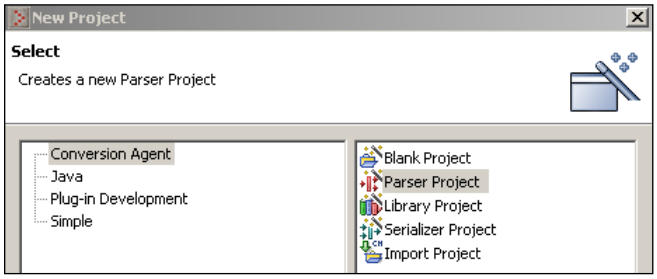


Figure 6 New parser project wizard screen

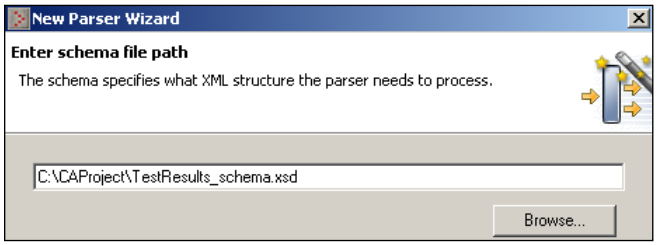


Figure 7 Specify the target schema

we want to convert — in the example, the PDF document. Click on the Browse button, navigate to the folder containing the file (e.g., C:\CAProject),

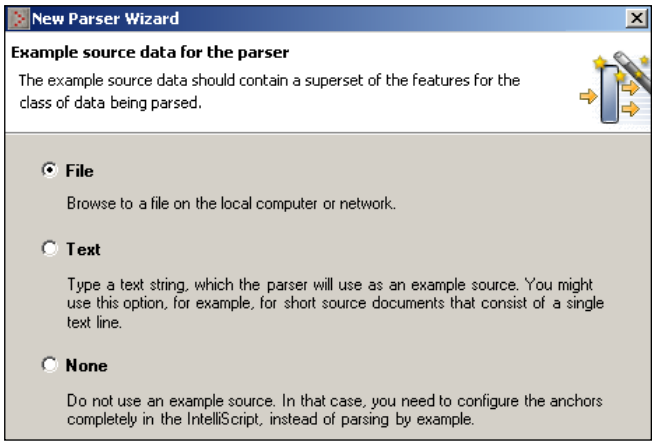


Figure 8 Select the File radio button

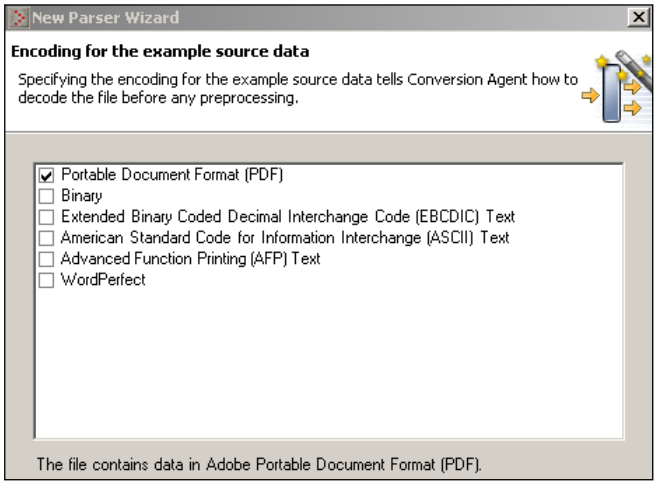


Figure 9 Specify encoding for the source document

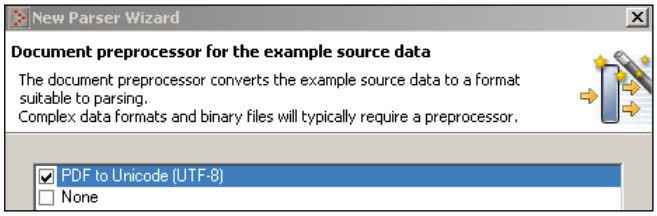


Figure 10 Specify the document preprocessor

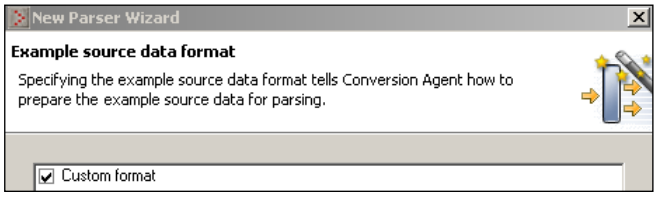


Figure 11 Specify the source data format

select the file (e.g., TestResultsDocument.pdf), and click on Next.

8. In the next screen (**Figure 9**), select the type of encoding for the source file — Portable Document Format (PDF) for the example — so that the Conversion Agent knows what type of file it is, and click on Next.

9. Now we need to select a preprocessor that will convert the PDF to a text format that the parser can work with (in this case, Unicode, which is much easier to parse than the binary PDF format). As shown in **Figure 10**, select the PDF to Unicode (UTF-8) check box and click on Next.

10. In the next screen, we specify the format of the source data that the Conversion Agent will use to interpret the document using the delimiters and processors associated with the format. For the example, we select the Custom format check box, as shown in **Figure 11**, so we can add our own delimiters and processors. Click on Next and then click on Finish to complete the wizard.

The new project PDFParserProject should now appear in the Conversion Agent Explorer, as shown in **Figure 12** (on the next page), including an automatically generated script PDFParserScript.tgp, which also appears in the script view. The text of the source document appears in the source view, and in the Component View, you can see a simple overview of the high-level objects contained in the project — the "+" sign next to Parsers indicates that it contains our newly created parser. In the Schema View, you can see namespaces and data holders, such as XML attributes and variables, defined in the target schema we specified.

Now that we have created the new parser, we need to set the input, the working area (the project's working files, such as the TGP script files), and the output of the project to use UTF-8 as the default encoding. Follow these steps:

1. Right-click on the project (PDFParserProject) in the Conversion Agent Explorer, and choose Project Properties from the context menu.

2. In the Properties for PDFParserProject window, shown in **Figure 13** (on the next page), choose Encoding in the left panel. In the Input, Working,

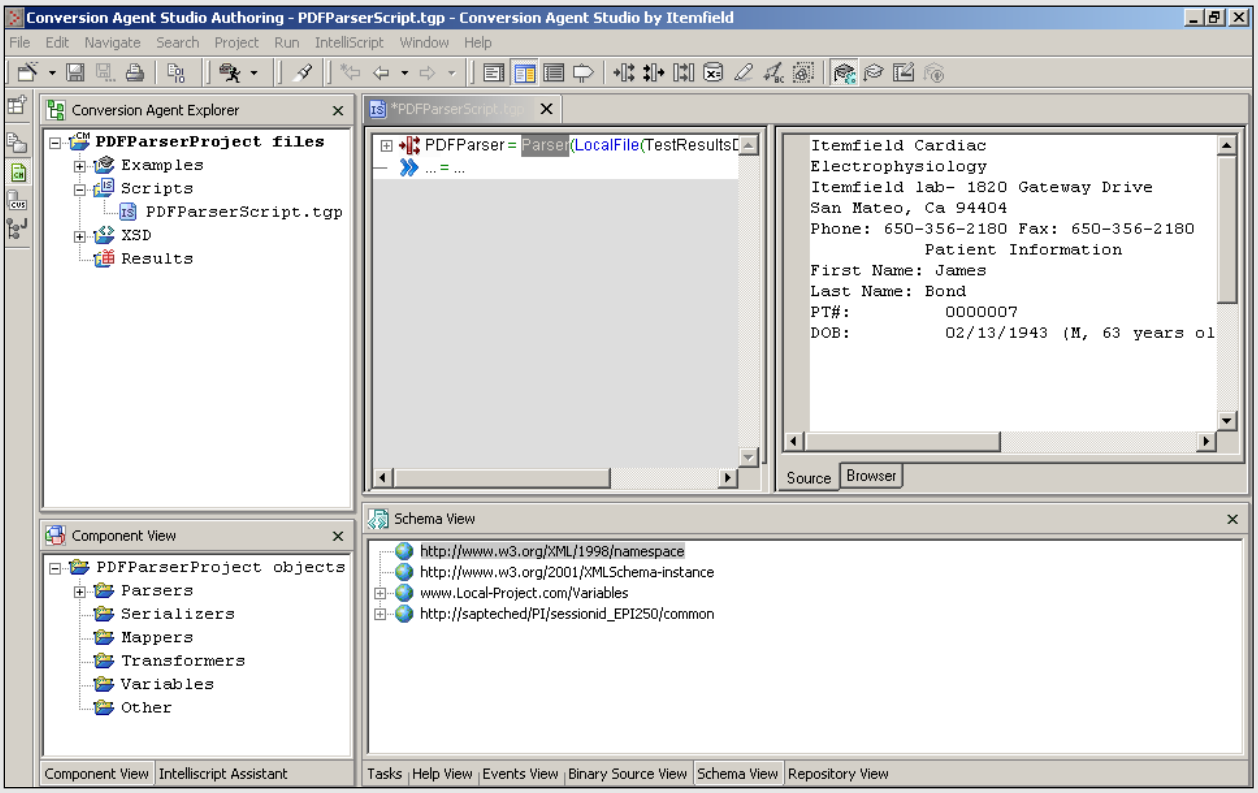


Figure 12 The newly created project

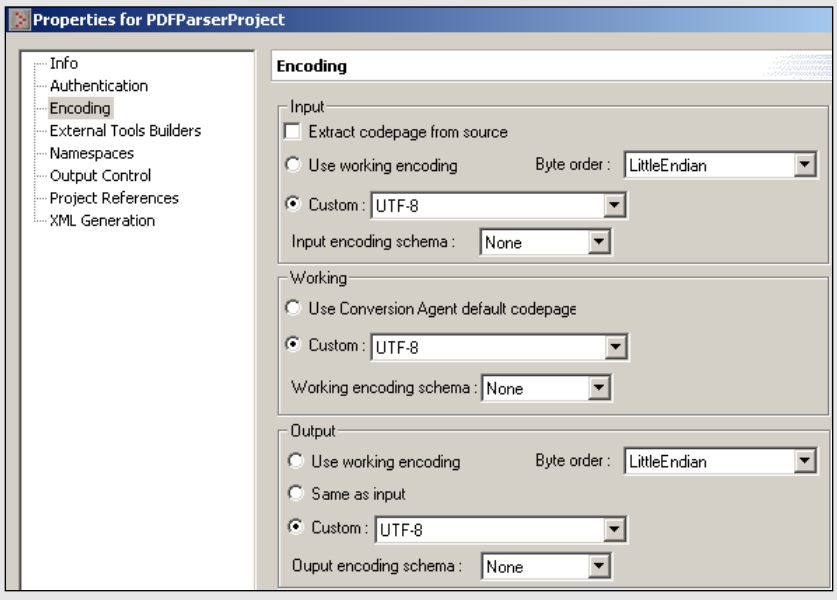


Figure 13 Parser project properties window

and Output sections in the right panel, click on the Custom radio buttons and choose UTF-8 from the drop-down field boxes.

The last thing we need to do is to add a default transformer to PDFParserScript to automatically remove white space from all of the field mappings in the transformation:

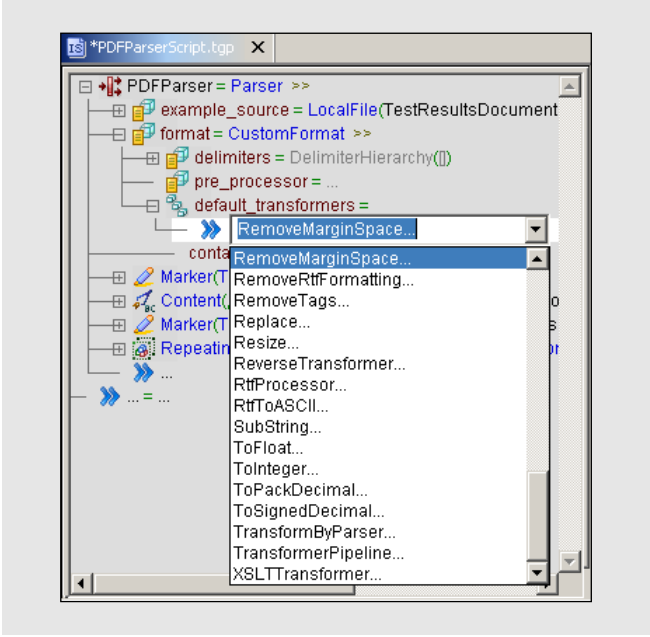


Figure 14 Adding a transformer to PDFParserScript

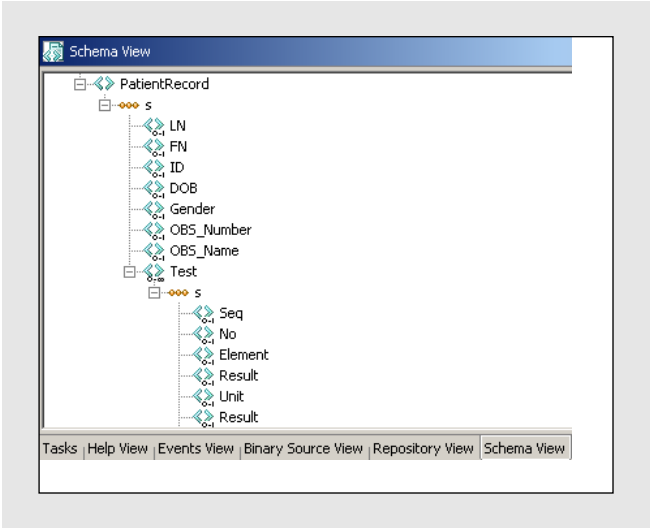


Figure 15 Schema View window expanded

1. In the script view of the Conversion Agent Studio, expand the parser by clicking on the "+" sign to the left of PDFParser, and then expand `format = CustomFormat` and `default_transformers =`.

2. Add a new default transformer by double-clicking on the ellipsis ("...") next to the two right-pointing blue arrows (») under `default_transformers =` in the script view. From the drop-down box that appears, select the transformer RemoveMarginSpace, as shown in **Figure 14**.

Now that we've created the basics for the parser, we next need to identify the header information we want to extract from the source file and map it to the XSD schema structure we want it to follow when it is converted.

Identify the header data to extract

The next step is for us to create "marker" anchors for the header data (the patient information shown in the upper part of **Figure 5**), which will identify the location of the fields we want to extract. Follow these steps:

1. First, open up the XSD schema we defined for the project — that is, the schema that defines how we want the parser to structure the resulting XML. In the Schema View in the lower right pane of the Conversion Agent Studio, expand the schema structure by clicking on the "+" signs next to the structure elements, as shown in **Figure 15**.

2. Next, we need to mark the source text we want to extract and map it to the schema. In the source view (**Figure 16** on the next page), highlight the patient's first name ("James" in the example). This text is your "content text" (i.e., the values that we want to extract). Click and drag the highlighted text to the FN field in the Schema View to map it to the schema, which will create the corresponding placeholder for the data in the parser script. After doing this, the corresponding area in the source view should look like this:

   ```
   First Name: James
   ```

3. Highlight the text "First Name:" (including the colon). This text is your "marker text" (i.e., the

field containing the values we want to extract). Right-click and choose Insert Marker. This will serve as an anchor to tell the newly created parser where to find the text to extract. The corresponding area in the source view should now look like this:

```
First Name: James
```

4. Save your parser, and in the script view, right-click on the parser name (PDFParser) and select it as the start-up component, which tells the Conversion Agent that the parser should be activated when you run the project.

5. Then, from the Conversion Agent Studio menu at the top of the screen, select Run → Run:

6. In the Conversion Agent Explorer pane, drill down in the Results tree, and double-click on the Output.xml file:

A browser will display your first mapping. If this mapping is correct, you should see something like the coding shown in **Figure 17**. As you can see, the First Name: James text in the PDF file has

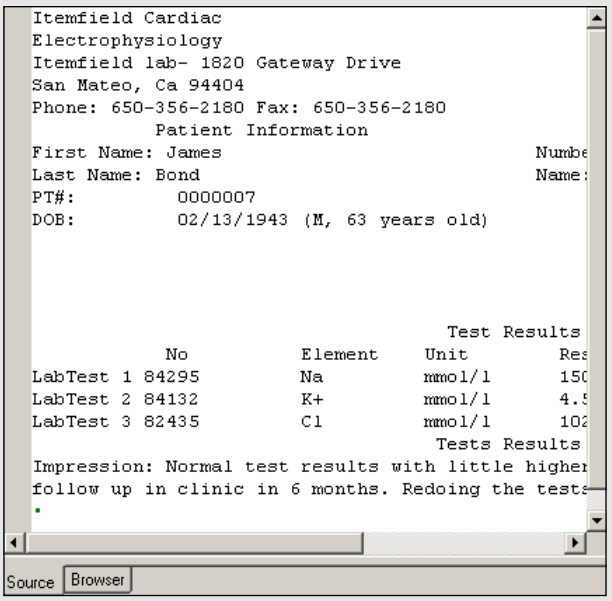


Figure 16 Document source view

```
<?xml version="1.0" encoding="windows-1252" ?>
- <PatientRecord xmlns="http://sapteched/PI/sessionid_EPI250/common">
    <FN>James</FN>
  </PatientRecord>
```

Figure 17 Header mapping for first name

been converted into an XML tag field `<FN>James</FN>` for the example.

7. Map the remaining header data shown in **Figure 18** by completing steps 1 through 3 for each field.
8. Save and run your parser. Your results should now look something like the code shown in **Figure 19**.

Now that we've identified the header data to extract, we need to identify the data in the body of the PDF file to extract.

Identify the body data to extract

The final step in creating the parser is to extract the body data (the details from the Test Results table in the lower part of **Figure 5**). For this step, we again create markers and extract the content details within the markers, but this time we will do it for a body of data, rather than a single instance of data, using a "repeating group" anchor. Follow these steps:

1. We need to indicate where the repeating group begins. Scroll down in the source view (**Figure 16**) and find the line under Test Results that reads:

```
No    Element    Unit    Result    Range
```

   Highlight the entire line, right-click on it, and choose Insert Marker on the context menu. This will serve as an anchor to tell our newly created parser where to start looking for the repeating group.

2. Highlight the text LabTest in the first line of the test results, right-click on it, and choose InsertRepeatingGroup on the context menu.
3. To map the fields of the table, we use an offset content anchor, which inserts a content anchor that enables you to extract text at specific positions.

Content text	Schema target field	Marker text
Janes	FN	First Name:
Bond	LN	Last Name:
0000007	ID	PT#
02/13/1943	DOB	DOB:
M	Gender	(
80004	OBS_Number	Number:
Electrolytes	OBS_Name	Name:

Figure 18 Header fields to map

```
<?xml version="1.0" encoding="windows-1252" ?>
- <PatientRecord xmlns="http://sapteched/PI/sessionid_EPI250/common">
    <LN>Bond</LN>
    <FN>James</FN>
    <ID>0000007</ID>
    <DOB>02/13/1943</DOB>
    <Gender>M</Gender>
    <OBS_Number>80004</OBS_Number>
    <OBS_Name>Electrolytes</OBS_Name>
  </PatientRecord>
```

Figure 19 Mapping of all header fields

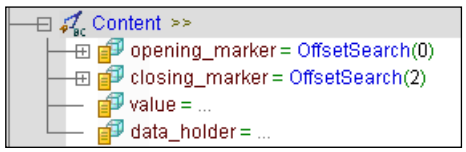
Note!

A repeating group anchor parses a repetitive area of a source document (e.g., multiple rows in a table). The repeating units (called *iterations*) are typically delimited by a separator. The repeating group contains a sequence of nested anchors and actions that parse each iteration. The repeating group anchor treats all iterations in the same way.

In our example PDF document, we have multiple rows of lab tests. So, instead of parsing each and every row, we parse only the first row using the repeating group anchor. This method takes care of parsing all remaining rows.

Highlight the text in the first row from the end of LabTest up to and including the first 1. Right-click and choose Insert Offset Content on the context menu, as shown in **Figure 20**.

4. Check the script view in the Conversion Agent Studio to inspect the resulting content anchors. The script has discerned that we are grabbing the content starting immediately after LabTest — `OffsetSearch(0)` — and continuing for two characters — `OffsetSearch(2)` — as shown under Content:



This means that the script will grab the "1" and the white space before it, but the white space will get stripped out because of the default transformer we attached to the parser earlier.

5. We next need to map this content to the target schema. Although we could use the same drag-and-drop technique we used previously, let's try another method. Double-click on the ellipsis ("...")

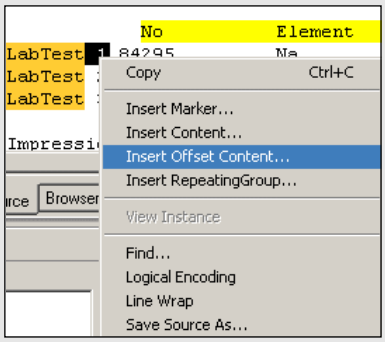


Figure 20 Inserting an offset content anchor

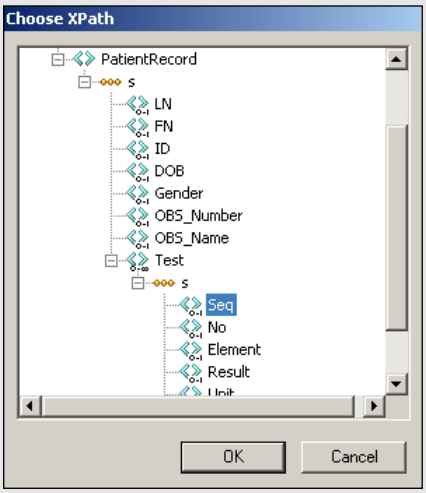


Figure 21 Choose XPath window to assign field to target schema

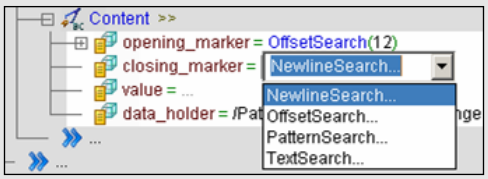
after `data_holder =`, which displays a window containing the target schema (see **Figure 21**), so that we can assign the content to the appropriate schema field — in this case, the Seq field.

6. Repeat steps 3 through 5 to perform the offset mappings for the remaining table fields for LabTest 1 under No, Element, Unit, Result, and Range. We only need to do this for the first row in the table. Because we defined a repeating group, the script will obtain the corresponding fields for the additional rows of the table automatically.

Note!

A problem exists with the last field (Range) with the way we just mapped it, because the data in this field will be different lengths depending on the medical test, and the mapping we used is a fixed offset length. With the Range field, we want to obtain all of the text to the end of the line (i.e., everything up until the new line in the document).

To correct this problem, go to the content anchor for the Range field in the script view. You will see two markers in this anchor, an opening and a closing marker, both of type OffsetSearch. Double-click on OffsetSearch for the closing marker and choose NewlineSearch on the context menu, as shown in the following screenshot:



This setting allows the script to parse the row with a fixed offset length.

```
Itemfield Cardiac
Electrophysiology
Itemfield lab- 1820 Gateway Drive
San Mateo, Ca 94404
Phone: 650-356-2180 Fax: 650-356-2180
          Patient Information                    Observation
First Name: James                          Number: 80004
Last Name: Bond                            Name: Electrolytes
PT#:          0000007
DOB:          02/13/1943 (M, 63 years old)



                                  Test Results
            No          Element     Unit        Result          Range
LabTest 1 84295         Na          mmol/l      150             136-148
LabTest 2 84132         K+          mmol/l      4.5             3.5-5
LabTest 3 82435         Cl          mmol/l      102             94-105
                                  Tests Results
Impression: Normal test results with little higher than normal Na recommended
follow up in clinic in 6 months. Redoing the tests in 3 month
```

Figure 22 Final source view

7. Save your parser, and click on the Mark Entire Example icon ( ) on the toolbar. You should see all of the fields in all of the rows of the table marked, as shown in **Figure 22**.

```
<?xml version="1.0" encoding="windows-1252" ?>
- <PatientRecord xmlns="http://sapteched/PI/sessionid_EPI250/common">
    <LN>Bond</LN>
    <FN>James</FN>
    <ID>0000007</ID>
    <DOB>02/13/1943</DOB>
    <Gender>M</Gender>
    <OBS_Number>80004</OBS_Number>
    <OBS_Name>Electrolytes</OBS_Name>
  - <Test>
      <Seq>1</Seq>
      <No>84295</No>
      <Element>Na</Element>
      <Result>150</Result>
      <Unit>mmol/l</Unit>
      <Range>136-148</Range>
    </Test>
  - <Test>
      <Seq>2</Seq>
      <No>84132</No>
      <Element>K+</Element>
      <Result>4.5</Result>
      <Unit>mmol/l</Unit>
      <Range>3.5-5</Range>
    </Test>
  - <Test>
      <Seq>3</Seq>
      <No>82435</No>
      <Element>Cl</Element>
      <Result>102</Result>
      <Unit>mmol/l</Unit>
      <Range>94-105</Range>
    </Test>
  </PatientRecord>
```

Figure 23 Final output

8. Run your parser one more time to verify that you get the results shown in **Figure 23**.

This completes the steps to create the parser. Now we can move on to creating the HL7 library object, which is a serializer that will convert the XML into an HL7 flat file format for use in a back-end HIS, after which we will:

- Deploy the parser and HL7 library projects to the Conversion Agent Engine on the PI Integration Server.
- Configure the sender and receiver communication channels to execute the end-to-end business process.

Let's start with creating the HL7 library project.

Step 2: Create the HL7 library project

In this step, we implement a library project to "serialize" an XML document into the HL7 format. Follow these steps:

1. From the File menu, choose New → Project to open the new project wizard.
2. In the next screen, choose Conversion Agent → Library Project and click on Next.

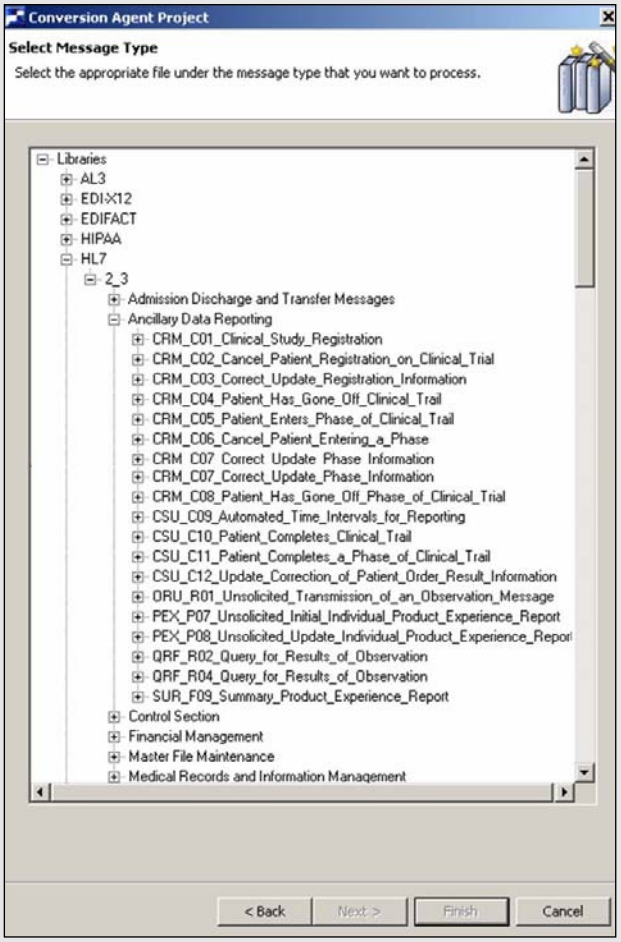


Figure 24 Conversion Agent library project

3. In the Conversion Agent Library Project screen, name your project HL7SerializerProject and specify a storage location. Click on Next.
4. In the next wizard screen shown in **Figure 24**, we need to select a predefined message type (i.e., a data structure) for the conversion. For this particular example, select HL7 → 2_3 → Ancillary Data Reporting → ORU_R01_Unsolicited_Transmission_of_an_Observation_Message, and choose ORU_R01_Unsolicited_Transmission_of_an_Observation_Message_Serializers. Click on Finish, at which time the Conversion Agent imports the predefined script and XSD schema for the serializer.
5. Navigate to the serializer script in the Conversion Agent Explorer pane, and double-click to open it in the script view. The result is shown in **Figure 25**.
6. Test the serializer by going to the Run menu and choosing Run ORU_R01_Unsolicited_Transmission_of _an_Observation_Message_Serializers. When prompted, navigate to the location you defined when you created the project and select the HL7Sample.xml file.[8]
7. After the serializer finishes running, navigate to the results in the Conversion Agent Explorer pane,

[8] The HL7Sample.xml file is based on the HL7 XSD schema and is generated by the associated mapping program in the Integration Repository. This mapping program is created as past of the overall PI configuration, which is beyond the scope of this article.

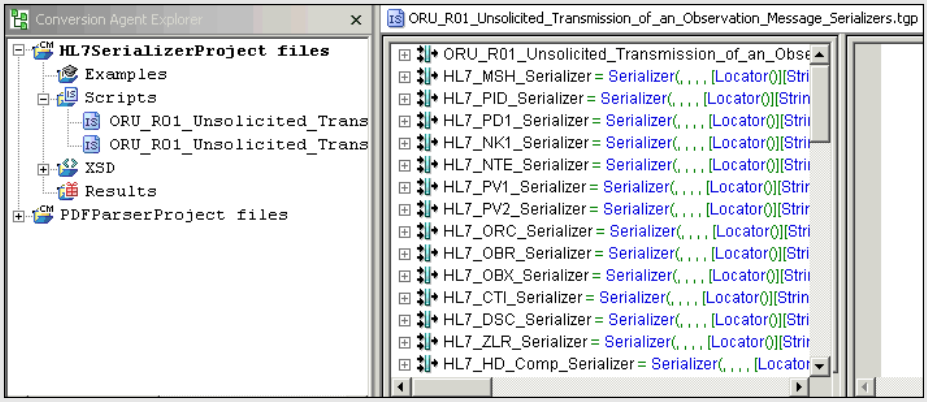


Figure 25 Serializer script results

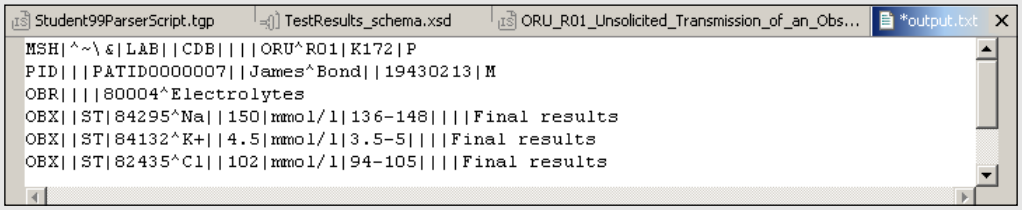


Figure 26 Serializer output file

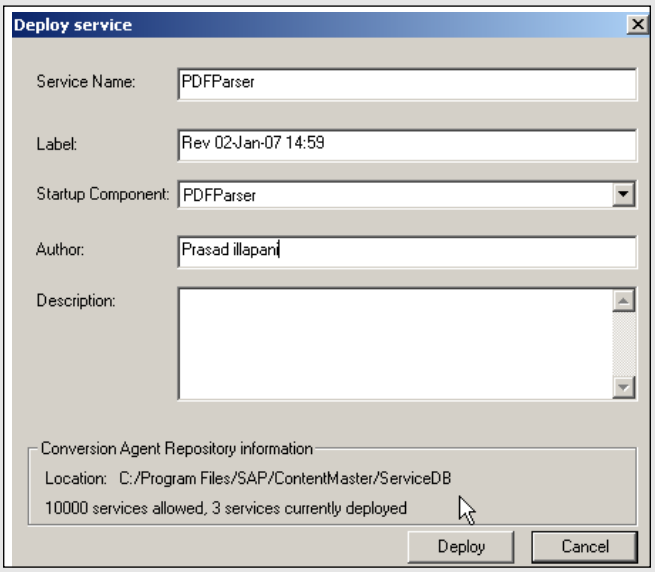


Figure 27 Deploy the parser project to the local ServiceDB

and double-click on the file Output.txt to open it. The file should look something like the HL7 output displayed in **Figure 26**.

Now that we have created both the parser project and the HL7 library serializer project, the next step is to deploy the project files so they can be accessed by the Adapter Engine on the PI Integration Server.

Step 3: Deploy the parser and HL7 library projects

We deploy the parser project and the serialized HL7 library projects in two steps:

- First, we deploy the projects as services from the Conversion Agent Studio to the ServiceDB directory on the local drive.

- Next, we copy these files from the local drive to the ServiceDB directory of the Conversion Agent Engine on the PI Integration Server.

Deploy the projects to the local ServiceDB directory

Follow these steps to deploy the projects to the local ServiceDB directory:

1. In the Conversion Agent Studio, highlight your project PDFParserProject, and choose Deploy from the Project menu.
2. As shown in **Figure 27**, enter a service name for the parser project (the default name is the name of the project), specify a label for identifying the service deployment, select the component that should run when the service is started (the parser we created), and enter an author name.
3. Click on Deploy. You should see a pop-up window that says "Service successfully deployed."

> ***Tip!***
>
> If you do not receive a success message, check the file system to see whether or not the parser was deployed. If it did not deploy, go back and check the parser for any errors.

4. Highlight the project HL7SerializerProject and repeat the deployment procedure for this project.

Now both of your projects should be published to the local ServiceDB directory.

Copy the published projects to the ServiceDB directory of the Conversion Agent Engine on the PI Integration Server

In Windows Explorer, navigate to the folder C:\Program Files\SAP\ConversionAgent\ServiceDB in your local directory. You should see the following two sub-folders:

- PDFParserProject
- HL7SerializerProject

Copy both folders to the ServiceDB directory on the PI Integration Server.

We have just completed the development of the parser and library projects using the Conversion Agent Studio, and tested and deployed the projects to the Conversion Agent Engine running on the PI Integration Server. To test the business scenario end-to-end using the Integration Server, we have to configure sender and receiver communication channels in the PI Integration Server to convert the source document from PDF to XML to HL7, and to then communicate with the Integration Server, which is explained in the next section.

Step 4: Configure the sender and receiver communication channels

In the following sections, we configure a sender communication channel, which sends a message containing the PDF data in XML format to the Integration Server, and we configure a receiver communication channel, which receives the message containing the final HL7 data in XML format from the Integration Server.

> ***Note!***
>
> Remember that this article assumes that most of the necessary PI configuration work has already been done in the Integration Repository of the PI Integration Server. At this point, the only remaining PI configuration work is to create and activate the sender and receiver communication channels based on our parser and serializer, which I briefly walk you through in the next sections.

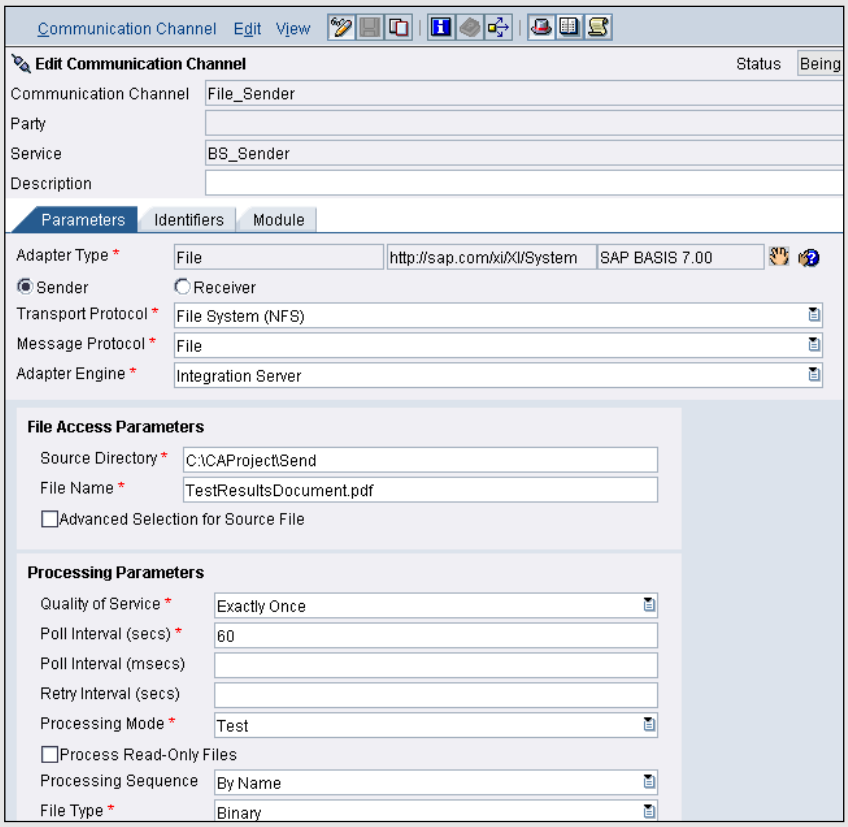


Figure 28 Sender communication channel parameters

Configure and activate the sender communication channel

Log in to the PI Integration Directory component to configure a communication channel — in this case, a sender communication channel that uses a file adapter to access the converted PDF document via the Conversion Agent Process Module in the Conversion Agent Engine, and to send the resulting XML document as a message to the Adapter Engine. Follow these steps:

1. To set the basic connectivity details, go to the Parameters tab. Name the channel and the service, select the adapter type, select the Sender radio button to define it as a sender communication channel, and enter the parameters shown in **Figure 28**. (The details of the file adapter settings are beyond the scope of this article. See http://help.sap.com and the previously mentioned *SAP Professional Journal* articles for more information.)
2. Choose the Module tab (see **Figure 29**). In the Processing Sequence section, click on the  button to add a new line, and enter the following module parameters:
   - For Module Name, enter localejbs/sap.com/com.sap.nw.cm.xi/CMTransformBean, which is the default name from the Conversion Agent Enterprise Java Bean for transformations.
   - For Type, use the drop-down to select Local Enterprise Bean.
3. In the Module Configuration section, click on the

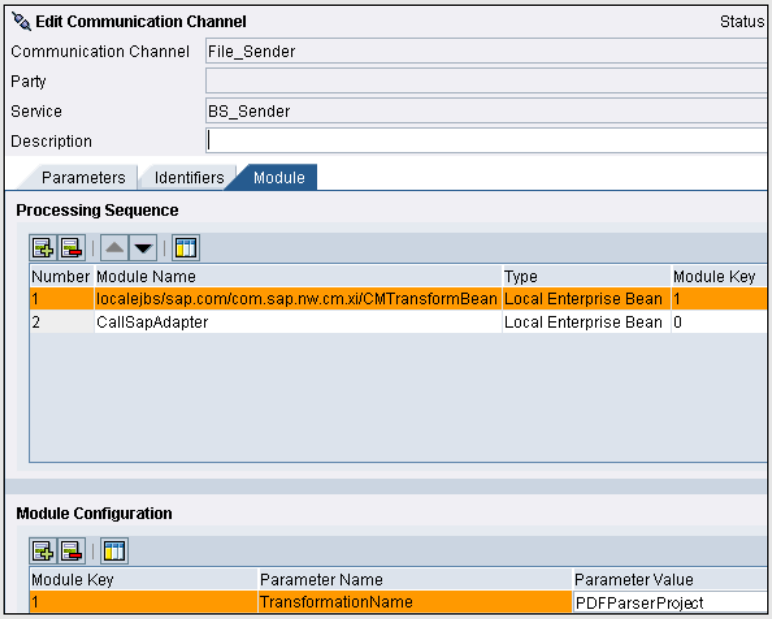


Figure 29 Configure the module for the sender communication channel

> ***Note!***
>
> The text is case-sensitive and must be entered exactly as shown in **Figure 29**.

button to add a new line. Use the drop-down in the Module Key field to select the module key that corresponds to the CMTransformBean (for the example, 1). Add the following parameters:

- For the Parameter Name, enter TransformationName (this is case-sensitive).
- For the Parameter Value, enter the name of the defined data transformation (in this case, the deployed parser project PDFParserProject).

4. To complete the configuration of the sender communication channel, we need to activate it. In the Integration Directory, go to the Change Lists tab. Expand the Standard Change list, right-click on the object (e.g., File_Sender), and then select

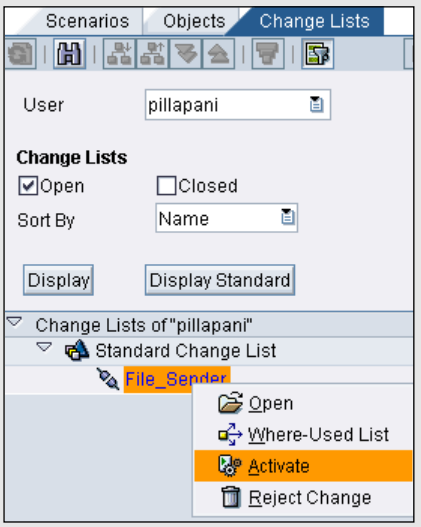


Figure 30 Activate the sender communication channel

Activate on the context menu, as shown in **Figure 30**.

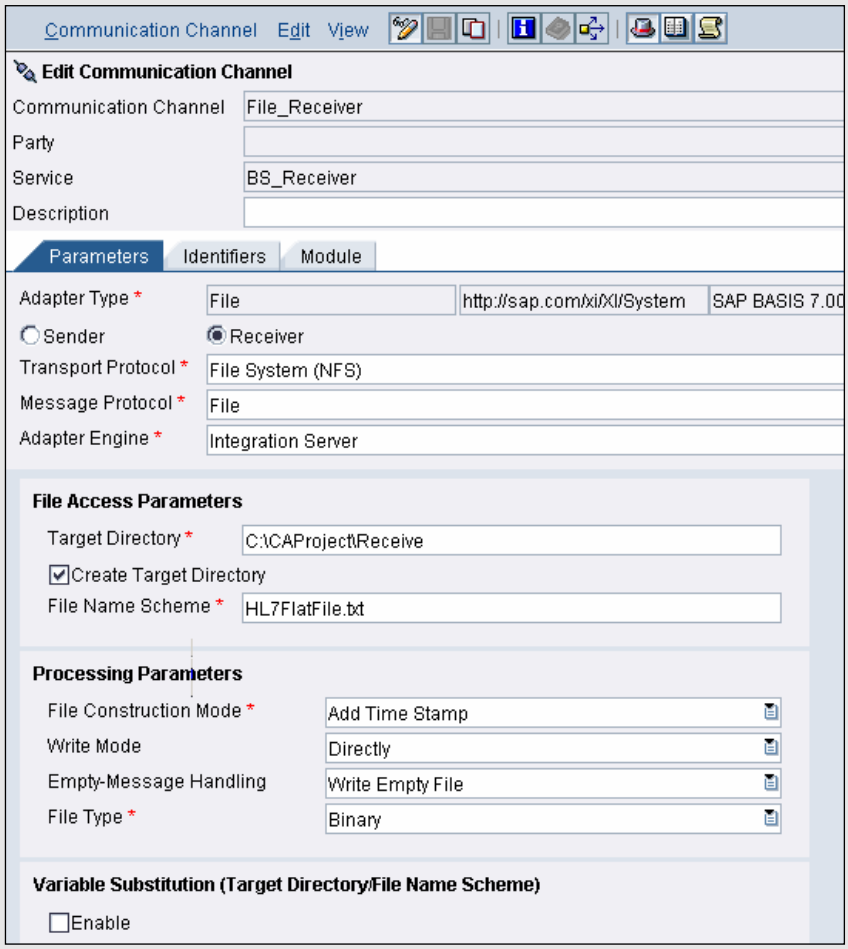


Figure 31 Receiver communication channel parameters

Configure and activate the receiver communication channel

Next, we need to create a receiver communication channel that uses a file adapter to receive the message containing the XML results and, by accessing the Conversion Agent Process Module in the Conversion Agent Engine, transform the XML results into the HL7 format. Create the receiver communication channel in the same manner as we created the sender communication channel:

1. **Figure 31** shows the settings for the Parameters tab. Note that once you select the Receiver radio button to define it as a receiver communication channel, the remaining settings on the tab will accordingly become receiver-specific.

2. Choose the Module tab (see **Figure 32**). The only setting that is different on this tab is the Parameter Value field in the Module Configuration section, which is the name of

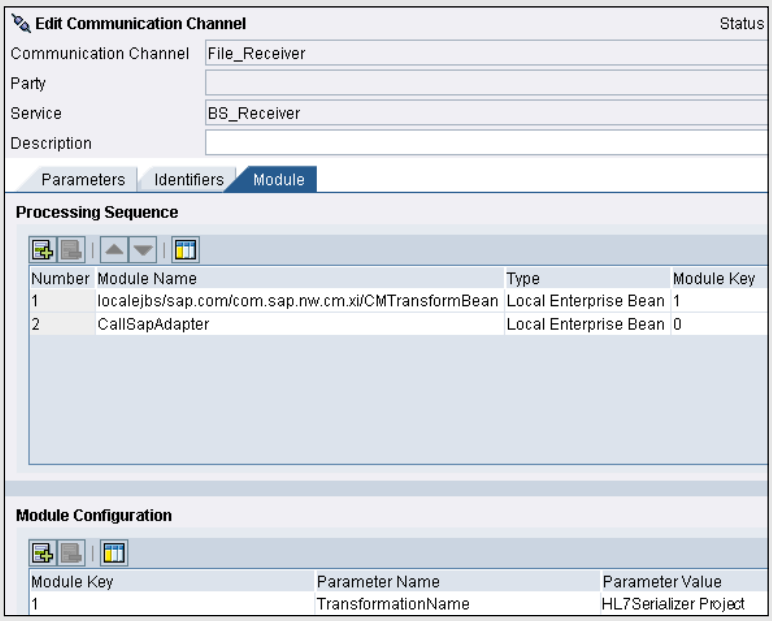


Figure 32 Configure the module for the receiver communication channel

the deployed HL7 project (HL7SerializerProject) instead of the parser project.

3. Activate the receiver communication channel in the Change Lists tab of the Integration Directory. Expand the Standard Change list, right-click on the object (e.g., File_Receiver), and select Activate on the context menu.

So far we've:

- Installed the Conversion Agent Engine and Conversion Agent Studio.
- Created a PDF parser project and an HL7 library project and deployed them in the Conversion Agent Engine.
- Added the configuration parameters for both the sender and receiver communication channels in the Integration Directory to execute the transformation beans.
- Activated the communication channel objects.

Now that the parser project and the library project are completely set up and integrated into the business scenario, we need to make sure the whole thing works.

Step 5: Test the business scenario

To test our scenario, go to the folder based on the sender communication channel configuration. This folder is on the PI Integration Server where the file adapter we just configured for it is running. Follow these steps:

1. Right-click on the sender file (in the example, TestResultsDocument.pdf), and choose Properties from the context menu. See **Figure 33** on the next page.
2. Clear the Read-only check box, and click on OK.

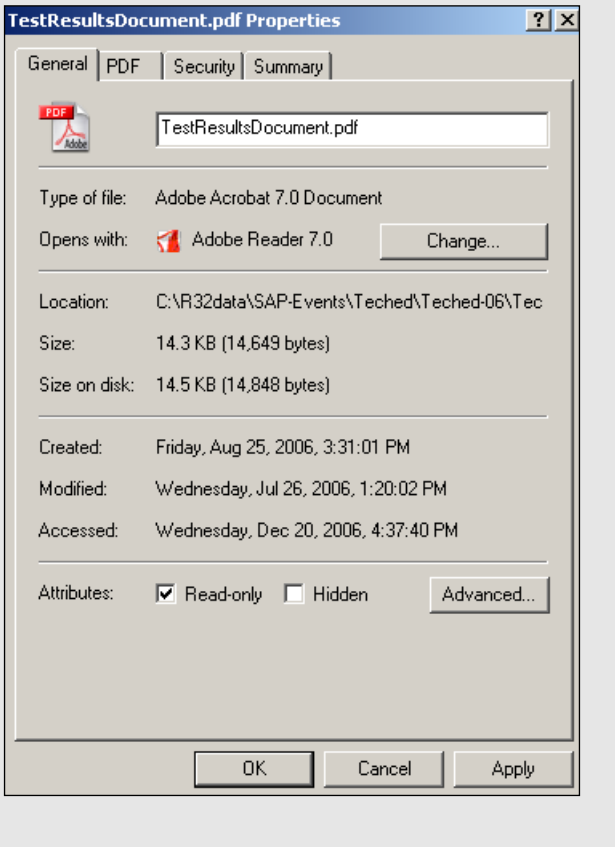


Figure 33 Adjust properties of the PDF file to allow the conversion

3. After the file has been processed,[9] navigate to the target directory (the Receive folder) defined in the receiver communication channel (see **Figure 31**). In the Receive folder, you should find a file named HL7FlatFile.txt. If this file exists, then your scenario executed successfully.

To further identify what occurred during the conversion, you can also monitor and view your messages in the Integration Server, which we'll look at next.

[9] The sender communication channel (**Figure 28**) has been configured to process only files that are not read-only and to set the Read-only flag when the file is processed. The sender communication channel is also configured to poll every 60 seconds; thus, it will take up to 60 seconds to process the file.

Tip!

If you don't find HL7FlatFile.txt, you can:

- Monitor the message in the PI Runtime Workbench tool under the Message Monitoring link. Look for the Message Status and Payload Data after the mapping step has executed.
- Select the adapter engine component for "Messages from Component." Monitor the message at the adapter engine level.

Step 6: Monitor Conversion Agent messages in the Integration Server

You can monitor the message flow from the sender to the receiver in the Integration Server (also referred to as the Integration Engine) and troubleshoot any errors that may have occurred during message processing.

1. Log in to your SAPGUI session, and navigate to transaction SXI_Monitor (or from the user menu, choose Exchange Infrastructure → Monitoring → Integration Engine – Monitoring).
2. In the screen shown in **Figure 34**, double-click on Monitor for Processed XML Messages.
3. In the Sender area of the Selection screen, use the selection help in the Service field to select your sender service BS_SENDER.
4. Press F8 or click on the check mark icon to retrieve the list of processed messages for your sender (there should be just one).
5. Navigate to the message display to see your message as it was processed in the pipeline. You can put the original payload in one window pane and the payload after the message mapping step in a second window pane (the output from your parser and the input to your serializer) as shown in **Figure 35**.

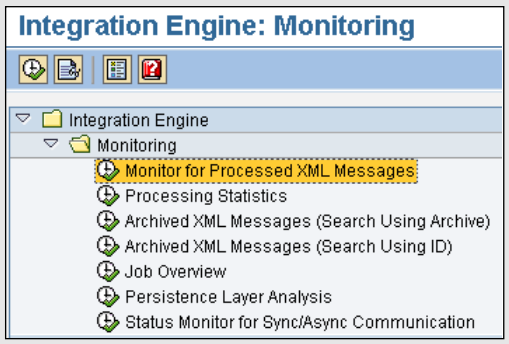


Figure 34 Monitoring messages

You can see how the PDF document is converted into an XML message based upon the parser configuration in the communication channel of the Adapter Engine.

Conclusion

The Conversion Agent tool is very simple to install and use to configure, develop, and deploy the transformations. The tool is easily integrated with the SAP Process Integration infrastructure to convert unstructured and semi-structured data formats into XML format without using any additional adapters for these conversions. No additional adapters are installed on the Integration Server, which improves the performance of the message transformation.

This tool is well suited for use in many integration business scenarios, from collecting resumes in Word documents and posting the resume data into a database or a Human Resources application, to enabling quick and easy communication between business partners by mapping content between EDI messages and SAP IDocs. Give it a try and see what it can do for you!

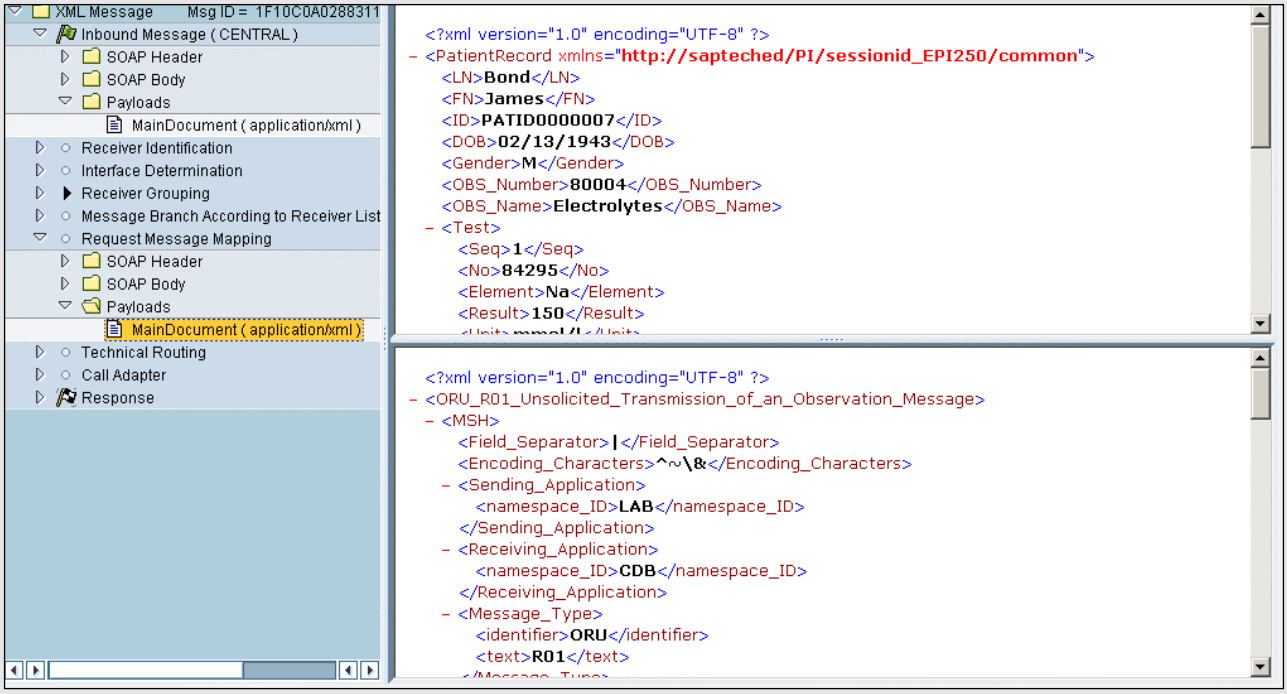


Figure 35 The message display

Acknowledgements

I would like to thank the following people for their valuable input and feedback on this article: Paul Medaille, SAP Netweaver Customer Advisor, SAP America Inc., and Ronen Schwartz, VP, Technical Sales & Services, Informatica Inc.

Prasad Illapani is a Platinum RIG Specialist with the SAP NetWeaver Integration team at SAP Labs LLC in Redmond, Washington. He focuses both on the development and systems areas of SAP Process Integration (PI). He is a regular speaker at WIS, TechEd, and SDN conferences and works closely with SAP Product Definition/Management, Development, and Field Services teams in the PI area. He also teaches PI 3.0 training classes and workshops for partners, customers, and consultants. Prasad has been with SAP Labs for almost six years. During this time, he was part of the SAP SRM team working as an RIG specialist in various SAP SRM integration projects. Prior to joining SAP, he worked as an SAP technical consultant with an SAP partner company. Prasad holds a master's degree in Computer Science and a bachelor's degree in Civil Engineering. You can reach him at prasad.illapani@sap.com.

