An Oracle White Paper
v 9.3.1.1
Part No. E18604-02
January 2012

Agile PLM Document Publishing Solution

Disclaimer

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Table of Contents

Executive Overview

During the life of a product, Agile PLM acquires, processes, and maintains a wide range of data related to the product. This data is used in many ways and for different requirements to expedite, manage, and control product development activities. Dynamic Publishing of product information enables publishing documents such as product data sheets, Parts List, or service manuals with embedded PLM data. To support this solution, Agile PLM provides two new Web services APIs for XML publishing. The Dynamic Document Publishing of product information can be used by Industrial, Retail, Life Sciences, Pharmaceutical, and High Tech industries to:

- Create new structured document templates (Product Data sheets, Parts List, Service Manual)
- Create documents in the native document publishing tool such as MS Word or Adobe Framemaker
- Browse and insert PLM metadata and file contents into documents
- Create formatted reports from PLM objects, search results, and push selected rows to reporting tools (compliance report, pricing model, quality report)
- Push a selected object ID, or all search results to a report for formatting purposes
- Modify the content that is shared by other documents already stored in PLM
- Update documents that reference content that was modified

This White Paper provides background and procedural information to install and configure the necessary components to update, format, and publish product documents using Agile PLM-based data about the given product. This includes procedures to create, and publish a sample document using the Oracle-supplied Process Extensions.

About this Whitepaper

This White Paper is a supplement to the release Readme and other Agile manuals, for example, the *Capacity Planning Guide*, *PLM Administrator Guide*, or the *SDK Developer Guide*s. The purpose of this document is to introduce Dynamic Document Publishing and is not intended as a User or Developer Guide, and will be augmented with these documents in the future.

Content and Organization

Information provided in this document is organized in the following parts:

- Dynamic Document Publishing Overview – This section describes the solution, the required environment, and applicable processes.
- Configuring the PLM Client and PLM Server – This section provides information to configure the PLM client and PLM server, and develop the Event Management process extensions (PXs) that enable the Dynamic Document Generation capability.
- Setting up BI Publisher and BI Publisher Plugins – This section provides information to install and set up the BI Publisher and ancillary tools and define templates and publish reports.
- Generating sample reports – This section provides several examples that vary the Event Trigger and objects to publish document using Agile PLM data. Information to configure Agile PLM for specific reports and to generate and store Templates is also included.

Intended Audience

The primary users of the Dynamic Document Publishing solution are document authors who will use its features to prepare and maintain documents with embedded PLM data, for example, product data sheets, parts lists, or service manuals. In this process, they are supported by Agile PLM administrators and, where applicable, SDK developers who create and manage the necessary templates and Event subscriptions that automate document updating and generation.

Cited References

The following Oracle Agile PLM and BI Publisher publications provide useful information to install and configure the Dynamic Document Publishing components and publish documents.

Oracle Agile PLM[1]

- *Agile PLM Readme*
- *Agile PLM SDK Developer Guide - Developing PLM Extensions*
- *Agile PLM AIS Developer Guide*
- *Installing Agile PLM for WebLogic Server/Installing Agile PLM for Oracle Application Server*
- *Agile PLM Administrator Guide*
- *Agile PLM Web Services User Guide*

Oracle BI Publisher[2]

- *Oracle BI Publisher 10g*

[1] These Oracle Agile PLM documents are available at Oracle Technology Network (OTN) Web site: http://www.oracle.com/technetwork/documentation/agile-085940.html

[2] Oracle BI Publisher 10g documents are available at: http://www.oracle.com/technetwork/middleware/bi-publisher/documentation/xmlpdocs-084437.html

Introduction

To support the Document Publishing Solution, Agile PLM provides two new Web services APIs to support XML publishing. These APIs return an XML package containing the object's schema and the actual data. These XML packages are used with a publishing tool such as Oracle's BI Publisher to generate any type of document based on Agile PLM metadata.

Solution Architecture

While the flexible architecture of this solution is can support other authoring tools such as Adobe Framemaker, Figure 1 summarizes the document and template formatting tasks by integrating Oracle Agile PLM and Oracle BI Publisher. BI Publisher is a reporting and document management solution. BI Publisher report formats are designed with MS Word and published in PDF, HTML, RTF, and Excel formats. The flow of data from Agile PLM, output formats, and potential destinations are summarized in the following illustration.

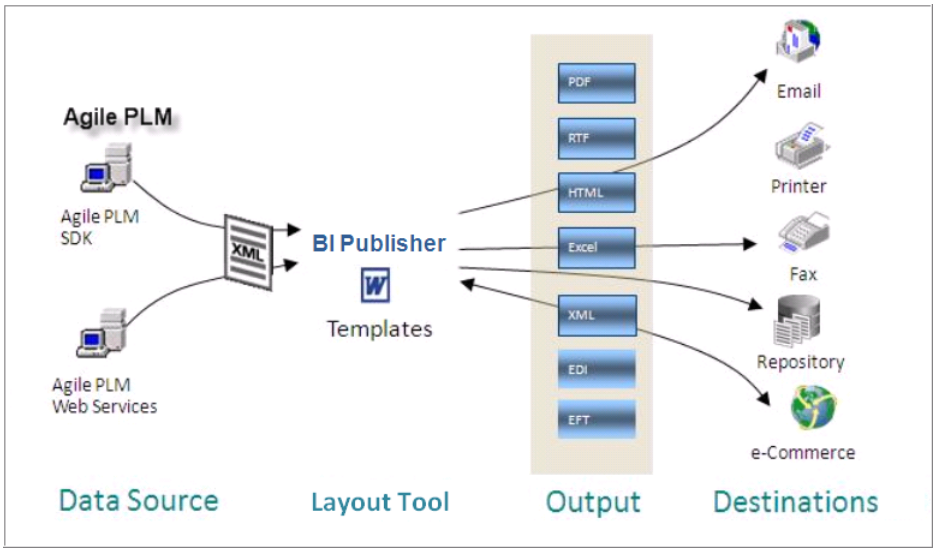

Figure 1 Document Publishing architecture

Operating Environment

Oracle's Dynamic Document Publishing is the integration of Oracle BI Publisher and Oracle Agile PLM. PLM is Oracle's product lifecycle management solution and BI Publisher is a reporting and document output management solution from Oracle. The operating environment includes:

- Oracle Agile PLM
- Oracle BI Publisher
- Microsoft Word

Oracle Agile PLM

Oracle Agile PLM components are:

- Agile PLM Release 9.3.1.1 (Server and databases)
- Agile PLM Release 9.3.1 File Manager
- Agile PLM Release 9.3.1 SDK (Template Management Java and Script PXs)
- Agile PLM Release 9.3.1 Web Services APIs – The following APIs support Dynamic Document Generation[3]:
  - **loadXMLSchema** – This Web Service API returns an XML package that fully describes the attributes of the object. This Web Service is used to create XML schema files that are used by BI Publisher to create the Templates. For example, if you use this Web Service against a subclass like Engineering Change Order (ECO), it will tell BI Publisher all of the possible attributes for ECOs. This is useful to enable using all potential attributes of an object when creating a Template.
  - **loadXMLData** – This Web Service API returns the actual data that is stored for an object in an XML package. This Web service is used to retrieve the object data that is combined with the Template to create the output file. You can also use the saved output from this Web Service to test a Template in BI Publisher

[3] For more information about these APIs, refer to Agile PLM Web Services User Guide.

Oracle BI Publisher

BI Publisher products and MS Word 2003/2007 serve as template builders. BI Publisher Enterprise is the document generation engine. BI Publisher report formats are designed using Microsoft Word and support creating reports from multiple data sources.

- BI Publisher Desktop - This is a *prerequisite* to implement Dynamic Document Publishing.
- BI Publisher Enterprise (BI Publisher v10gR3) - BI Publisher Enterprise is required if there is a need to publish from multiple sources versus a single source of data[4].

Microsoft Word 2003/2007 and BI Publisher

Microsoft (MS) Word 2003/2007 is fully integrated with BI Publisher and serves as the Document Template Authoring tool.

The Dynamic Document Publishing Process

Figure 2 is a streamlined view of the Document Publishing process. In this process, a "trigger" invokes a "handler" and that causes steps 1, 2, 3, and 4 to execute automatically.

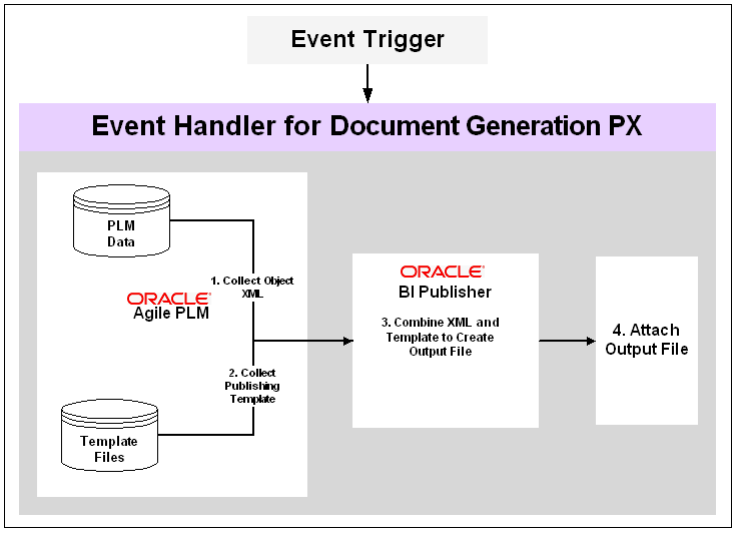

Figure 2 Dynamic Document Publishing Process and Steps

[4] A license is necessary for Oracle BI Publisher Enterprise.

To describe what actually occurs, consider the Datasheet in Figure 3. It provides information about an assembly part that is not yet a PLM object. When this object is loaded into the PLM, applicable information about this object is maintained in the object's attributes. You can see some of these attributes in The Datasheet Configuration in Agile PLM.

Because of recent business activities, there is a need to update some of these attributes. For example, the Name and Address attributes, and then publish the updated Datasheet. The next few paragraphs summarize the Dynamic Document Publishing process that updates and publishes this Datasheet.

SUN SPARC ENTERPRISE M4000 SERVER

KEY FEATURES

MAINFRAME-CLASS RELIABILITY, AVAILABILITY, AND SERVICEABILITY IN A VALUE-PRICED SERVER

- Mix and match the boards with earlier versions of the SPARC processor in a single system for continued investment protection
- Binary compatibility with earlier versions of your applications
- Scalable, mainframe-class computing for the open systems market
- Advanced virtualization technologies, methodologies, and services, making Sun SPARC Enterprise servers ideal for consolidation
- Up to four quad-core SPARC64 VII or dual-core SPARC64 VI processors
- Maximum system utilization through hardware partitioning, with up to two physical Dynamic Domains, with granularity down to a single socket
- Leading performance, utilization, and speed to implementation with Oracle's global support network and professional services for Sun products

Companies can't afford to have business-critical services go offline. To meet these increasing demands for compute services, platforms must be flexible and provide a cost-effective growth path. Oracle's midrange Sun SPARC Enterprise M4000 server boasts reliability, flexibility, and binary compatibility in a value-priced server by combining the power of Oracle's Sun Solaris Operating System with mainframe RAS features. Built on the latest and most advanced SPARC64 VII quad-core or SPARC64 VI dual-core processors, the Sun SPARC Enterprise M4000 server delivers enterprise-class service levels for essential business applications, databases, and smaller consolidation projects.

The Sun SPARC Enterprise M4000 server delivers enterprise-class service levels.

Investment Protection, Scalability, Reliability, and Flexibility

With the Sun SPARC Enterprise M4000 servers, you can protect your IT investment and scale out as needed with "in-box" upgrades. The option to mix and match different speeds/generations of SPARC64 processors in existing and new M-series servers uniquely protects investments and enables easy and low-cost upgrades not offered by IBM or HP.

Mainframe-class RAS features come standard in the Sun SPARC Enterprise M4000 server, including automatic recovery with instruction retry, up to 128 GB of system memory error-correcting code (ECC) protection with extended ECC support, guaranteed data path integrity, total SRAM and register protection, and configurable memory mirroring. In addition, the disks, power supply, and fans are redundant and hot-swappable, while the I/O cards are also hot-swappable. Many features unique to the Solaris 10 OS enhance system reliability even further, including Predictive Self-Healing, which automatically identifies and isolates faults and provides specific guidance when action is required.

Sun ORACLE

1

Figure 3 Datasheet before it is loaded into Agile PLM

The Datasheet Configuration in Agile PLM

The Datasheet attributes are shown in the following illustration. As a PLM object, anytime the Datasheet is updated, its attributes such as date, product title, and descriptions are subject to change. Dynamic Document Publishing enables publication of the Datasheet with the latest information. However, because BI Publisher generates the final document, it is necessary is to convert these attributes to a Data XML file for BI Publisher processing.

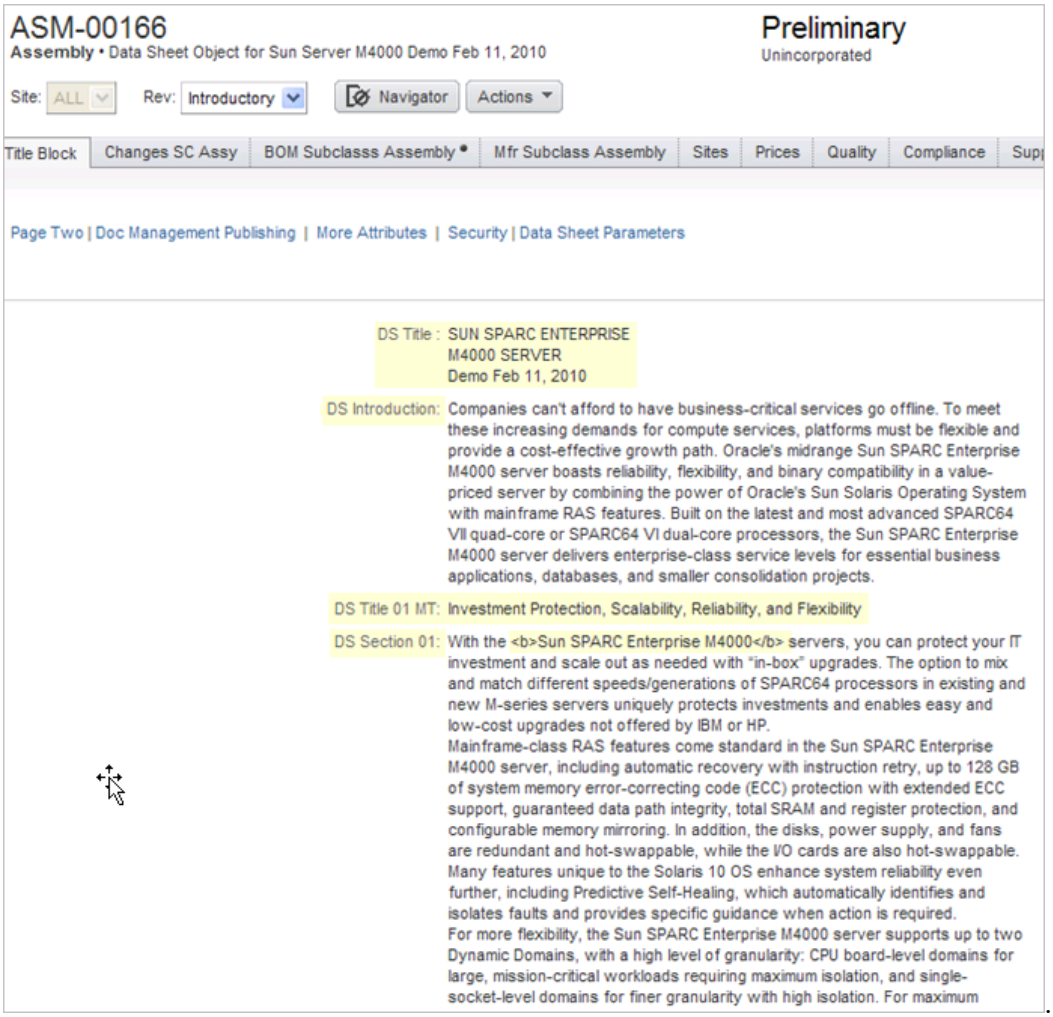

Figure 4 Data sheet attributes in Agile PLM

Generating the Data XML File

This step assumes you have installed BI Publisher Desktop and the necessary BI Publisher commands are accessible from MS Word Add-Ins. This is performed by invoking the `loadXMLData` Web Services API from MS Word and selecting **Add-Ins > Data > Load XML Data**. The next step is to combine the Data XML and Schema XML (Template) files for BI Publisher to generate the Datasheet

```
<?xml version="1.0" encoding="UTF-8"?>
<AgileData xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<Assembly>
<TitleBlock>
<number>ASM-00166</number>
<itemType>Assembly</itemType>
<lifecyclePhase>Preliminary</lifecyclePhase>
<description>Data Sheet Object for Sun Server M4000 Demo Feb 11, 2010</description>
<productLineS><Value>Automotive - Components</Value></productLineS>
<shippableItem>No</shippableItem>
<excludeFromRollup>No</excludeFromRollup></TitleBlock>
<Attachments>
<filename>AgileData_Assembly.xml</filename>
<fileDescription>Agile Data for ASM-00166</fileDescription>
<fileSize>5441</fileSize>
<fileType>xml</fileType>
<folderNumber>FOLDER0001017</folderNumber>
<folderVersion>2</folderVersion>
<modifiedDate>2010-02-13T06:25:23Z</modifiedDate>
<lastViewDate>2010-02-13T06:25:22Z</lastViewDate>
<checkinUser>Deron Johnstone</checkinUser></Attachments>
<PageThree>
<DSTitle>SUN SPARC ENTERPRISE
M4000 SERVER
Demo Feb 11, 2010</DSTitle>
<DSIntroduction>Companies can&apos;t afford to have business-critical services go offline. To meet t
<DSTitle01MT>Investment Protection, Scalability, Reliability, and Flexibility</DSTitle01MT>
<DSSection01>With the &lt;b&gt;Sun SPARC Enterprise M4000&lt;/b&gt; servers, you can protect your IT
Mainframe-class RAS features come standard in the Sun SPARC Enterprise M4000 server, including autor
For more flexibility, the Sun SPARC Enterprise M4000 server supports up to two Dynamic Domains, with
<DSTitle02MT>Solaris: The World&apos;s Most Advanced Operating System</DSTitle02MT>
<DSSection02>The foundation of the Sun SPARC Enterprise M4000 server is the Solaris 10 OS, which cor
<DSKeyFeatures>MAINFRAME-CLASS RELIABILITY, AVAILABILITY, AND SERVICEABILITY IN A VALUE-PRICED SERVE
&lt;ul&gt;&lt;li&gt;Mix and match the boards with earlier versions of the SPARC processor in a singl
&lt;li&gt;Binary compatibility with earlier versions of your applications&lt;/li&gt;
&lt;li&gt;Scalable, mainframe-class computing for the open systems market&lt;/li&gt;
&lt;li&gt;Advanced virtualization technologies, methodologies, and services, making Sun SPARC Enterp
&lt;li&gt;Up to four quad-core SPARC64 VII or dual-core SPARC64 VI processors&lt;/li&gt;
```

Figure 5 Data XML file

Creating the Template

Template is an RTF file created and formatted using Word, BI Publisher, and object's attributes in Agile PLM. For information and procedures to create Templates, see Building BI Publisher Templates and Cited References. Assuming you have created the Template file, running the `loadXMLSchema` API, again, from MS Word and selecting **Add-Ins > Data > Load XML** Data will generate the following XSD file showing the selected attributes.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="AgileData" type="AgileDataType"/>

    <xs:complexType name="AgileDataType">
        <xs:sequence>
            <xs:element name="Assembly" type="AssemblyType" minOccurs="0" maxOccurs="unbounded" nillable="true"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="AssemblyType">
        <xs:sequence>
            <xs:element name="BOM" type="AssemblyBOMType" minOccurs="0" maxOccurs="1" nillable="true"/>
            <xs:element name="TitleBlock" type="AssemblyTitleBlockType" minOccurs="0" maxOccurs="1" nillable="true"/>
            <xs:element name="Attachments" type="AssemblyAttachmentsType" minOccurs="0" maxOccurs="1" nillable="true"/>
            <xs:element name="Manufacturers" type="AssemblyManufacturersType" minOccurs="0" maxOccurs="1" nillable="tru
            <xs:element name="Specifications" type="AssemblySpecificationsType" minOccurs="0" maxOccurs="1" nillable="t
            <xs:element name="Relationships" type="AssemblyRelationshipsType" minOccurs="0" maxOccurs="1" nillable="tru
            <xs:element name="Quality" type="AssemblyQualityType" minOccurs="0" maxOccurs="1" nillable="true"/>
            <xs:element name="Sites" type="AssemblySitesType" minOccurs="0" maxOccurs="1" nillable="true"/>
            <xs:element name="Prices" type="AssemblyPricesType" minOccurs="0" maxOccurs="1" nillable="true"/>
            <xs:element name="PendingChanges" type="AssemblyPendingChangesType" minOccurs="0" maxOccurs="1" nillable="t
            <xs:element name="WhereUsed" type="AssemblyWhereUsedType" minOccurs="0" maxOccurs="1" nillable="true"/>
            <xs:element name="PendingChangeWhereUsed" type="AssemblyPendingChangeWhereUsedType" minOccurs="0" maxOccurs
            <xs:element name="Compositions" type="AssemblyCompositionsType" minOccurs="0" maxOccurs="1" nillable="true"
            <xs:element name="Substances" type="AssemblySubstancesType" minOccurs="0" maxOccurs="1" nillable="true"/>
            <xs:element name="PageTwo" type="AssemblyPageTwoType" minOccurs="0" maxOccurs="1" nillable="true"/>
            <xs:element name="ChangeHistory" type="AssemblyChangeHistoryType" minOccurs="0" maxOccurs="1" nillable="tru
            <xs:element name="Suppliers" type="AssemblySuppliersType" minOccurs="0" maxOccurs="1" nillable="true"/>
            <xs:element name="QCRs" type="AssemblyQCRsType" minOccurs="0" maxOccurs="1" nillable="true"/>
            <xs:element name="History" type="AssemblyHistoryType" minOccurs="0" maxOccurs="1" nillable="true"/>
            <xs:element name="Instances" type="AssemblyInstancesType" minOccurs="0" maxOccurs="1" nillable="true"/>
            <xs:element name="PageThree" type="AssemblyPageThreeType" minOccurs="0" maxOccurs="1" nillable="true"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="AssemblyBOMType">
        <xs:sequence>
            <xs:element name="BOMRow" type="AssemblyBOMRowType" minOccurs="0" maxOccurs="unbounded" nillable="true"/>
```

Figure 6 Datasheet Schema XML file

Testing/Viewing the Template

Now, you can view the Template and make sure it is properly formatted and the specified PLM attributes are selected. To view the output in MS Word, select **Add-Ins > Preview > <File_Format>**, where file format is RTF, PDF, EXCEL, HTML, or PowerPoint. Figure 7 shows the output in RTF format.

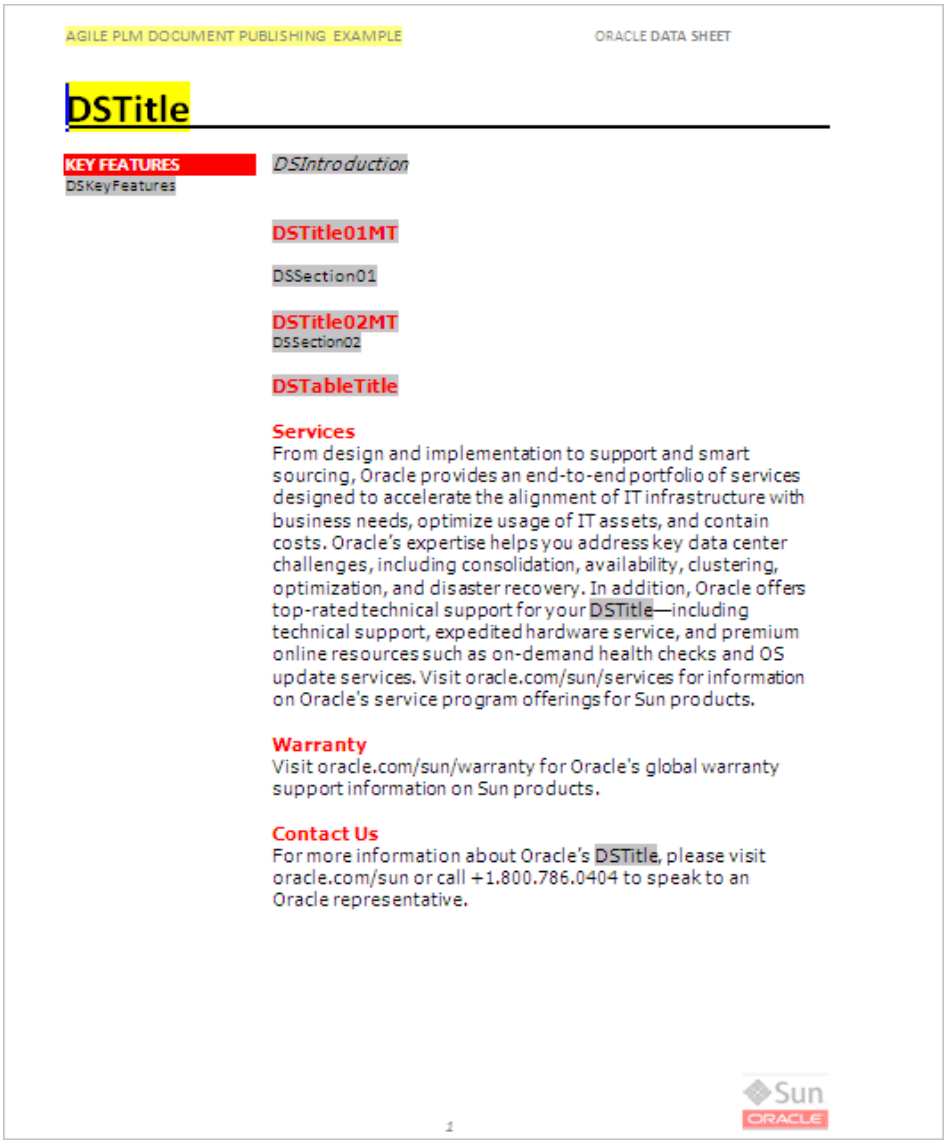

Figure 7 Template output

Combining XML Data with Template - Publishing the Document

When the appropriate Event subscription which is set up in advance is triggered, XML Data is combined with the Template and the document is generated in the specified format. In this case, in PDF format. For information to set up the various Event subscriptions, see Getting Started with Publishing the Sample.

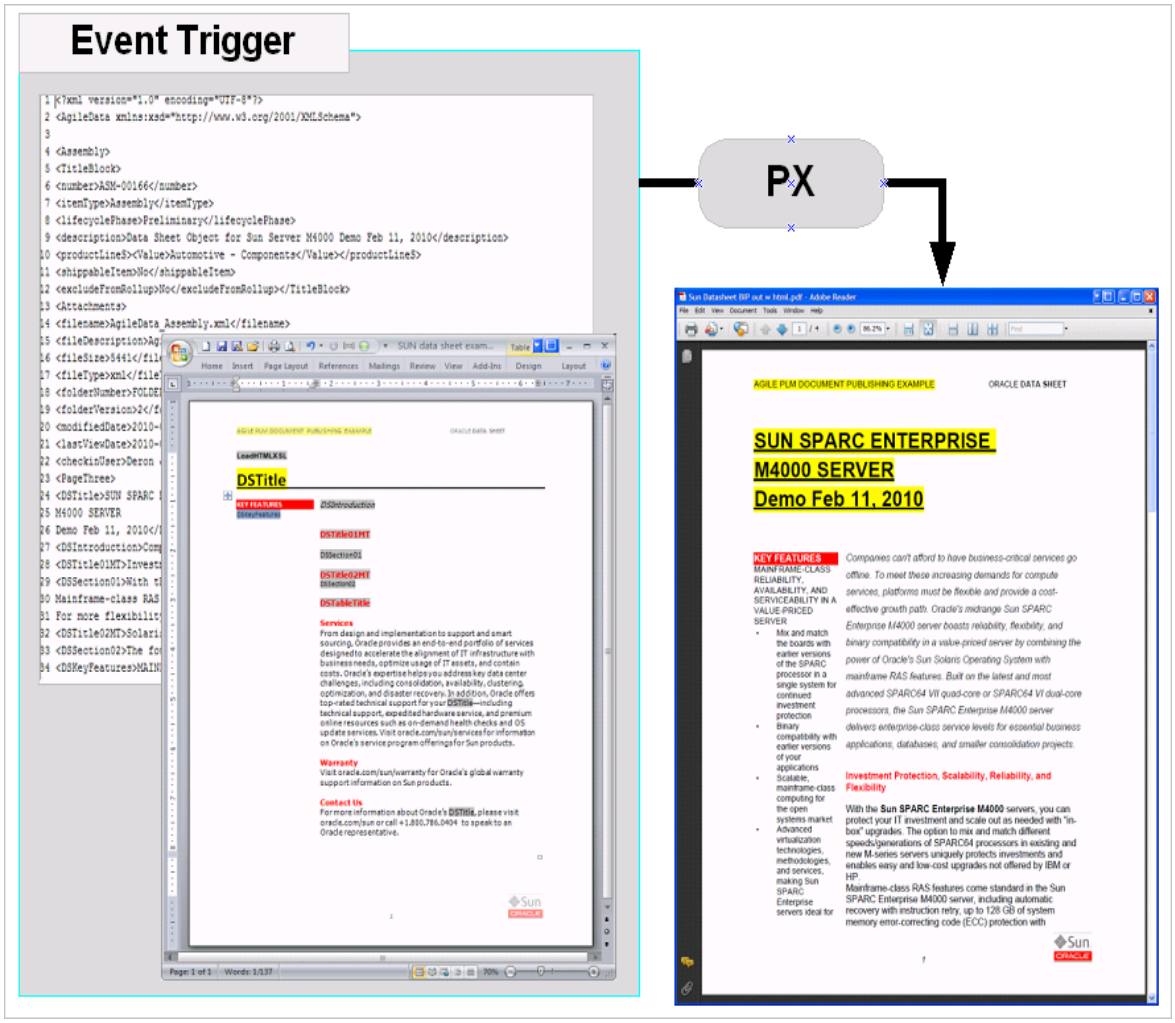

Figure 8 Combining Data XML and Template to generate the document

Setting Up the Environment for Document Publishing

Setting up the environment requires installing and the following installations and BI Publisher and Agile PLM configurations.

Installing and Configuring BI Publisher Templates

These steps include installing BI Publisher to enable:

- Inserting data fields into RTF templates
- Inserting data driven tables and crosstabs
- Inserting data driven charts
- Previewing and Validating RTF templates with sample XML data
- Browsing and updating the data in the selected fields

Configuring Agile PLM Administrator

These configurations include:

- Creating the Template Subclass
- Configuring Attributes and Agile Content Services (ACS) Filter

Configuring Agile PLM Server

Server configuration involves creating and setting up the following Process Extensions (PXs)[5]:

- Template Management Structure Creation PX
- SchemaGeneration PX
- DataGeneration PX
- DocumentGeneration PX

[5] You can use the Oracle supplied PXs described in **Error! Reference source not found.**.

Installing BI Publisher Desktop

The BI Publisher extension to Microsoft Word simplifies the development of RTF templates.

To install BI Publisher Desktop, do as follows:

- Step 1. Download and install BI Publisher Desktop 10g or 11g from OTN at[6]: http://www.oracle.com/technetwork/middleware/bi-publisher/downloads/index.html
- Step 2. Verify Add-Ins is available in MS Word.

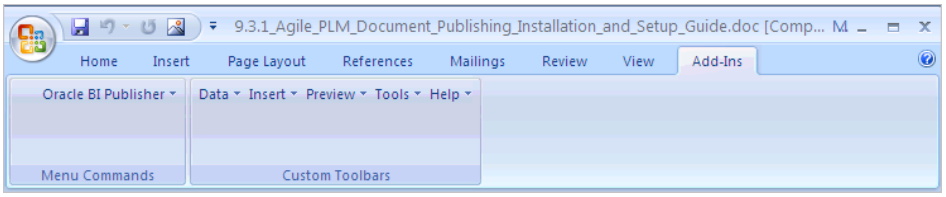

Figure 9 BI Publisher Add-In in MS Word

Performing Agile PLM Administrator Configurations

The Agile PLM Administrator configurations include:

- One time configurations
  - Adding a Subclass called DocumentTemplate
- Object-level configurations
  - Adding Page 2 fields
  - Defining ACS Filters
  - Creating Event Subscriptions consisting of Event Masks, Handler Masks, and Subscriber Masks for Script PX or Java PX[7]

[6] Installing Desktop 11g requires downloading all four installation files on the OTN.

[7] For information on Event Management framework, refer to the *Agile PLM Administrator Guide*.

Creating the Documents Subclass

for the XML schema (Templates) files and is used in Script PX and Java PX configurations. The PX that creates the object schema XML automatically creates an object of this subclass for every object in the system and attaches the schema XML to this object.

This is a onetime configuration and provides a placeholder for all Template files. To create the DocumentTemplate Subclass do as follows[8].

- Step 1. Log in to Java Client as an administrator.
- Step 2. Select **Admin > Classes > Items > Documents** to open the Class:Documents dialog.
- Step 3. In Class:Documents dialog, select the **Subclasses** tab and click **New Subclass**  to open the **New Subclass** dialog.
- Step 4. Create a new Documents Subclass called DocumentTemplate for Item as shown below.

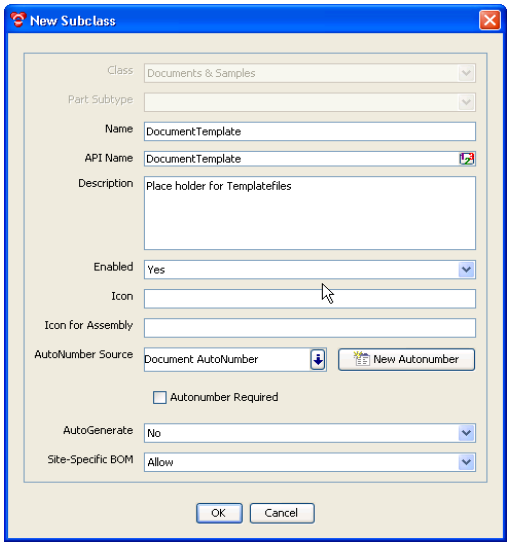

Figure 10 DocumentTemplate subclass settings

Be sure to select **Document Number** for **New Autonumber**. For details, see TemplateManagement.properties file in Template Management Process Extensions. You can find a copy in the Doc-Publishing folder described in SDK Samples Folder and Document Publishing Examples.

- Step 5. Click **OK**. The DocumentTemplate subclass opens in the General Information page.

[8] This is a onetime configuration and provides a placeholder for all Template files.

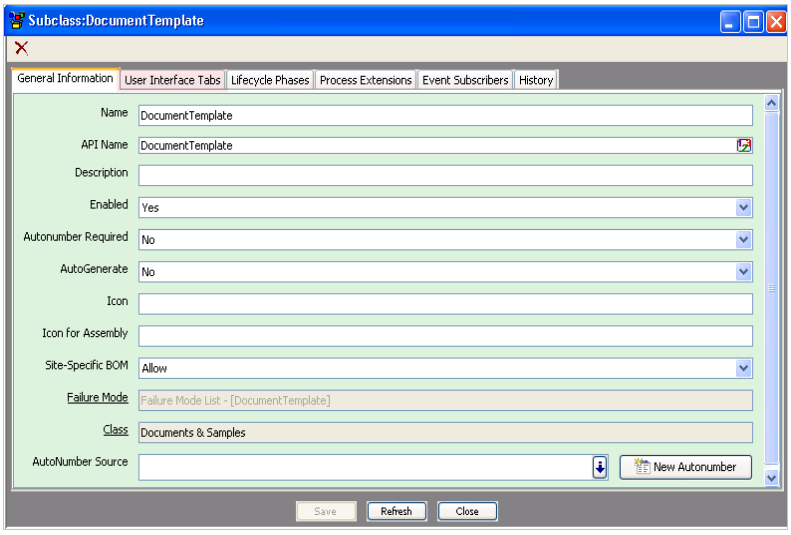

Figure 11 DocumentTemplate General Information page

- Step 6. To continue, see Defining Page 2 Fields for the Object.

Setting the Title Block Number Fields

Complete the following steps to set these fields.

- Step 1. In Agile PLM Java client select the **Admin** tab.
- Step 2. Select **Classes> Items > Documents> User Interface Tabs > TitleBlock > Attributes:Title Block > Number**. The Attributes:Number page appears.

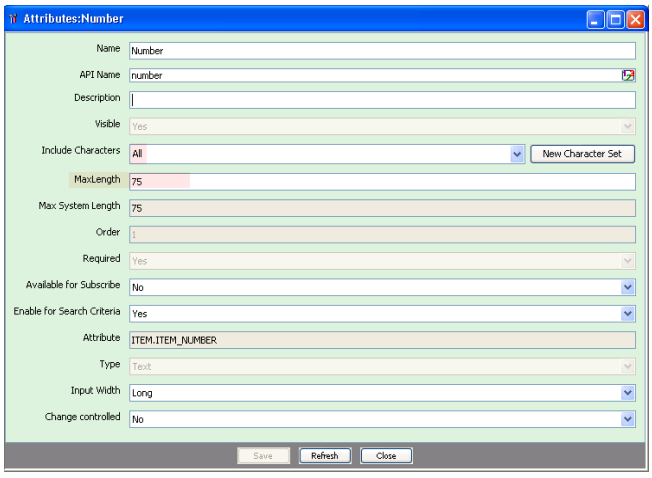

Figure 12 Page 2 Attributes for the object

- Step 3. Set the **MaxLength** field **75** and set the **Include Characters** field to **All**.
- Step 4. Click **Save** to complete this task.[9].

Configuring Agile Content Services Filters

Document publishing Web Services rely on Agile Content Services (ACS) filters to determine:

- Where the Template is stored
- The data that is returned for the object in the XML file
- How to output the document

You can tailor these filters to return the minimum information to improve performance and minimize waste during data transfer. An ACS filter is referred to by its API Name. As indicated earlier, these are object-level configurations.

Defining Page 2 Fields for the Object

The required fields for the Sample are a Heading field, two Text fields, and one List field[10]. The Base IDs are for later use.

- **Heading field** – This is for BI Publisher to display the Doc Publishing attributes in a Heading area.
- **List field** – This Alpha Type filed determines the Output Type (EXCEL, RTF, PDF, and HTML) and assumes **List11** and Base ID **1271**.
- **Text field** – This is for the Filter and assumes **Text 12** and Base ID **1302**.
- **Text field** – This is for the Template Holder and assumes **Text11** and Base ID **1301**.

Complete the following steps to define these fields.

- Step 1. Log in to Java Client and select **Admin > Classes > Documents > User Interface Tabs > Page 2 > Attributes:Page Two > List11**.

[9] You can rename this subclass if you modify the configuration of the PX. For example, if you change TEMPLATE_SUBCLASS_API_NAME=DocumentTemplate in ManagementStructure.properties file.

[10] The selected fields provide flexibility for the sample and may not be necessary in a production implementation

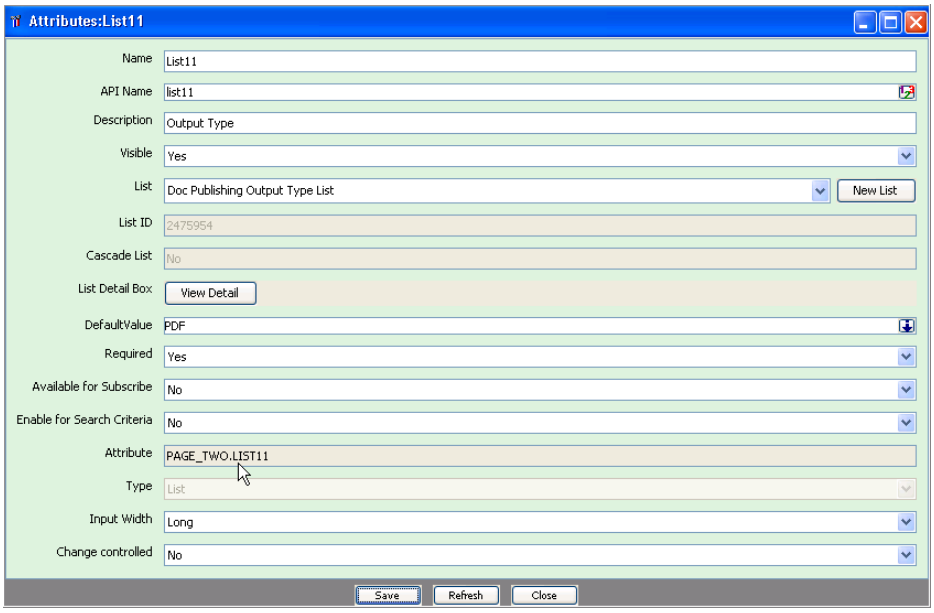

Figure 13 Text field attributes settings

- Step 2. Click **Save**.
- Step 3. In Java client, select **Admin > Data Settings > Classes > Documents Class > User Interface Tabs > Page 2**, and configure the remaining Text and Header fields as shown in the following figure.

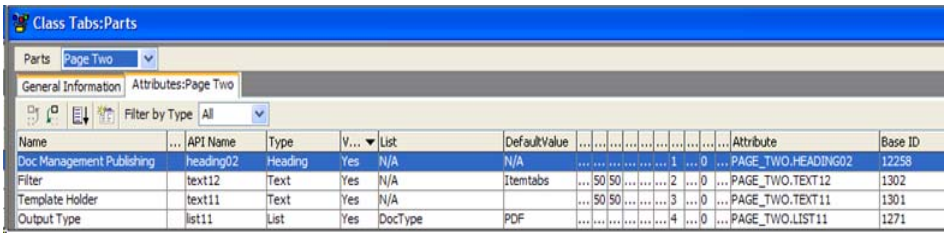

Figure 14 Object Page 2 fields for Document Publishing sample

The Base ID values are used in the Java PX Properties files and the Script PX text files. If different fields are used, you must change the Java or Script PXs to reflect the new Base ID values.

Defining Agile Content Services Filters for XML Data Files

Document Publishing PXs use ACS filters to determine how to build the XML files. Agile PLM provides a set of Agile PLM filters and you can use these filters or define your own filters. Fields selected for the filter provide flexibility for the sample and you can alter them for a production environment[11].

To create the ACS filter complete the following steps.

- Step 1. Log in to Agile PLM Java Client with administrator privileges.
- Step 2. Select **Admin > System Settings > Agile Content Service > Filters** to open the Create Filter dialog.
- Step 3. Create a new filter called Itemstabs with a minimum of Title Block, Page Two, Page Three, and Attachments tabs.

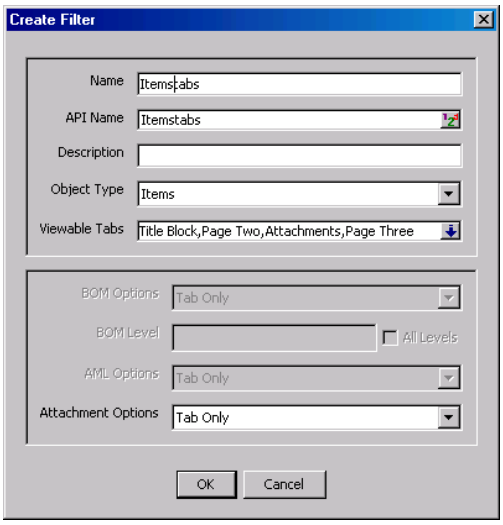

Figure 15 The Create Filter dialog box attributes and settings

Dynamic Document Generation does not support the **Files** option. Use only the **Tabs Only** option and avoid the **Tabs and Files** option.

- Step 4. Click **OK** and select the **Filter** tab to view the new filter
- Step 5. Select the settings shown in the following figure and then click **Save**.

[11] When defining a filter, use the API Name that was used for the object that you plan to publish.

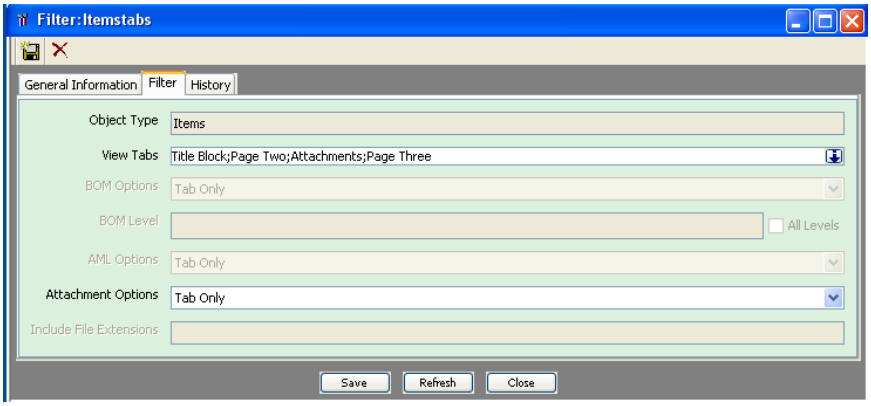

Figure 16 The ACS Filter settings

Agile PLM Server Configurations

Dynamic Document Publishing involves configuring and deploying the following Agile PLM Event Management components:

- **Event Node** – Event masks are configured around Event types. For example, Create Object, Delete Object, Audit for Workflow. Agile PLM provides a list of pre-defined Events for which an event can occur.
- **Event Handler** – Handler masks configure a custom action that is called when the Event is raised. They extend the function of an action taken by a user, interface, or the system when the Event subscription is triggered.
- **Event Subscriber** – Subscriber masks link a Handler mask to an Event mask.

Deploying these components enables creating the Templates, and generating the schema and document files. These configurations make use of PXs described in **Error! Reference source not found.**. For more information on Event components, refer to *Agile PLM Administrator Guide* and *Agile PLM SDK Developer Guide - Developing PLM Extensions*.

Understanding Process Extensions and Events Framework

Process Extensions (PX) is a framework for extending the functionality of the Agile PLM system. The functionality can be server-side extensions, or extensions to client-side functionalities, such as external reports or new commands added to the Actions menu or Tools menu. Regardless of the type of functionality a PX provides, all custom actions are invoked on the Agile Application Server rather than the local client.

In Agile SDK environment, Event Management framework extends the PX framework to enable developing and deploying event-driven applications. Events act as trigger points for generating an automation action within the PLM application. Every Event is generated from a source within Agile PLM applications. The source can be a business action triggered by a user, a UI action, or a system

initiated action. Agile PLM's Event framework supports developing extensions using the Java programming language and Groovy Script.

For information to develop Java PXs/Script PXs and Events, refer to the latest release of the *Agile PLM SDK Developer Guide - Developing PLM Extensions* and *Agile PLM Administrator Guide*. You can find referential and procedural information about PXs, Events, and Event triggers in these documents.

Java and Script PXs described in this chapter, namely, TemplateManagementStructureCreationPX, SchemaGenerationPX, DataGenerationPX, and DocumentGenerationPX make use of settings defined in Agile PLM Server Configurations.

Using Oracle Supplied Document Publishing PXs

Oracle provides Document Publishing configuration examples for PLM server and PLM Administrator. Server examples include PXs and related Java and properties files. Configurations described in setting up the PLM server use the following Oracle-supplied Java and Script PXs (Event Handlers):

- TemplateManagementStructureCreation
- SchemaGeneration
- DataGeneration
- DocumentGeneration

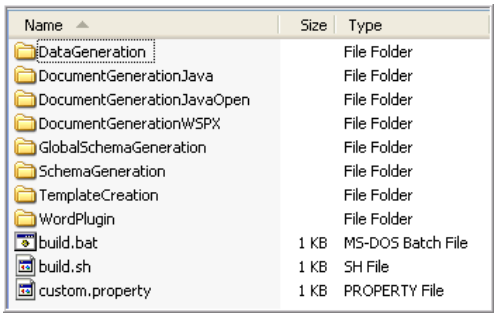

Figure 17 Oracle-supplied PXs

You can find these files in the Doc-Publishing folder, in SDK_samples.zip. The SDK_samples.zip folder which also contains the API HTML reference files and Sample applications is maintained on the Oracle Agile PLM Event and Web Services Samples Web site at: https://codesamples.samplecode.oracle.com/servlets/tracking/id/S614.

You can use these PXs to create the Event Handler and Event Subscriber that triggers the Event. For details, see Template Management PX, Schema Generation PX, Data Generation PX, and Document Generation PX. Alternatively, you can use the information to develop your own Java Client and server configuration. For more information and procedures to access its contents, contact your system administrator, or refer to your Agile PLM Installation Guide.

Working with Script and Java PX Handlers

Doc-Publishing folder contains both the Java PX and Script PX handlers. The Java Handlers provide the Java, Properties, and Resources files that you need to deploy the sample Java PX and Script PX handlers.

To deploy the Script PXs, refer to *Agile PLM Administrator Guide*. To deploy the Oracle supplied Java PXs, you must first create the JAR files by completing the following steps.

- Step 1. In **Doc-Publishing** folder, open the custom.property file and specify the name and path for the following as shown in **Error! Reference source not found.** below.
    - Agile PLM server in **ias.deploy**
    - Application server (OAS, or WLS) in **ias.home**
    - The location the PXs will reside in your environment in **px.deply**

```
ias.deploy.dir=D:/build/agile931/agileDomain/applications/APP-INF/lib
ias.home=D:/OC4J10133
px.deploy.loc=D:/build/agile931/integration/sdk/extensions

ANT_HOME=C:/apache-ant-1.7.1
```

Figure 18 Custom Properties file settings for Jar files

- Step 2. Make sure you are running apache-ant-1.7.1 and specify the path for ANT_HOME.
- Step 3. Run **build.bat** or **build.sh** to create all JAR files for Windows and UNIX environments respectively.
- Step 4. Make sure the JAR files are in "<AgileHomeDir>\integration\sdk\extensions" directory.

Conclusion

Businesses are starting to realize the many benefits of dynamic Document Publishing of product information. Agile PLM's document publishing solution supports dynamic generation of a wide range of product documents in RTF, PDF, EXCEL, HTML, and PowerPoint file formats using the most current product data maintained by the Oracle Agile PLM Application Suite.

The Oracle Agile PLM Application Suite enables enterprises to accelerate product innovation and maximize product profitability by managing the information, processes, and decisions about their products throughout the product lifecycle.

This solution can benefit Industrial, Retail, Life Sciences, Pharmaceutical, and High Technology industries. These enterprises can dynamically generate new structured document templates (Product Data sheets, Parts List, Service Manual), browse and insert PLM metadata and file contents into documents, create formatted reports from PLM objects, modify the content that is shared by other documents already stored in PLM, or update documents that reference the content that was modified.

The Oracle Agile PLM product suite provides comprehensive support for the PLM business process through Product Collaboration (PC), Product Quality Management (PQM), Product Portfolio Management (PPM), Product Compliance Management (PCM), Product Governance & Compliance (PG&C), and Engineering Collaboration (EC) modules.

Document Publishing Solution
January 2012

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200

oracle.com

Oracle is committed to developing practices and products that help protect the environment

Hardware and Software, Engineered to Work Together