Oracle White Paper
v 9.3.2
Part No. E28667-01
December 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 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.

New in Release 9.3.2

The major changes in the Document Publishing solution for Release 9.3.2 are the result of operating in a Web Logic Server (WLS) environment. Other enhancements and changes are:

- Resolution of reported issues to extract the sample files on WLS and broken PXs
- Setting BI Publisher option to read Agile XML files. See Installing and Setting Up BI Publisher Desktop.

Content and Organization

Information provided in this document is organized as follows:

- **Introduction** – This section describes the solution, the required environment, and applicable processes.
- **Installing BI Publisher and Defining the Template files** – This section provides information to install and set up the BI Publisher and ancillary tools and define templates and publish reports
- **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.
- **Generating a sample report** – 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 performing these tasks, 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 document 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 can support other authoring tools such as Adobe's Framemaker, this Whitepaper uses Word and Oracle BI Publisher to generate these reports. 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.

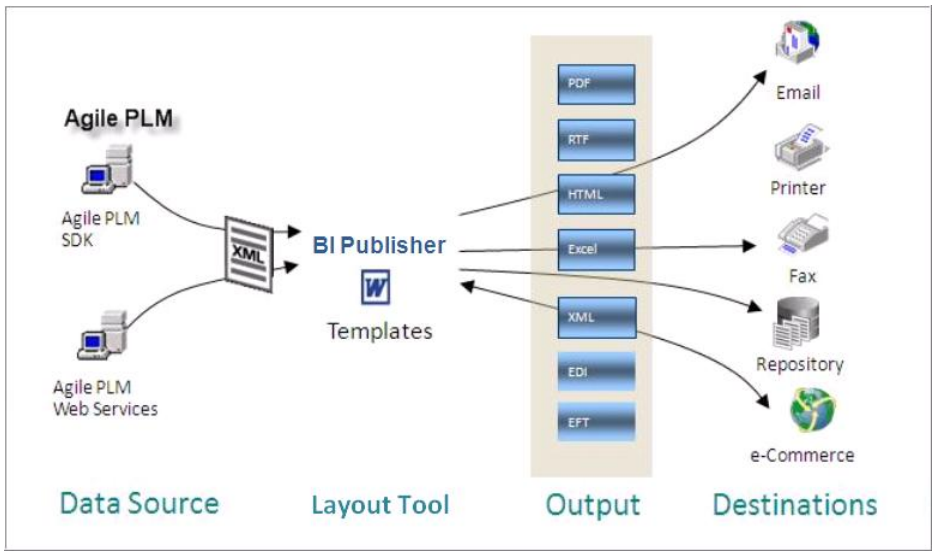


Figure 1 Document Publishing architecture

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 generation and management solution from Oracle. The operating environment includes:

- Oracle Agile PLM
- Oracle BI Publisher
- Microsoft Word

Oracle Agile PLM Components

- Agile PLM Release 9.3.2 (Server and databases)
- Agile PLM Release 9.3.2 File Manager
- Agile PLM Release 9.3.2 SDK (Template Management Java and Script PXs)
- Agile PLM Release 9.3.2 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.

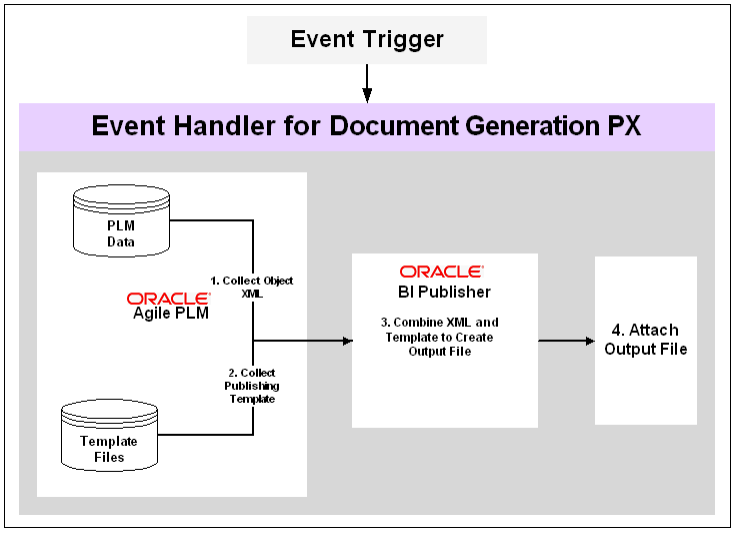


Figure 2 Dynamic Document Publishing Process and Steps

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,

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

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.

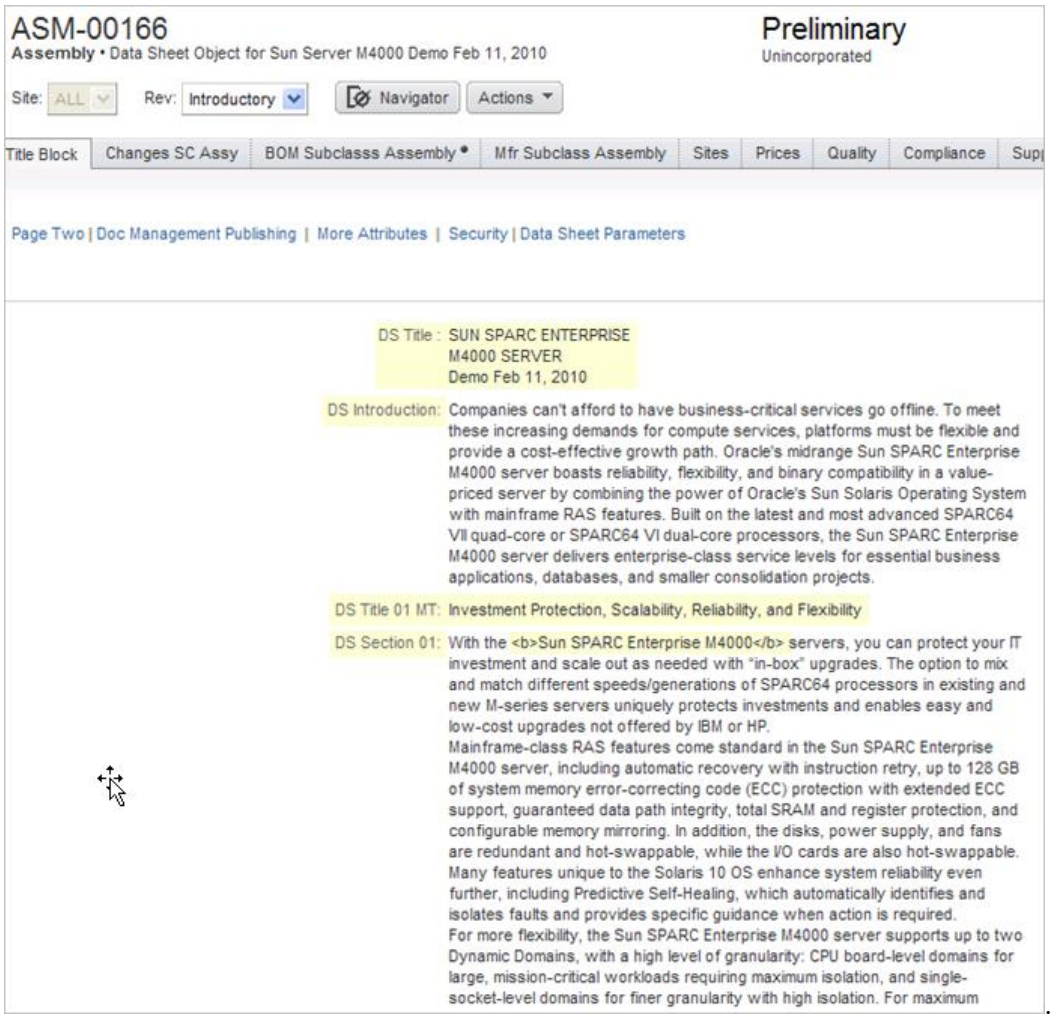


Figure 4 Data sheet attributes in Agile PLM

Generating the Data XML File

When the Web Service loadXMLData is invoked, the resulting XML output looks like the figure below. The samples in this document describe how to generate an attachment in Agile PLM of the XML file. Download the XML file from Agile to your computer. Using the BI Publisher menus in Word, select to load Sample XML and open this file. 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

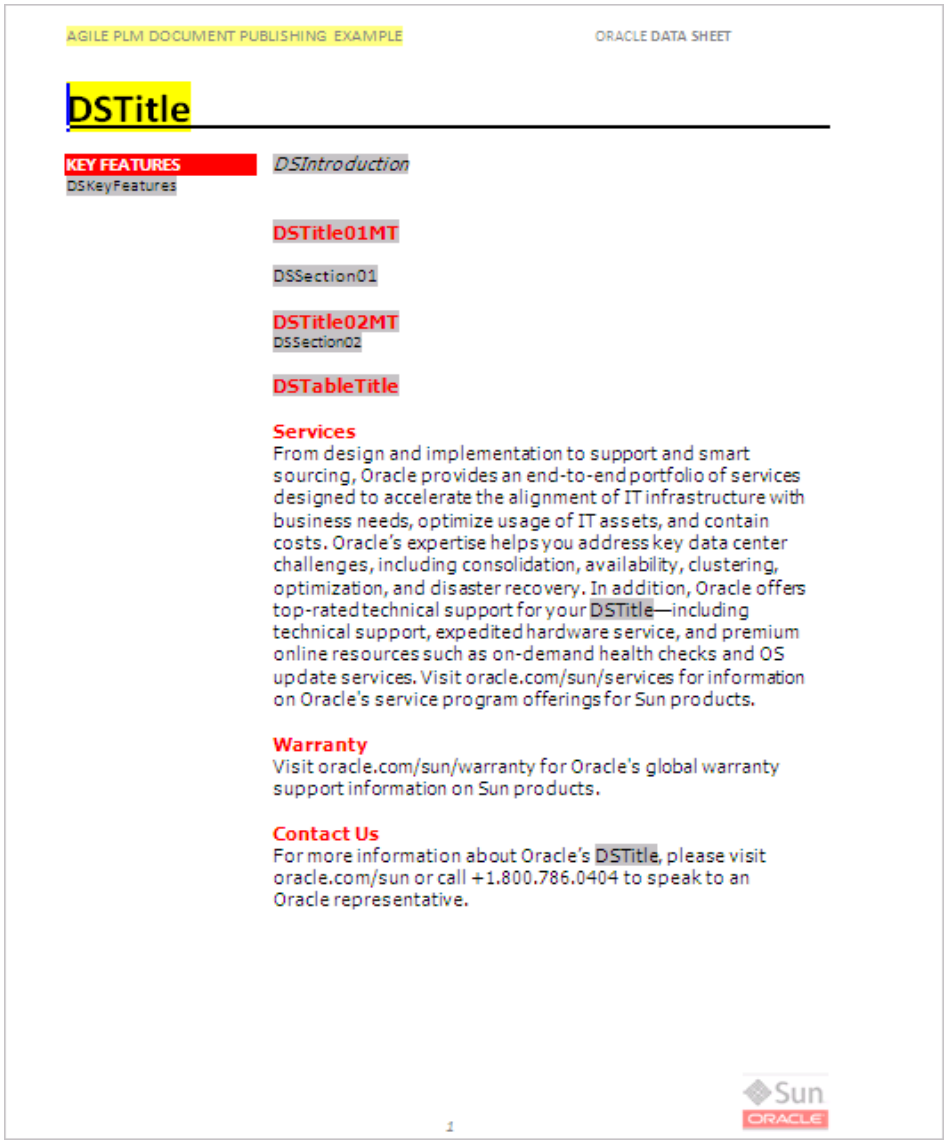
Template is an RTF file created and formatted using Word, BI Publisher, and object's attributes in Agile PLM. For procedures, see Building BI Publisher Templates and Cited References. The following XSD file assumes you have generated the Schema file using the Sample, and then downloaded and loaded it using the BI Publisher,

```
<?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. Figure 7 shows the output in RTF format. Upon completion of the testing process, you must load the completed template into Agile PLM template location for use in the Even trigger.


AGILE PLM DOCUMENT PUBLISHING EXAMPLE

ORACLE DATA SHEET

DSTitle

KEY FEATURES
DSKeyFeatures

DSIntroduction

DSTitle01MT

DSSection01

DSTitle02MT
DSSection02

DSTableTitle

Services
From design and implementation to support and smart sourcing, Oracle provides an end-to-end portfolio of services designed to accelerate the alignment of IT infrastructure with business needs, optimize usage of IT assets, and contain costs. Oracle's expertise helps you address key data center challenges, including consolidation, availability, clustering, optimization, and disaster recovery. In addition, Oracle offers top-rated technical support for your DSTitle—including technical support, expedited hardware service, and premium online resources such as on-demand health checks and OS update services. Visit oracle.com/sun/services for information on Oracle's service program offerings for Sun products.

Warranty
Visit oracle.com/sun/warranty for Oracle's global warranty support information on Sun products.

Contact Us
For more information about Oracle's DSTitle, please visit oracle.com/sun or call +1.800.786.0404 to speak to an Oracle representative.

1

Sun ORACLE

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.

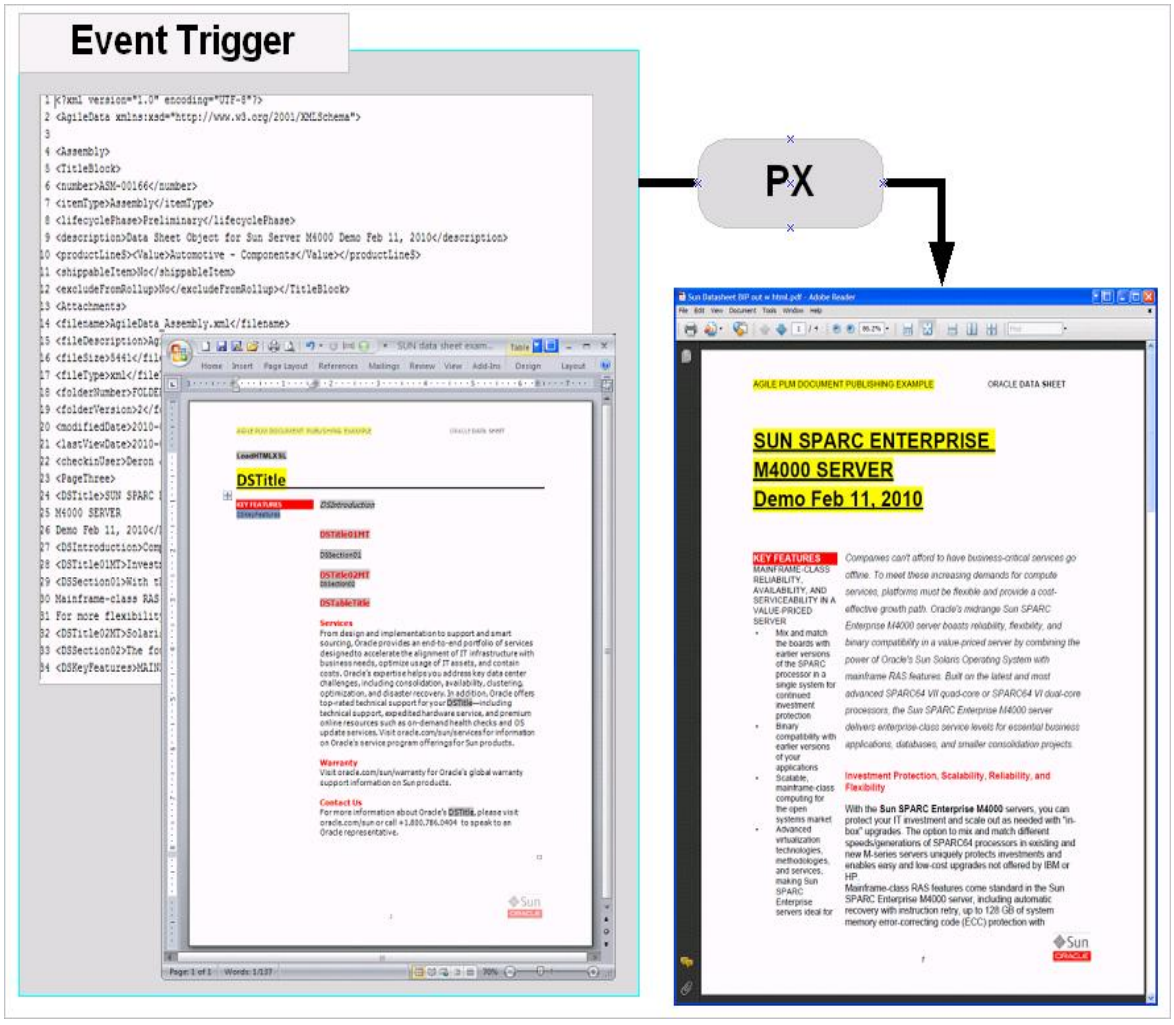


Figure 8 Combining Data XML and Template to generate the document

Setting Up the Environment for Document Publishing

Setting up the environment requires installing and the following installations and BI Publisher and Agile PLM configurations. The setup documented below, supports the shipped samples. If the samples are altered to use different classes and attributes, then these configurations are not necessary.

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

.

Installing and Setting Up BI Publisher Desktop

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

To install BI Publisher Desktop:

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
2. Depending on your version of Windows and Word, verify either the Add-Ins, or BI Publisher tab is available in MS Word.

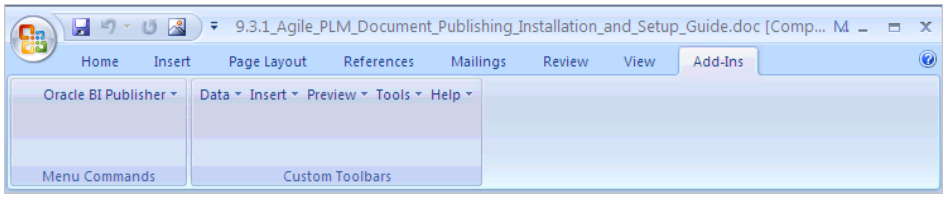


Figure 9: BI Publisher Add-Ins in MS Word

To set up BI Publisher Desktop to read aXML files in Design mode:

1. Open a Word document and select **Add-Ins** > **Tools > Options** or **BI Publisher > Options.** The **Option** dialog appears.
2. In **Option** dialog, select the **Build** tab. The following appears.

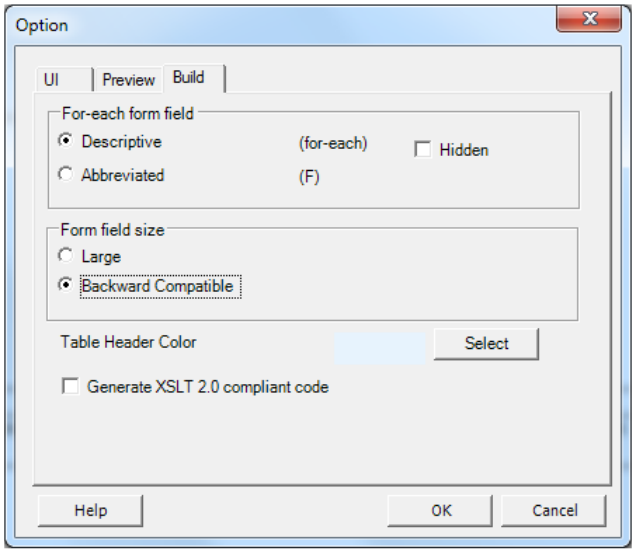


Figure 10: The option dialog

3. In **Form field size** box, select **Backward Compatible**.
4. Click **OK**.

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

Note: Configuring or not configuring this setting does not prevent BI Publisher from reading aXML files in the Design mode. It only produces an error when the embedded BI Publisher tries to access a document from within Agile.

Performing Agile PLM Administrator Configurations

The Agile PLM Administrator configurations include:

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

Creating the DocumentTemplate Subclass

This is a new Document 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.

Note: This is a onetime configuration that creates a subclass which serves as a place holder for all Template files organized by Base Class, Class, and Subclass. This is typically used in a Test or QA system, and is not required in a production environment.

To create the DocumentTemplate Subclass do as follows[8]:

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

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

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

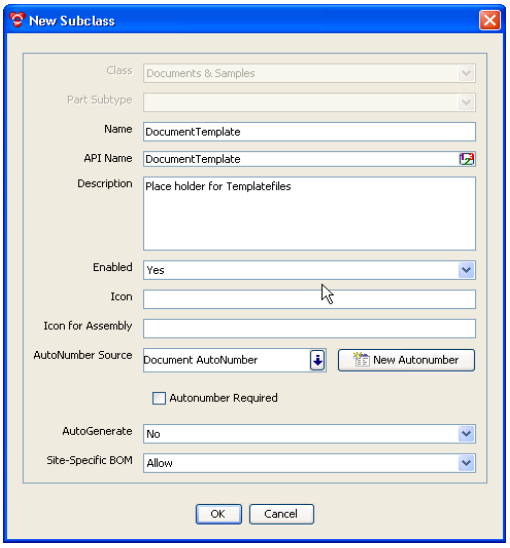


Figure 10 DocumentTemplate subclass settings

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.

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

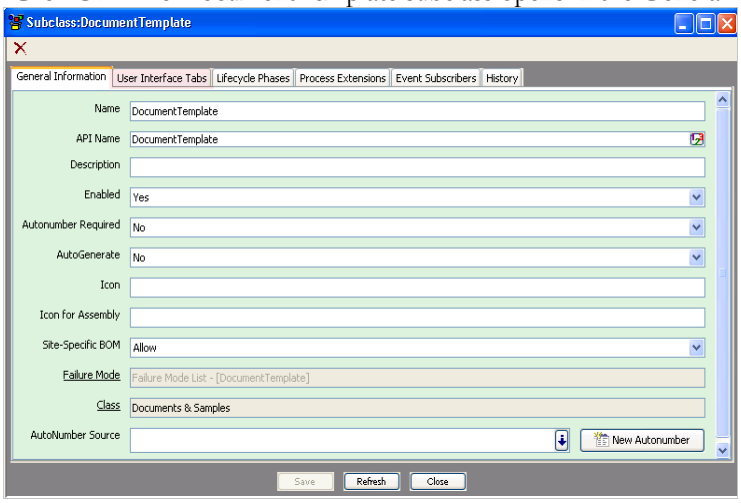


Figure 11 DocumentTemplate General Information page

Setting the Title Block Number Fields

Complete the following steps to set these fields.

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

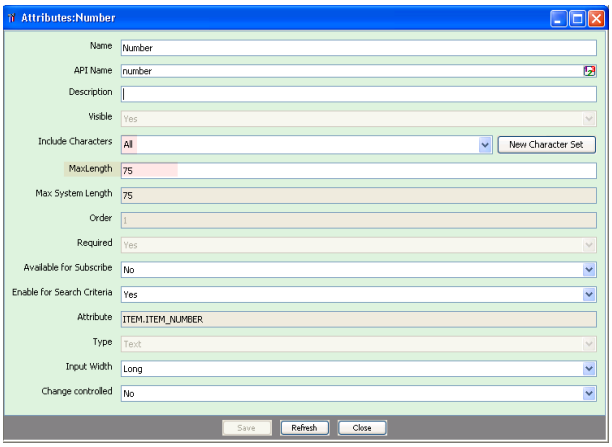


Figure 12 Page 2 Attributes for the object

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

Note: 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

Configuring Object Information

The samples expect to read 3 pieces of information from attributes on the Object. These are: the location of the BI Publisher Template, ACS filter API name for the object, and output format. Document publishing Web Services rely on Agile Content Services (ACS) filters to determine the data that is returned for the object in the XML file.

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[9]. The Base IDs are for later use.

- **Heading field** – This is for BI Publisher to display the Doc Publishing attributes in a Heading area.
- **Output Type -List field** – This Alpha Type filed determines the Output Type (EXCEL, RTF, PDF, and HTML). The sample PXs assume this field is stored as **List11** with Base **ID 1271**.
- **ACS Filter -Text field** – This is for the Filter and assumes **Text 12** and Base ID **1302**. PXs read this attribute to correctly call the Web Service with a Filter for Exporting the object information. An empty filter will cause an error running the PXs.
- **Template Holder -Text field** – This is for the Object identifier of the BI Publisher Template and assumes **Text11** and Base **ID 1301**. The PXs will retrieve the BI Publisher template from the Attachments Tab of the object in this attribute.

Complete the following steps to define these fields.

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

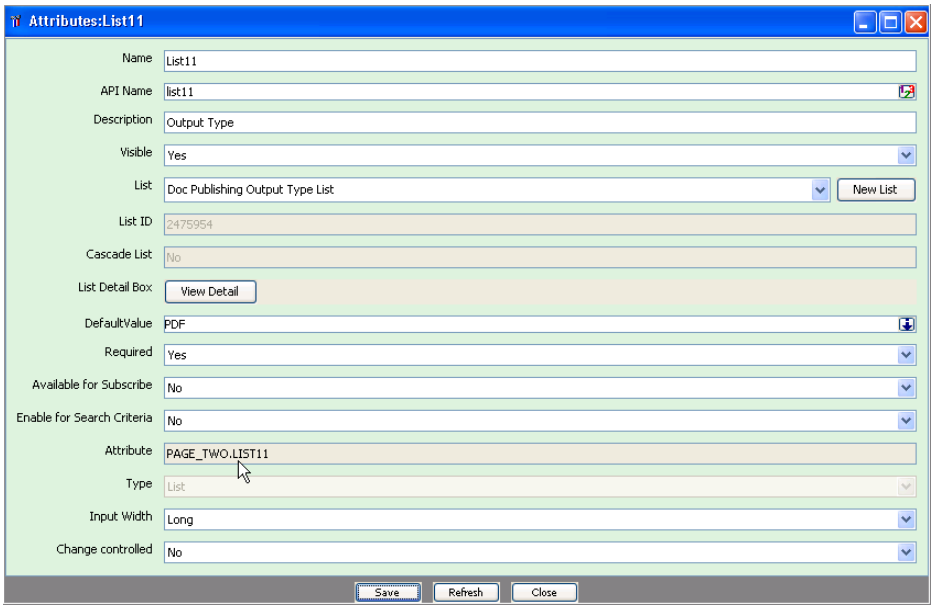


Figure 13 Text field attributes settings

2. Click **Save**.
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 .

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

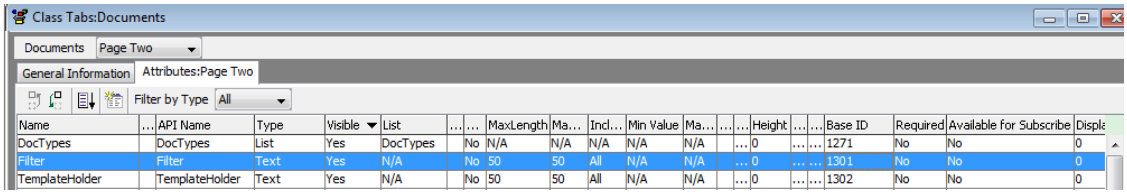


Figure 14 Object Page 2 fields for Document Publishing sample

> 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. Fields selected for the filter provide flexibility for the sample and you can alter them for a production environment[10]. In this Whitepaper, the Default Item Filter is selected for this purpose.

To access and set the Default ACS Item Filter:

1. Log in to Agile PLM Java Client with administrator privileges.
2. Select **Admin > System Settings > Agile Content Service > Filters > Default Item Filter**. The following dialog appears.

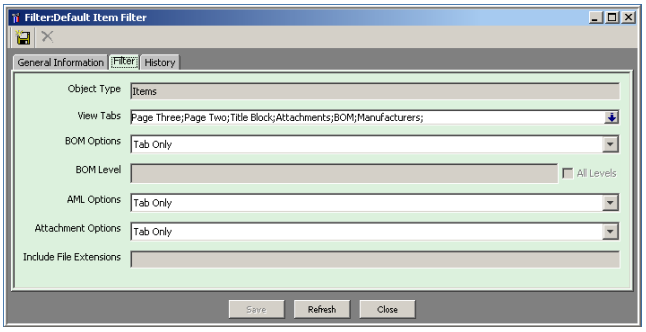


Figure 15 The Create Filter dialog box attributes and settings

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

3. Make sure the **Tab Only** option is selected for BOM Options, AML Options, and Attachments Options. In **View Tabs**, a minimum of **Page Three**, **Page Two**, **Title Block, Attachments**, **BOM**, and **Manufatctures** options are selected.
4. Click **Save** to save the new settings.

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

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. For 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** – Generates objects in DocumentTemplate subclass in a tree representing every base class, class, and subclass.
- **SchemaGeneration** - Generates an XML Schema Attachment for the current object's subclass and adds it to the DocumentTemplate object for the current subclass.
- **DataGeneration** - Creates an XML Data attachment and adds to a new object in DocumentTemplate subclass called "DocumentTemplate."
- **DocumentGeneration** – Publishes the document when the Event I triggered.

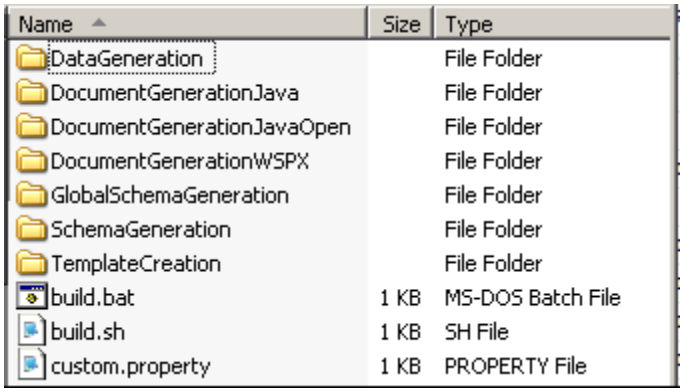


Figure 16 Oracle-supplied PXs

You can find these files in <release#>Doc-Publishing_samples.zip which is maintained on the Oracle Agile PLM Event and Web Services Samples Web site at: http://www.oracle.com/technetwork/indexes/samplecode/agileplm-sample-520945.html You can use these PXs to create the Event Handler and Event Subscriber that trigger 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*.

Customizing Settings in JavaOpen PX's web.xml File

This is one of the files in the DocumenGenerationJavaOpen folder. You can find DocumenGenerationJavaOpen in the Doc-Publishing folder.

Customizing this file to enable the PX to run in your environment requires modifying the URL, USERNAME, and PASSWORD parameters. To this end, set **USERNAM**E to **admin**, **PASSWORD** to **agile1**, and the **URL** parameter as shown below.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
 <web-app xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd" version="2.4">
   <display-name>Agile932 Doc-Publishing URL PX</display-name>
   <description>URLPX Servlet</description>
   <servlet>
   <servlet-name>PX</servlet-name>
   <servlet-class>samples.DocumentGeneration.DocumentGenerationJavaPxOpen.DocumentGenerationJavaPxOpen</servlet-class>
  <init-param>
   <param-name>URL</param-name>
   <param-value>
        http://<your-server>.us.oracle.com:7001/Agile</param-value>
   </init-param>
   <init-param>
   <param-name>USERNAME</param-name>
   <param-value>admin</param-value>
   </init-param>
    <init-param>
   <param-name>PASSWORD
   </param-name>
   <param-value>agile1</param-value>
   </init-param>
   </servlet>
   <servlet-mapping>
   <servlet-name>PX</servlet-name>
   <url-pattern>/PX</url-pattern>
   </servlet-mapping>
   </web-app>
```

Extracting the WLS Application File

The required Agile Application file is **application.ear** which is a .ZIP file and is located in the 9.3.2 Install folder. The path to this file is **agileDomain > applications > application.ear**. You must extract this file into **Application** folder and specify the path as shown in **Figure 17**.

Creating JAR Files and Deploying 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.

1. In **Doc-Publishing** folder, open the **custom.property** file and using the information in Figure 17, specify the name and path for the Agile PLM server in **wls.deploy**, the name and path for the Application server in **wls.home**, and the location that the PXs will reside in your environment, in **px.deploy**.

```
#ias.deploy.dir=D:/builds/agile932/wls12c/apcm/agileDomain/applications/APP-INF/lib
#ias.home=D:/OC4J10133
# WLS is only supported server since 932
wls.deploy.dir=D:/builds/agile932/wls12c/apcm/agileDomain/applications/APP-INF/lib
wls.home=c:/oracle/Middleware12c
px.deploy.loc=D:/builds/agile932/wls12c/apcm/integration/sdk/extensions

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

Figure 17 Custom Properties file settings for Jar files

2. Make sure you are running apache-ant-1.7.1 and specify the path for ANT_HOME.
3. Make sure the JAR files are in "<AgileHomeDir>\integration\sdk\extensions" directory and **wls.home** is set to Weblogic Server 12 installation folder, and then run **build.bat** to create the JAR files for Windows environment, or **build.sh** to create them for the UNIX environment.

   After runnning the build files, you will find the following .JAR files in the Doc-Publishing folder.

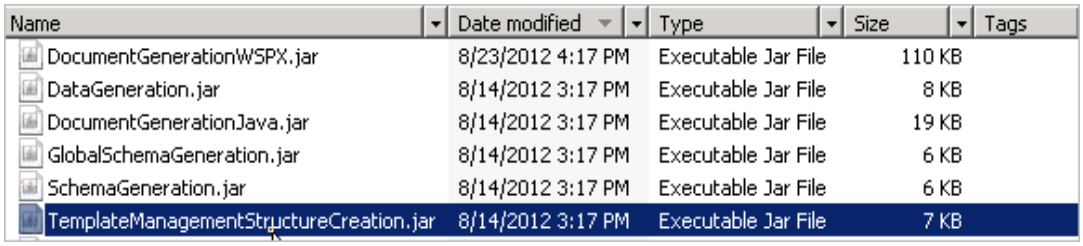


Figure 18 PX JAR files

You can use these PXs to implement the Dynamic Document Publishing capabilities and create the Event Handler and Event Subscribers that trigger these Events.

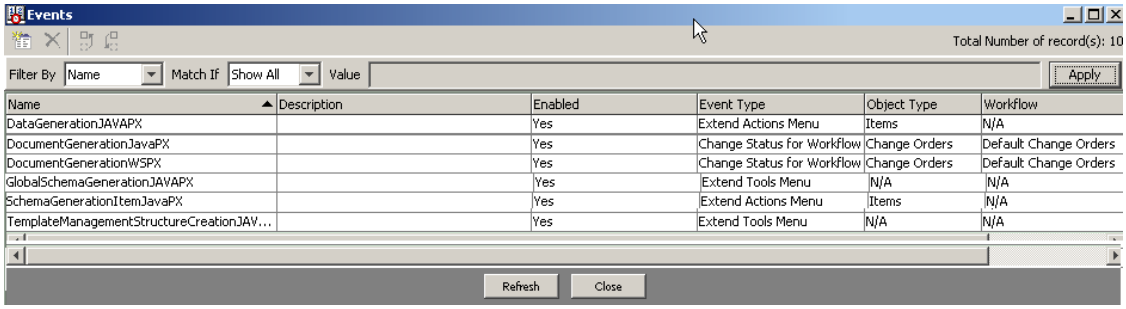


Figure 19 Events

Creating Events and Even Subscribers

For procedures, see: "Configuring the TemplateManagementStructureCreationPX", "Configuring the SchemaGenerationPX", "Configuring the DataGenerationPX", "Configuring DocumentGenerationPX". Alternatively, you can use the information to develop your own Java Client and server configurations.

Publishing the Sample

Steps in publishing the sample are illustrated in Figure 20 Steps to publish a document. For more information to go about completing these steps, see The Task Sequence.

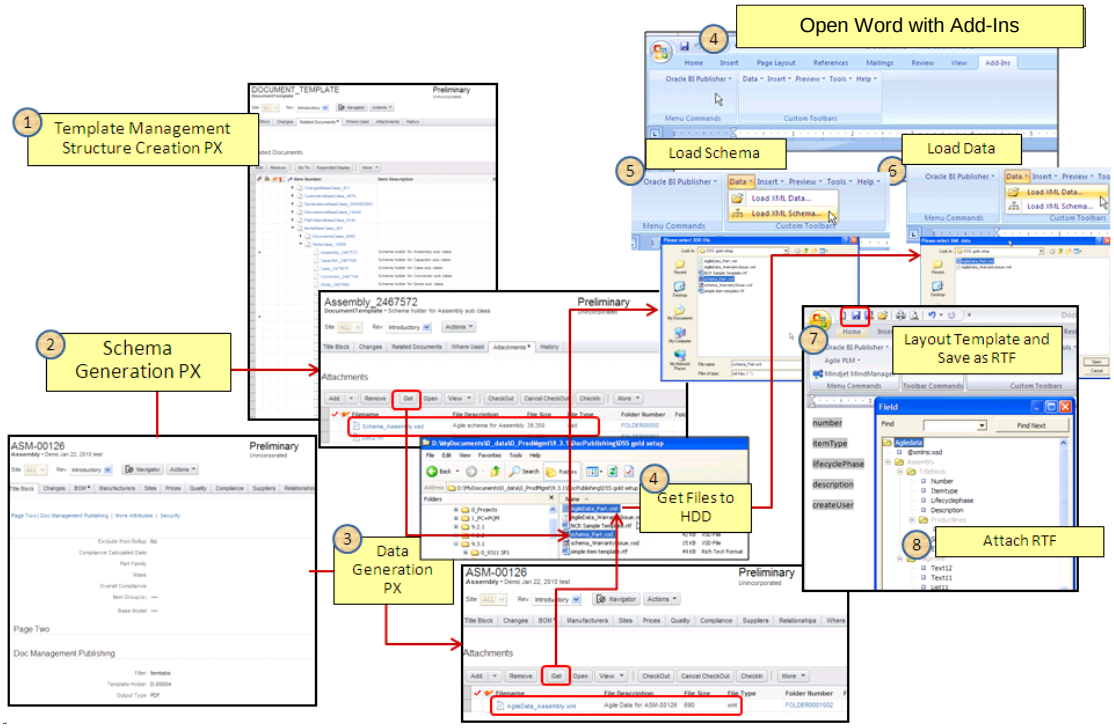


Figure 20 Steps to publish a document

The Task Sequence

Publishing a document requires completing of the following tasks:

1. Configuring and running the TemplateManagementStructureCreationPX
2. Configuring and running the SchemaGenerationPX
3. Configuring the DataGenerationPX
4. Downloading the schema (XSD) and data (XML) files to the local drive
5. Loading the schema file
6. Loading the data file
7. Laying out the BI Publisher Template and saving the Word file in RTF format
8. Uploading the Template into Agile PLM
9. Triggering the Event to create the output file

Configuring the TemplateManagementStructureCreationPX

The TemplateManagementStructureCreationPX creates a 3 level Bill of Material (BOM) for Base Classes, Classes, and Subclasses defined in Creating the DocumentTemplate Subclass. This is a placeholder for all future .RTF template files[11].

You can find this Script or Java PX in SDK Samples Folder and Document Publishing Examples. The paths to the Script PX and Java PX with its .JAR and Properties files are:

- **Script PX for WLS** – 932_wls_sdk\samples\Doc-Publishing\TemplateCreation\TemplateCreationScript.groovy
- **Java PX for WLS** – 932_wls_sdk\samples\Doc-Publishing\TemplateCreation\samples\TemplateManagementStructureCreation\TemplateManagementStructureCreationPX.java

Configuring Event Masks for TemplateManagementStructureCreationPX

This procedure creates the necessary Event masks, Handler masks, and Subscriber masks for the PX.

To create Event mask and set Event Type do as follows:

1. Log in to Java Client with Admin privileges.
2. In Java Client, select **Admin > System Settings > Event Management > Events**.
3. In Events page, select the **New** button to open the **Create Event** dialog and define an Event mask called **CreateBomTemplate** for **Object Type Parts** with the settings shown in Figure 21 and the Click **OK**.

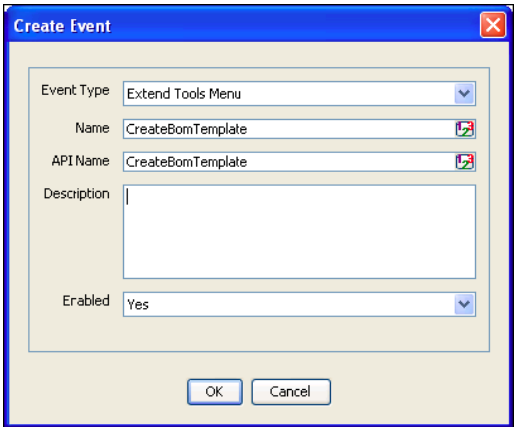


Figure 21 Create Event mask for TemplateManagementStructureCreationPX

To set up the Event Handler mask do as follows:

1. In Java Client with Admin privileges, select **Admin > System Settings > Event Management > Event Handlers**.
2. In **Event Handlers** pane, select the **New** button to open the Create Event Handler dialog.

[11] This PX is run once only and is not necessary if the Schema Structure is not needed.

3. Create a new Event Handler mask called **CreateBomTemplate**.
4. Set Enabled to **Yes**, and for Role, select the applicable roles. For example, **Quality Administrator**, **Quality Analyst**, **Quality Analytics User**. For **Event Handler Type**, you have the option to select the **Script** or **Java PX**. You can find the Oracle-Supplied Script and Java PXs in Doc-Publishing folder in SDK_samples.zip. See Oracle-Supplied Document Publishing PXs.

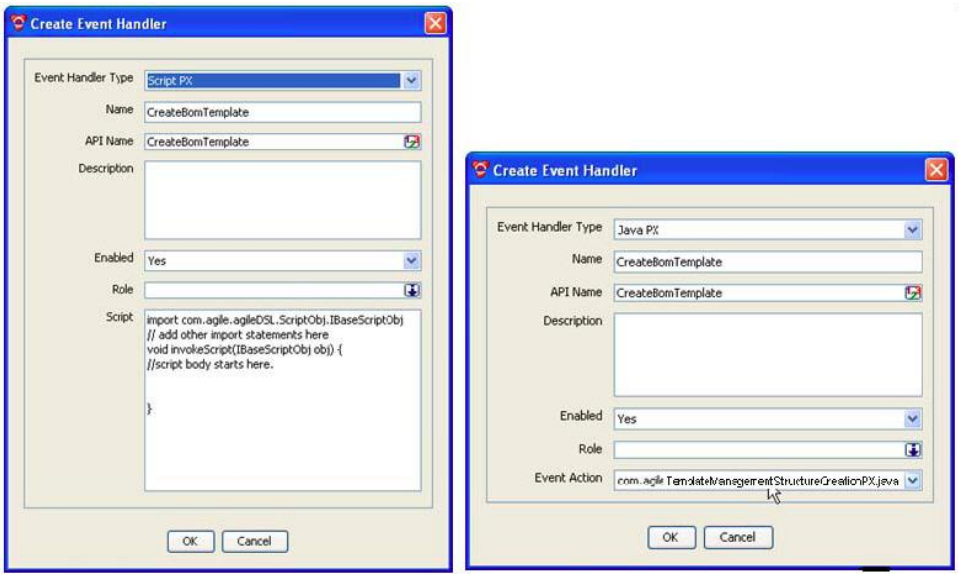


Figure 22 Create Event Handler dialogs for Java and Script PXs

To configure the Script PX Event Handler:

1. In Create Event Handler, paste the contents of TemplateCreationScript.groovy file in the dialog's **Script** box.
2. Click **OK**.
   For more information, refer to Agile PLM Events and Event Framework chapter in *Agile PLM SDK Developer Guide - Developing PLM Extensions*. You can also find information on configuring Script PXs in *Agile PLM Administrator Guide*.

To configure the Script or Java PX Handler:

1. Make sure the Event Action for this Java PX is deployed. For procedures, see Creating JAR Files and Deploying Script and Java PX Handlers.
2. Check the values in TemplateManagementStructureCreation.properties file and make sure they conform to Java Client Admin settings shown in Properties File Settings for TemplateManagementStructureCreationPX.
3. Click **OK**.

To configure the Event Subscriber mask:

1. In Java client with Admin privileges, select Admin > System Settings > Event Management > Event Subscribers.
2. In **Event Subscribers** pane, select the **New** button to open the Create Event Subscriber dialog.
3. Create a new Event Subscriber mask called CreateBomTemplate with the following settings:
   - Enabled to **Yes**
   - Trigger Type to **post**
   - Execution Mode to **Synchronous**
   - Error Handling Rule to **Stop**

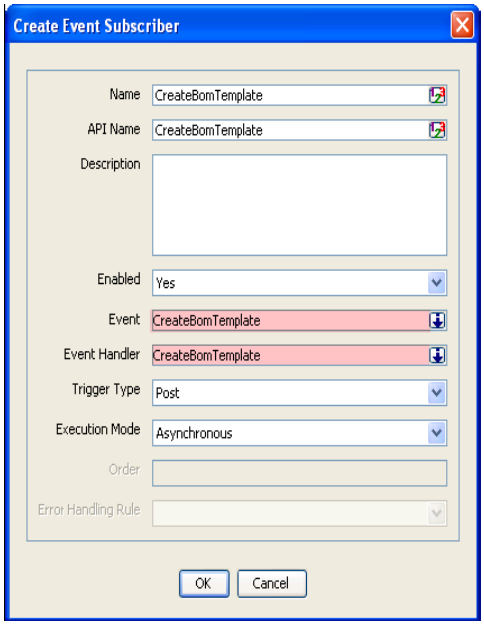


Figure 23 Template Management Event Subscriber

4. Click the drop-down arrow to select the Event and Event Handler you created earlier.
5. Click **OK**.

Properties File Settings for TemplateManagementStructureCreationPX

Values set in the Oracle-Supplied Properties file are shown in the shaded region. Make sure these values conform to Java Client Admin settings for this PX.

```
# TEMPLATE_SUBCLASS_API_NAME          -------------  Value for this property should be API name of the new subclass of "Documents" class which is
#                                                    created for holding the template BOM

# ROOT_TEMPLATE_OBJECT_NUMBER         -------------  Value for this property is the object number of the root object,the BOM tab of which will contain
#                                                    objects for each agile class in hierarchical manner

# CLASS_TEMPLATE_OBJECT_NUMBER_FORMAT  -------------  This defines constituents of the  object number created for each class of  Agile hierarchy

# OBJECT_DESCRIPTION                  -------------  This defines the description of child object created for every sub class.

# LOGGING                             -------------  This accepts true/false as value. When this is set to true log messages gets
#                                                    printed on the application server console

-------------------------------------------------------------------------------------------------------------------------------------------------------

# API_NAME                            ------------- Indicates API name of the class for which object is being created

# CLASS_NAME                          ------------- Indicates class name of the class for which object is being created

# CLASS_ID                            ------------- Indicates class id(ex 10141))  of the class for which object is being created

-------------------------------------------------------------------------------------------------------------------------------------------------------
# Note: No property should be removed from this property file


TEMPLATE_SUBCLASS_API_NAME=DocumentTemplate

ROOT_TEMPLATE_OBJECT_NUMBER=DOCUMENT_TEMPLATE

CLASS_TEMPLATE_OBJECT_NUMBER_FORMAT=API_NAME+ "_" +CLASS_ID

OBJECT_DESCRIPTION="Schema holder for "+CLASS_NAME+" sub class"

LOGGING=true
```

Figure 24 Properties file settings for TemplateManagementStructureCreationPX

If there are no changes to the PX, you can use the JAR files described in Creating JAR Files and Deploying PXs (Event Handlers). If you need to modify the Java or Script PX, do as follows:

- **For Java PX**:
    1. Copy "TemplateManagementStructreCreation.jar" to "<AgileHomeDir>\integration\sdk\extensions".
    2. Unpack "TemplateManagementStructreCreation.jar" to gain access to "ResourceTemplateManagement.properties" file.
    3. Update as needed.
    4. Repack and recopy to PLM server.
- **For Script PX**:
    1. Open the Handler in PLM client.
    2. Configuration is at the beginning of the Script. Modify the Script as needed.
    3. Save the modified Handler.

Output Generated by TemplateManagementStructureCreationPX

The Script PX and Java PXs are invoked from the Tools menu and when triggered will do as follows:

1. Configure the Template Subclass and create a new Documents Subclass called **"DocumentTemplate"** for Item – Document.

Create a 3-level BOM with Level 1 for all Agile Base classes in the system.

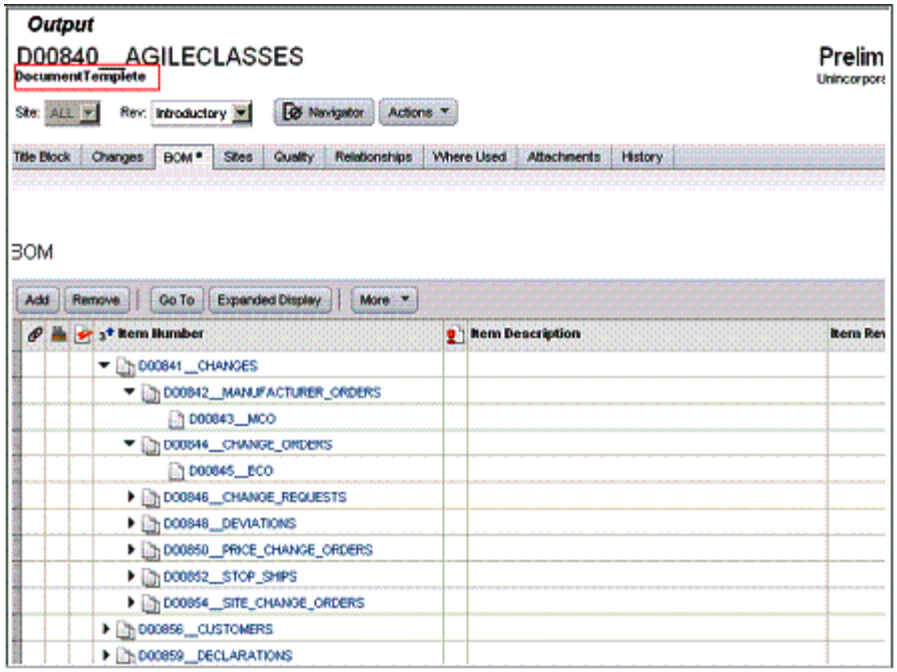


Figure 25 Template Management Outputs

Configuring the SchemaGenerationPX

The purpose of SchemaGenerationPX is to programmatically generate XML schema files using the Agile Java API for a given object.

It is necessary to run this PX for each Subclass to generate the Schema XSD file. Alternately, you can run the GlobalSchemaGeneration.jar Java PX from the Tools menu to generate a schema for ALL subclasses in the system. The Schema XSD will be attached to the applicable object in the Template (Schema) Management Structure'

The paths to the Script PX and Java PX with its .JAR and Properties files are:

- **Script PX** – 932_wls_sdk\samples\Doc-Publishing\SchemaGenerationScript.groovy
- **Java PX** – 932_wls_sdk\samples\Doc-Publishing\SchemaGeneration\samples\SchemaGenerationPX.java

Configuring Event Components for SchemaGenerationPX

Similar to TemplateManagementStructureCreationPX, these configurations require creating the Event and setting the Event Type, Event Handler (Java or Script PX), and Event Subscriber.

To create Event mask and set Event Type:

Follow the steps in " Configuring the TemplateManagementStructureCreationPX" to define an Event mask called **CreateSchema** for **Object Type Items** with settings in the following figure, and then click **OK**.

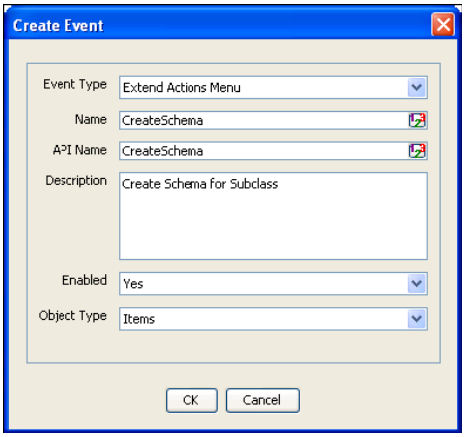


Figure 26 Create Event mask for SchemaGenerationPX

To set up the Event Handler mask:

1. Use the information in Configuring Event Masks for TemplateManagementStructureCreationPX and create a new Event Handler (Script PX, or Java PX) called **CreateSchema**.
2. Set Enabled to **Yes**, and for Role, select the applicable roles. For example, **Quality Administrator**, **Quality Analyst**, **Quality Analytics User**. For Event Handler Type, you have the option to select **Script PX**, or **Java PX**. You can find the Oracle-Supplied Script and Java PXs in Doc-Publishing folder in SDK_samples.zip. See Using Oracle-Supplied Document Publishing PXs for a description of these PXs and accessing SDK_samples.zip.
3. To configure your Script or Java PX Handler Type do as follows.
   - **For Script PX Event Handlers** - In Create Event Handler, paste the contents of Schema Generation Groovy script file in the dialog's Script box and click OK. For more information, see Agile PLM Events and Event Framework chapter in Agile PLM SDK Developer Guide - Developing PLM Extensions. You can also find information on configuring Script PXs in Agile PLM Administrator Guide.
   - **For Java PX Event Handler** - Make sure the Event Action for this Java PX is deployed. See Creating JAR Files and Deploying Script and Java PX Handlers. Check the values in SchemaGeneration.properties file and make sure they conform to settings defined in Java Client Admin in Properties File Settings for SchemaGenerationPX.

4. Click **OK**.

To configure the Event Subscriber mask:

1. In Java Client with Admin privileges, select **Admin > System Settings > Event Management > Event Subscribers**.
2. Event Subscribers pane, select the **New** button to open the Create Event Subscriber dialog.
3. Create a new Event Subscriber called **CreateSchema** with settings shown in Figure 27 and then Click **OK**.

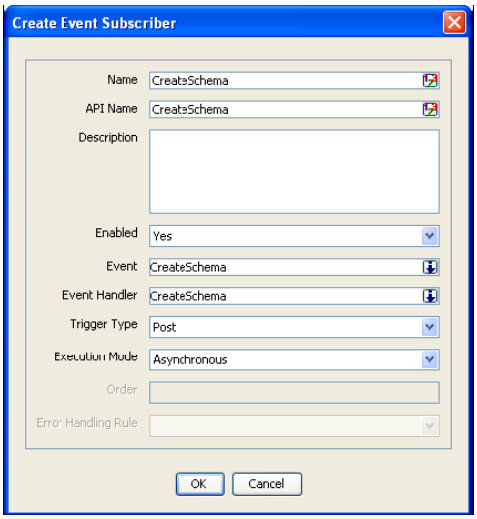


Figure 27 Create Event Subscriber

Properties File Settings for SchemaGenerationPX

Values set in the Oracle-Supplied Properties file are shown in the shaded region of the following illustration. Make sure these values conform to Java Client Admin settings for this PX.

```
# CLASS_TEMPLATE_OBJECT_NUMBER_FORMAT  -------------  value for this property should be same as mentioned in resourcetemplatemanagement.properties
# SCHEMA_FILE_NAME                     -------------  This property defines the constituents of the file name of the schema file
# SCHEMA_FILE_DESCRIPTION              -------------  This property defines the description for the schema file.
# LOGGING                              -------------  This accepts true/false as value. When this is set to true log messages gets
#                                                     printed on the application server console
-------------------------------------------------------------------------------------------------------------------------------------------------
# API_NAME indicates                   -------------  API name of the  object's  class for which schamea file is being generated
# CLASS_NAME                           -------------  Indicates class name of the  object's class for which schamea file is being generated
# CLASS_ID                             -------------  Indicates class id of the  object's class for which schamea file is being generated
-------------------------------------------------------------------------------------------------------------------------------------------------
# Note: No property should be removed from this property file
TEMPLATE_SUBCLASS_API_NAME=DocumentTemplate
CLASS_TEMPLATE_OBJECT_NUMBER_FORMAT=API_NAME+ "_" +CLASS_ID
# SCHEMA FILENAME should of xml type
SCHEMA_FILE_NAME="Schema"+ "_" +API_NAME+ ".xsd"
SCHEMA_FILE_DESCRIPTION="Agile schema for "+CLASS_NAME
LOGGING=true
```

Figure 28 Properties file settings for SchemaGenerationPX

If there are no changes to the PX, you can use the JAR files described in Creating JAR Files and Deploying PXs (Event Handlers). If you need to modify the Java or Script PX, then do as follows:

For Java PX:

1. Copy "SchemaGenerationPX.jar" to "<AgileHomeDir>\integration\sdk\extensions".
2. Unpack "SchemaGenerationPX.jar" to gain access to "SchemaGeneration.properties" file.
3. Update as needed
4. Repack and redeploy to PLM server

For Script PX:

1. Open the Handler in PLM client.
2. Configuration is at the beginning of the Script. Modify the Script as needed.
3. Save the modified Handler.

Output Generated by SchemaGenerationPX

When the Event is triggered from the Actions menu or Tools menu, a Schema for the sub class is created and added to the Template BOM created by TemplateManagementStructureCreationPX. It is necessary to run this PX for each Subclass you defined in Performing Agile PLM Administrator Configurations to generate the required Schema XSD file. Alternatively, you can run the GlobalSchemaGenerationPX from Tools menu and generate a Schema for all subclasses in the system. The Schema XSD file is attached to the applicable object in created in TemplateManagementStructureCreationPX..

The Schema Naming Convention is <ObjectClassName>:<ObjectSubClassName>:<SchemaSuffix>.

These attributes are:

ObjectClassName – This is the name of the class. For example, Document.

ObjectSubClassName – This is the name of the subclass. For example, Documents.

SchemaSuffix – The SchemaSuffix is set in the properties file.

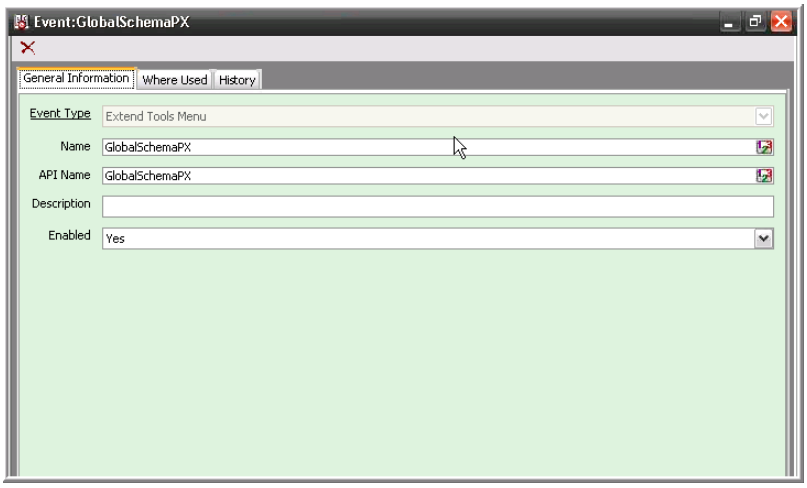


Figure 29 Event type settings for GlobalSchemaPX

In the SchemaGenerationPX output shown in Figure 30, Document is the Class name and Documents is name of the Subclass of Document.

```
<?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="DocumentTemplate" type="DocumentTemplateType" minOccurs="0" maxOccurs="unbounded" nillable="true"/
            </xs:sequence>
      </xs:complexType>

      <xs:complexType name="DocumentTemplateType">
            <xs:sequence>
                  <xs:element name="BOM" type="DocumentTemplateBOMType" minOccurs="0" maxOccurs="1" nillable="true"/>
                  <xs:element name="TitleBlock" type="DocumentTemplateTitleBlockType" minOccurs="0" maxOccurs="1" nillable="true"/>

                  ....
                  ....

                  <xs:element name="Instances" type="DocumentTemplateInstancesType" minOccurs="0" maxOccurs="1" nillable="true"/>
                  <xs:element name="PageThree" type="DocumentTemplatePageThreeType" minOccurs="0" maxOccurs="1" nillable="true"/>
            </xs:sequence>
      </xs:complexType>

      <xs:complexType name="DocumentTemplateBOMType">
            <xs:sequence>
                  <xs:element name="BOMRow" type="DocumentTemplateBOMRowType" minOccurs="0" maxOccurs="unbounded" nillable="true"/>
            </xs:sequence>
      </xs:complexType>

                  ....
                  ....

                  <xs:element name="itemDescription" type="xs:string" minOccurs="0" maxOccurs="1" nillable="true"/>
            </xs:sequence>
      </xs:complexType>
```

Generating Schema XSD and Data XML files

This requires triggering the first three Document Publishing Events. When they are triggered, the PXs will perform the following tasks in the listed order:

1. **TemplateManagementStructureCreationPX** – This PX will create a 3 level BOM for all Base Classes/Classes/Sub Classes in the system.
2. **SchemaGenerationPX** – This PX will generate the Schema file (.XSD) for the referenced objects.
3. **DataGenerationPX** – This PX will create the Data .XML file and attaches it to the object. As prerequisite, it requires creating Item – Part/Document and setting the Page Two attributes, in this case, DocType, Filter and TemplateHolder. These prerequisites for this PX were defined in Performing Agile PLM Administrator Configurations and "Configuring the DataGenerationPX."

Naming Convention for DocumentPublishing Events and Event Handlers

For your convenience and to facilitate search, names used for PX Events and Event Handlers start with letters **DP** for "Document Publishing." These are shown in the following illustrations.

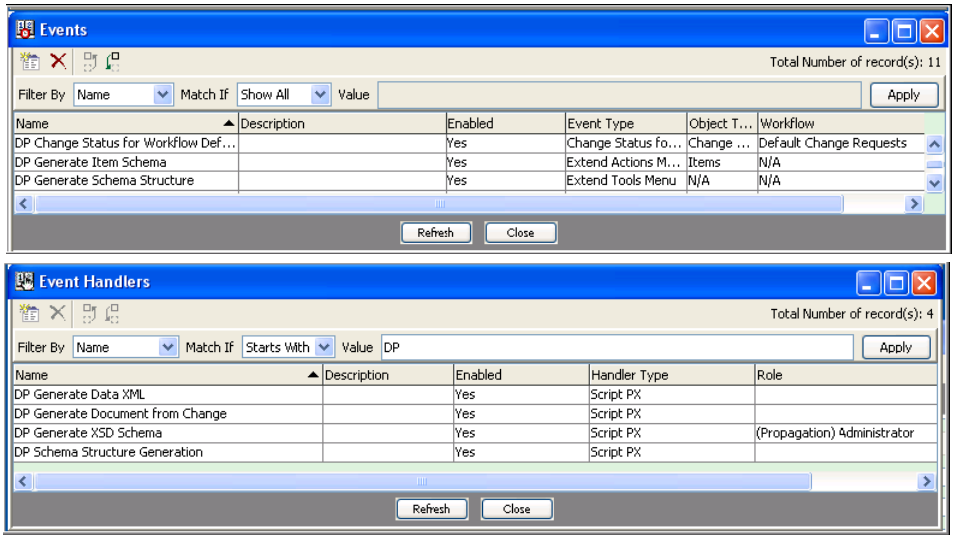


Figure 31 Document Publishing Events and Event Handlers

To run the PXs:

1. In Java Client (Admin client), select **Tools > DP Generate Schema Structure** to create the schema for the desired object.
2. In Java Client, select **Part - Action > Generate Schema**[12].
3. Trigger the Event that runs the Generate Data Handler.

- Make sure to update the Title Block of a Part in the Oracle-Supplied sample to trigger the PX to run.
- Make sure all Document Publishing attributes are correct before triggering this PX

Note: When triggered, this Event generates attachments for the Schema XSD and Data XML for use in Word with BI Publisher. Schema XSD is attached to a DocumentTemplate object, for example, "Assembly_2467572" where Assembly is the Item Type and 2467572 is the internal ID of the subclass.

[12] The PX does not rely on the Filter and generates the entire schema.

Configuring the DataGenerationPX

The purpose of the DataGenerationPX is to programmatically generate sample data using the Agile Java API Get XML Schema for document authors to preview the outputs in their selected authoring tool. When invoked, the PX creates and loads the XML file into the authoring tool (in this case, MS Word) to test the Template with BI Publisher[13]. As indicated in Creating JAR Files and Deploying PXs (Event Handlers), this PX requires creating the Item – Part/Document and setting values for Page Two attributes DocType, Filter and TemplateHolder.

Configuring Event Components for DataGenerationPX

These configurations are similar to the two preceding PXs. The Script PX or Java PX Event Handlers call the SDK Agile API to load Data for the object and add it as an attachment to the object.

To create Event mask and set Event Type:

1. Follow the steps in Configuring Event Masks for TemplateManagementStructureCreationPX and define an Event called Create Object for Object Type Part and the following settings:

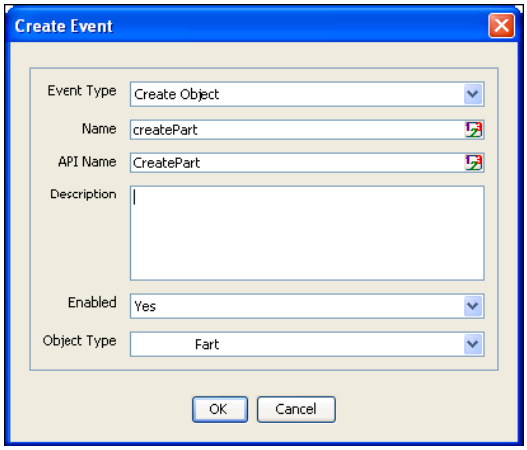


Figure 32 Create Event mask for DataGenerationPX

2. Click **OK**.

[13] This PX binds the Event to update the Title Block. This is not necessary because the Action menu Event alone will generate the required XML. Therefore, binding this PX to Create Items, leads to a recursive situation because this out of the box PX creates a document and attaches the XML file to the document.

To set up the Event Handler mask:

1. Use the information in Configuring Event Masks for TemplateManagementStructureCreationPX and create a new Event Handler mask (Script PX, or Java PX) called **part creation**.
2. Set Enabled to **Yes**, and for Role, select the applicable roles. For example, Quality Administrator, Quality Analyst, Quality Analytics User. For Event Handler Type, you have the option to select **Script PX**, or **Java PX**. You can find the Oracle-supplied Script and Java PXs in the Doc-Publishing folder in SDK_samples.zip.
3. Click **OK**.

To configure your Script PX or Java PX Handler Type do as follows.

- For Script PX Event Handler mask:, in **Create Event Handler**, paste the contents of Data Generation Groovy Script file in the dialog's Script box and then click **OK**. For more information, see Agile PLM Events and Event Framework chapter in *Agile PLM SDK Developer Guide - Developing PLM Extensions*. You can also find information on configuring Script PXs in *Agile PLM Administrator Guid*e. Following is an example of a Script PX Handle.

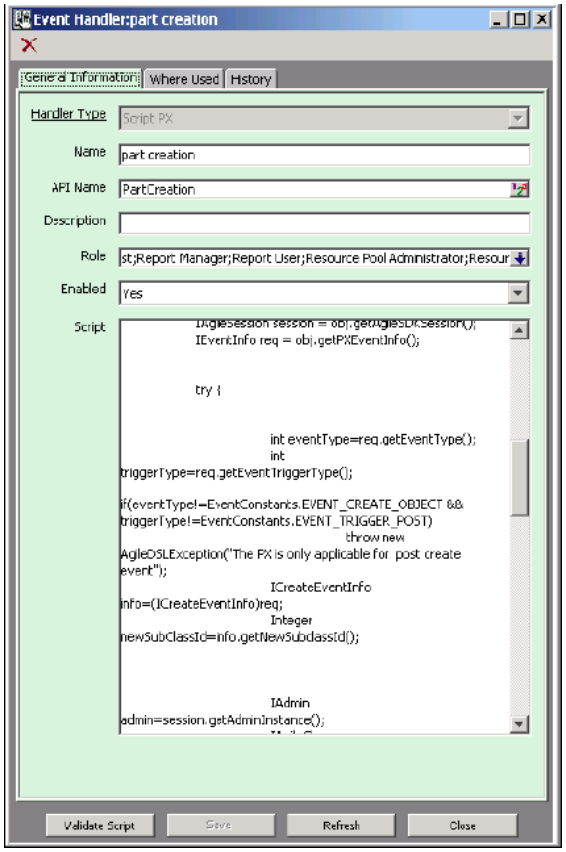


Figure 33 Script PX Handler for DataGenerationPX

- For Java PX Event Handler mask, make sure the Event Action for this Java PX is deployed, for procedures, see Creating JAR Files and Deploying PXs (Event Handlers) and check the values in DataGeneration.properties file and make sure they conform to Java Client Admin settings shown in "Properties File Settings for DataGenerationPX." .

To configure the Event Subscriber mask:

1. In Java Client with Admin privileges, select **Admin > System Settings > Event Management > Event Subscriber**s.
2. In Event Subscribers pane, select the **New** button to open the Create Event Subscriber dialog.
3. Create a new Event Subscriber called item creation with settings shown in the following figure.

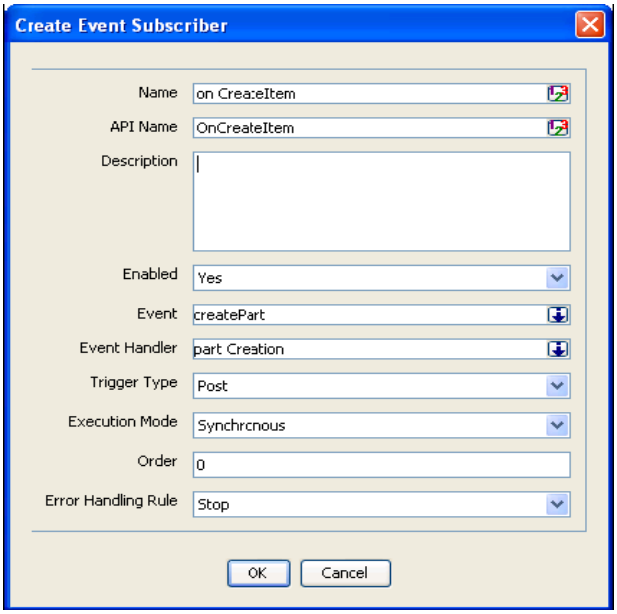


Figure 34 Creating an Event Subscriber for DataGenerationPX

4. Click the drop-down arrow to select the Event mask and Event Handler mask you created earlier.
5. Click **OK**.

Properties File Settings for DataGenerationPX

Values set in the Oracle-Supplied Properties file are shown in the shaded region of the following illustration. Make sure these values conform to Java Client Admin settings for this PX.

```
# TEMPLATE_SUBCLASS_API_NAME    -------------  Value for this property must be API name of a sub class of "Documents" class, objects of
#                                             which will be created to hold the XML data of an object.

# DATA_OBJECT_NUMBER            -------------  This property defines the constituents of the object number of the object which wil hold
#                                             the XML data in its attachment tab. Incase no value is specified fot this property, the
#                                             XML data will would be added an an attachment to the object from which the "Data Generation"
#                                             action is triggered.

# ACS_FILTER_ATTRIBUTE                       This property defines the nac  of the ACS(Agile Content Service) filter to be used while
#                                            genrating the XML data file.value for this should be API name or base id of the page two attr
#                                            which holds the filter name.

# DATA_FILE_NAME                -------------  This property defines the constituents of the file name of the XML data file.

# DATA_FILE_DESCRIPTION         -------------  This property defines the description for the XML data file.

# LOGGING                              -------------  This accepts true/false as value. When this is set to true log messages gets
#                                                     printed on the application server console

-------------------------------------------------------------------------------------------------------------------------------------------

# API_NAME                      -------------  Indicates API name of the  object's class for which XML data file is being generated

# CLASS_NAME                    -------------  Indicates class name of the object's class for which XML data file is being generated

# CLASS_ID                      -------------  Indicates class id of the  object's class for which XML data file is being generated

# OBJECT_NUMBER                 -------------  Indicates number of the object for which XML data file is being generated
-------------------------------------------------------------------------------------------------------------------------------------------
# Note: No property should be removed from this property file
TEMPLATE_SUBCLASS_API_NAME=DocumentTemplate

DATA_OBJECT_NUMBER="AgileData"+OBJECT_NUMBER

ACS_FILTER_ATTRIBUTE=1302

DATA_FILE_NAME="AgileData"+"_" +API_NAME+ ".xml"

DATA_FILE_DESCRIPTION="Agile Data for "+OBJECT_NUMBER

LOGGING=true
```

Figure 35 Properties file settings for DataGenerationPX

For Java PX:

1. Copy “DataGenerationPX.jar” to “<AgileHomeDir>\integration\sdk\extensions”.
2. Unpack “DataGenerationPX.jar” to gain access to “DataGeneration.properties” file.
3. Update as needed.
4. Repack and redeploy to PLM server.

For Script PX:

1. Open the Handler in PLM client.
2. Configuration is at the beginning of the Script. Modify the Script as needed.
3. Save the modified Handler.

Modifying the DataGenerationPX Script

The Sample creates a Document and then attaches the XML file to the new document. A better behavior is to simply attach the XML file to the source object, especially when dealing with processes such as Problem Reports.

To change this behavior, modify the script as shown in the bold font blow.

```
try {
   String TEMPLATE_SUBCLASS_API_NAME="DocumentTemplate";
   String DATA_OBJECT_NUMBER="OBJECT_NUMBER";
   String DATA_FILE_NAME=" \"AgileData\" + \"_\" +API_NAME+\".xml\"";
   int ACS_FILTER_ATTRIBUTE=1302;
   String DATA_FILE_DESCRIPTION=" \"AgileData for \"+OBJECT_NUMBER";
   ITable attachmentTable =null;
   IAgileObject agileObject=null;
   String msg="";
```

Output Generated by DataGenerationPX

When triggered from the Actions menu, the PX will perform the following:

1. Gets the current object data using the Agile SDK.
2. Gets the Template BOM ID, filter ID, and output format using Page 3 attributes in the property file.
3. Creates a Document and attaches the XML file to the new document.

The output of the PX is an XML file. The naming convention for the Data XML file is <ObjectSubclassName>:<ObjectName>:<Rev>:<DataSuffix>.<XML>.

These attributes are defined as follows:

- **ObjectSubClassName** – This is the name of the Subclass. For example, Documents.
- **ObjectName** – This is the instance of the Object. For example, D000001.
- **Rev** – This is the Revision name/number.
- **DataSuffix** – This is set by the user in the Properties file.

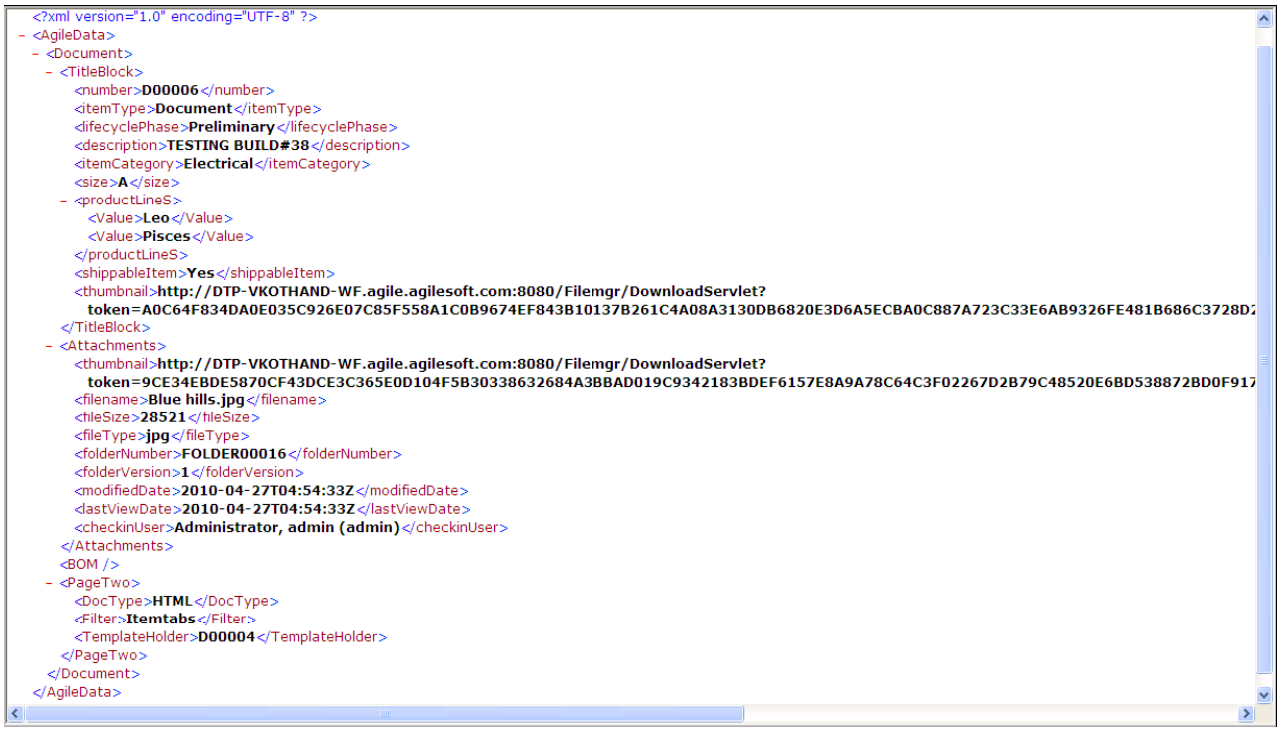


Figure 36 Document:D00006:A:Data.XML

Building BI Publisher Templates

To build a template, you need the Schema XML and Data XML files. For Doc-Publishing purposes, these are the files that are generated by invoking Web Services `loadXMLSchema` and `loadXMLData` APIs.

To configure the template

1. In the BI Publisher menu, select **Insert > Field**.
   This opens a BI Publisher screen that lists all available fields from the Agile PLM Schema previously loaded using their API Names.
2. In the Field selection dialog, point to the field of interest and using Insert, add them in the order that you want them to appear in the resulting document.
3. Scroll through the list, or use **Find Next** to select fields, for example, **CreateUser**.

4. Using Word features, customize the fonts and other formats for the inserted tags.

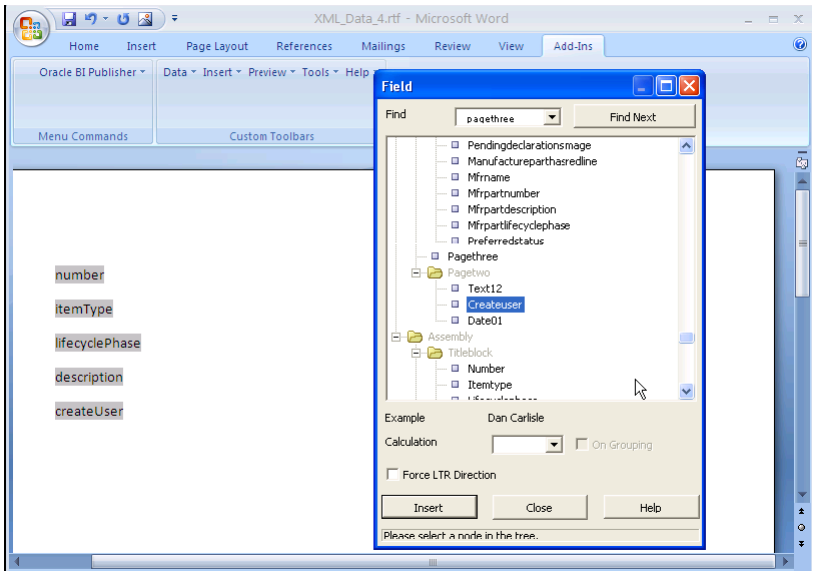


Figure 37 Building the BI Publisher Template

5. When you complete the layout, save your Template as an RTF file in the local drive.
6. From the BI Publisher menus, select **Preview Template > PDF** (or any format) to see the Data formatted in your Template.

Copying XSD and XML Files to the Local Drive

In Agile PLM Web client, search by document name and the click the **Attachments** tab.

To copy XSD and XML files to the local drive:

1. Select the XSD file and either click **Get**, or double-click on the file.

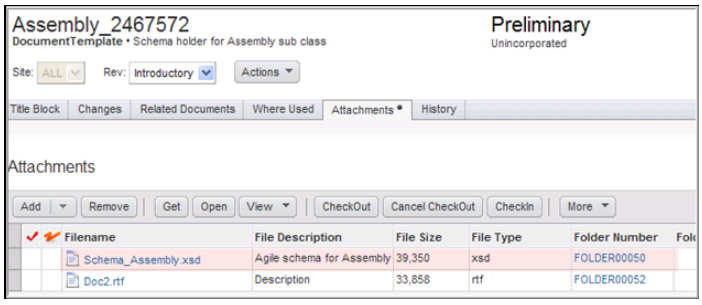


Figure 38 Downloading XSD file to local drive

2. Select the applicable Download method and then save the file to the local drive.
3. Repeat the process for the XML Data.

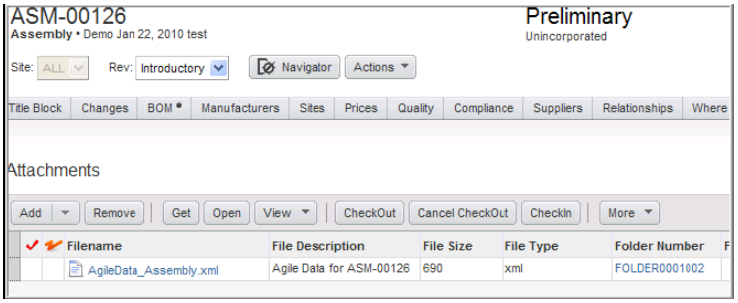


Figure 39 Copying the XML file to local drive

Loading Schema XSD and Data XML Using BI Publisher

The following procedure assumes that you have already:

- Installed BI Publisher Desktop on your system
- Ran the SchemaGenerationPX and DataGenerationPX and created the Schema XML and Data XML files.
- Ran FileManagementSetup.msi (The Word Plug in Installer)

To load the Schema XSD and Data XML files:

1. Open the document that you want to generate. For example, a data sheet containing text that is describes and is not subject to change and variable (data) such as Part Number, Date, and so on that you want to update with Agile PLM data for publication.
2. Save the Word document as an .RTF file and then load the XML Schema and XML Data files.
3. Open Microsoft Word and select **Add-Ins > Data**.

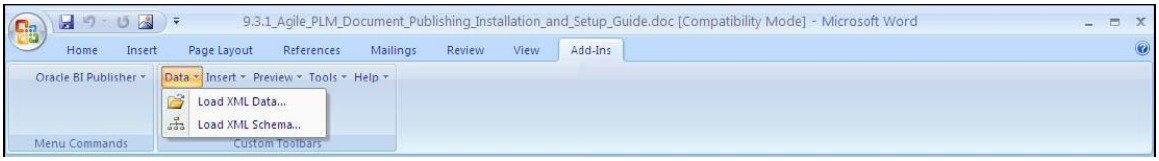


Figure 40 Load XML Schema and XML data

4. Select Load **XML Data...** and then **Load XML Schema...**to load the files.

   Word will display "Data loaded successfully" after each action

Loading the files enables BI Publisher to access Agile PLM fields in the XML file that were defined earlier for the Subclass. For example, for "Documents" subclass defined in Creating a Placeholder for Template Files, you can use all features of Word with BI Publisher to create a template for the data sheet.

Selecting Agile PLM Data Fields and Formatting the Template

BI Publisher facilitates selecting Agile PLM data fields and provides extensive facilities to format the data and output document.

To configure the template and inserting Agile PLM data in the Template

1. In the BI Publisher menu, select **Insert > Field**.
   This opens a BI Publisher screen that lists all available fields from the Agile PLM Schema previously loaded using their API Names.
2. In the Field selection dialog, point to the field of interest and using Insert, add them in the order that you want them to appear in the resulting document.
3. Scroll through the list, or use **Find Next** to select fields, for example, **CreateUser**.
4. Using Word features, customize the fonts and other formats of these inserted tags.

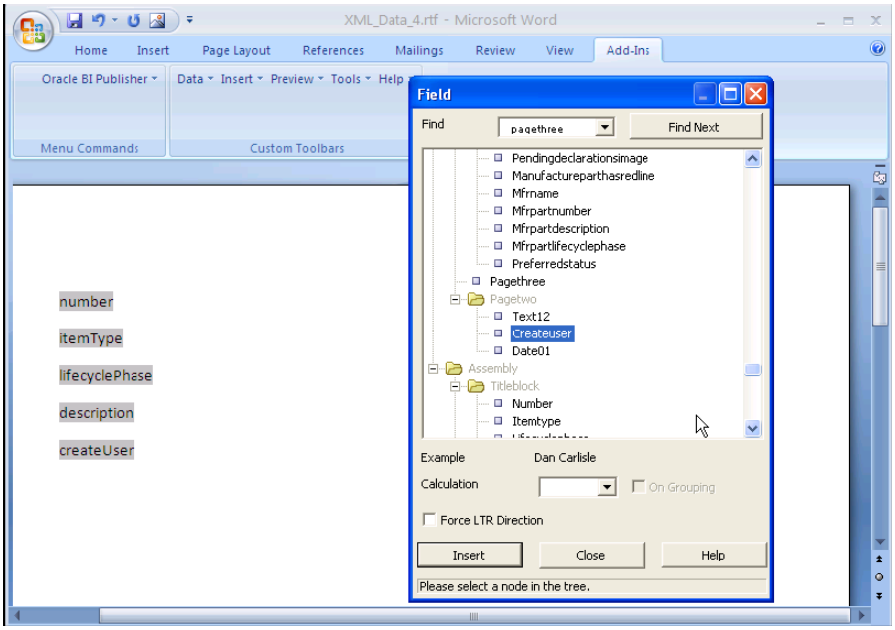


Figure 41 Building the BI Publisher Template

5. After completing the layout, save your Template as an RTF file.

6. From the BI Publisher menus, select **Preview Template > PDF** (or any format) to preview the Data formatted in your Template.

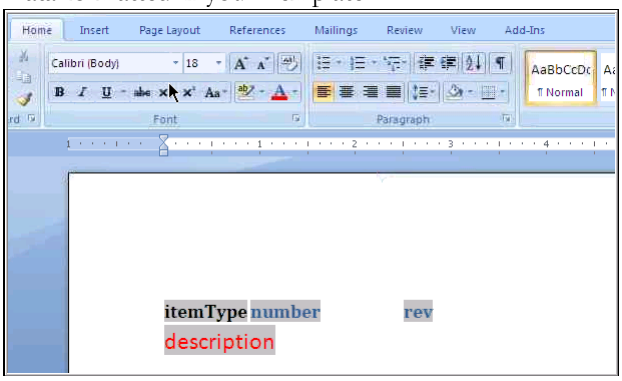


Figure 42 Viewing the template

Inserting and Formatting Tables

Using BI Publisher, you can insert and represent Agile PLM fields in a tabular form. BI Publisher's Table formatting combined with Word, provide rich formatting capabilities, for example, generating totals for numeric fields in columns or rows. For more information, see BI Publisher document in listings in Cited References.

To present Agile data in tabular format:

1. In Word with BI Publisher Desktop, select **Add-Ins > Insert > Table Wizard**. The Wizard prompts you to select the grouping fields that you want to report on.

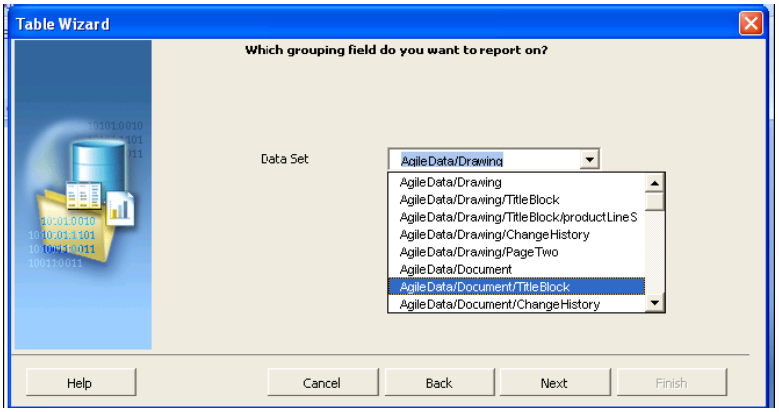


Figure 43 Formatting Agile data in tabular format

2. Select the applicable group, for example, **AgileDocumentTitleBlock**.

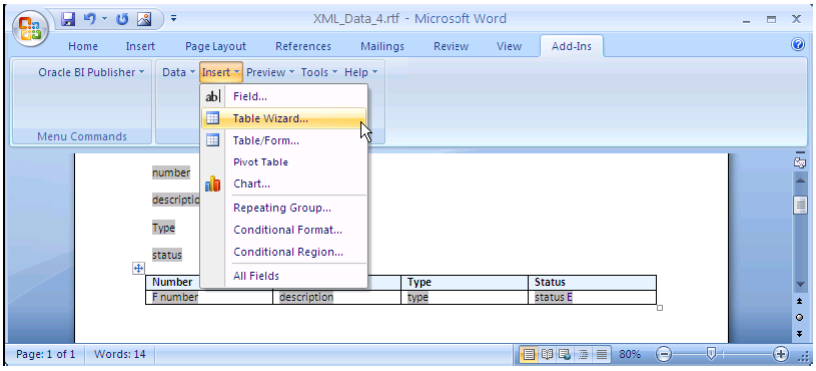


Figure 44 Displaying Agile data in tabular format

3. Select the fields and then format the table.

Inserting Images and Charts in Templates

BI Publisher supports several options for adding images in a published document. These options require including the image files in the document Template.

These options are:

- Direct insertion
- Using a URL Reference
- Referencing Elements in XML Files

To directly insert an image or chart:

Similar to inserting images or charts in Word documents, you can simply insert or paste JPG, GIF, or PNG images directly in the RTF Template.

To insert an image using a URL reference:

1. Insert/paste an image in the Template file. This is used to access MS Word's Picture Format dialog box.
2. Depending on the version of Word that you are using do as follows to open the **Alternative text** box:
   - In Word 2007, right click the image and select **Format Picture > Alt Text**.
   - For earlier versions of Word, right click the image and select **Format Picture ...** in the drop-down list and then select the **Web** tab.

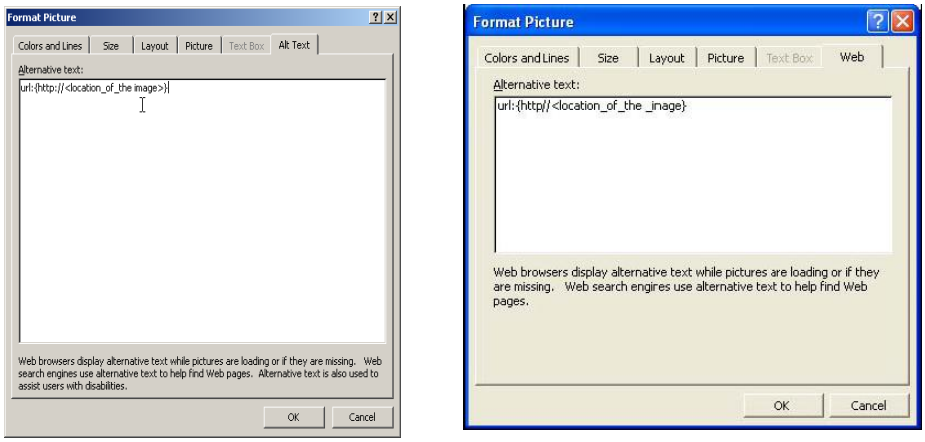


Figure 45 URL referencing in Word 2007 (on the left) and earlier versions on the right.

3. In Alternative text box, type the URL that is pointing to the location of the image that is using this syntax: url:{'http://<location_of_the_image>'}. For example, url:{'http://www.oracle.com/images/ora_log.gif"}

To insert an image using a URL reference:

3. Insert/paste an image in the Template file. This is used to access MS Word's Picture Format dialog box.
4. Depending on the version of Word you are using do as follows to open the **Alternative text** box:
   In Word 2007, right click the image and select **Format Picture > Size > Alt Text**

To reference an element in an XML File:

1. Similar to inserting an image using a URL reference, insert/paste an image in the Template file.
2. Open the Alternative text box as you did in "To insert an image using a URL reference:" above.
3. In the Alternative text box, type the path to the image, using this syntax url:{IMAGE_LOCATION}. IMAGE_LOCATION is an element in the XML file that holds the full URL to the image.
   By using the `concat` function to build the URL string, you can build a URL based on multiple elements at runtime. For example, url:{concat(SERVER,'/',IMAGE_DIR,'/',IMAGE_FILE)}, where SERVER, IMAGE_DIR, and IMAGE_FILE are element names in the XML file that holds the values to construct the URL.

Inserting Thumbnails into Templates

Similar to images, you must also furnish information about Thumbnails in the document Template as shown in the following figure.

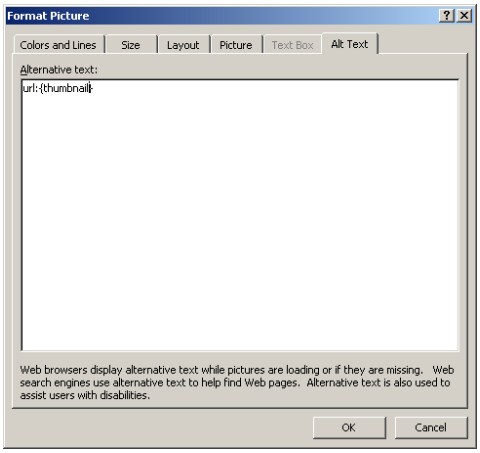


Figure 46 Adding Thumbnails to Templates

Loading the Template into Agile PLM

This is done using Web client's Add function as shown below.

To load the Template into Agile PLM:

1. Log in to Agile PLM and select the folder you want to load the file into. In this case, DOCUMENT_TEMPLATE that you defined earlier.
2. Select **Attachments > Add**. The Add Files dialog opens.
3. In Add Files dialog, use **Browse** to locate the file on the local drive and then click the ✔ Add

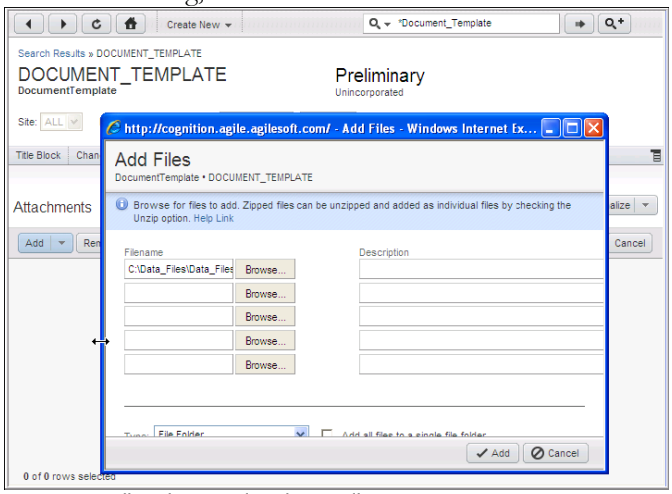


Figure 47 Loading the template into Agile PLM

4. Repeat the process to load other files.

Configuring DocumentGenerationPX

Document Generation provides the following options to publish a document:

- DocumentGenerationJavaPX – This PX generates a file based on a Template and using BI Publisher
- DocumentGenerationJavaPXOpen – This PX opens the document instead of saving it as an attachment
- DocumentGenerationWS PX - The purpose of this Document generation PX is to programmatically generate documents using the Document Publication engines such as BI Publisher.

Configuring DocumentGenerationJavaPX

The purpose of the Oracle-Supplied Document Generation PX is to programmatically generate a file based on a Template and a PLM object and use BI Publisher as the Document Publication engine to publish the file/document.

As prerequisite, this PX requires an object number for the TemplateHolder attribute, for example, P00001, or P00021.

Configuring Event Masks for DocumentGenerationJava PX

Similar to the preceding PXs, you must create an Event and set Event Type, Event Handler, and Event Subscriber for the Java or Script PX. Upon the release of an ECO, the PX loads all items from the BOM tab and will Generate Document from each BOM item using the Agile embedded BI Publisher.

To create Event mask and set Event Type:

1. Follow the steps in Configuring Event Masks for TemplateManagementStructureCreationPX and define an Event called CreateDocument for Object Type Change Requests with the following settings:

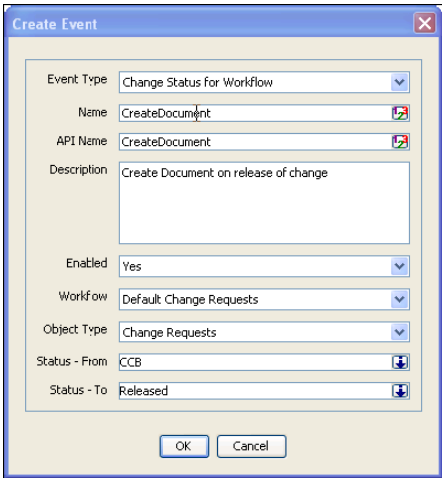


Figure 48 Configuring Event type for DcocumentGenerationJavaPX

2. Click **OK**.

To set up the Event Handler mask:

1. Use the information in Configuring Event Masks for TemplateManagementStructureCreationPX and create a new Event Handler mask (Script PX, or Java PX) called GenerateDocument.
2. Set Enabled to **Yes**, and for Role, select the applicable roles. For example, Quality Administrator, Quality Analyst, Quality Analytics User. For Event Handler Type, you have the option to select **Script PX**, or **Java PX**. You can find the Oracle-Supplied Script and Java PX in Doc-Publishing folder in SDK_samples.zip.
3. Click **OK**.

To configure your Script PX or Java PX Handler Type do as follows.

- For Script PX Event Handler mask, in Create Event Handler, paste the contents of Document Generation Groovy Script file in the dialog's **Script** box and click **OK**. For more information, see Agile PLM Events and Event Framework chapter in *Agile PLM SDK Developer Guide - Developing PLM Extensions*. You can also find information on configuring Script PXs in *Agile PLM Administrator Guide*. Following is an example of a Script PX Handler and then click **OK**.

.

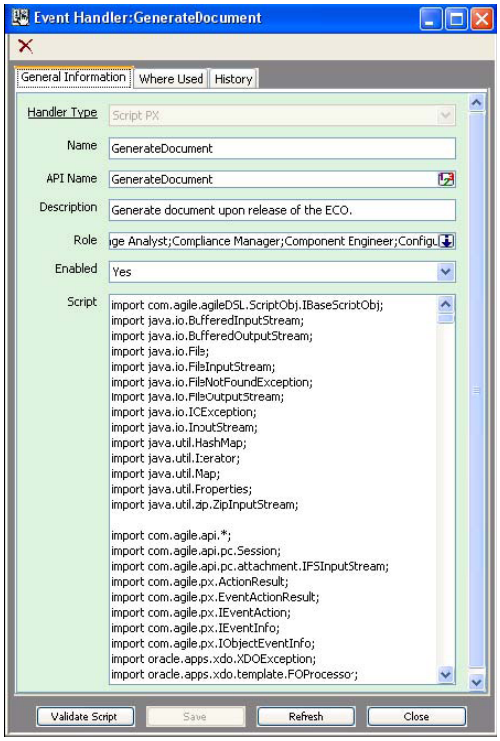


Figure 49 Script PX Handler for DocumentGenerationPX

- For Java PX Event Handler mask:, make sure the Event Action for this Java PX is deployed. See Deploying PXs (Event Handler masks) and then check the values in DocumentGeneration.properties file and make sure they conform to Java Client Admin settings shown in Properties File Settings for DocumentGenerationJavaPX. And then click **OK**.

To configure the Event Subscriber mask:

1. In Java Client with Admin privileges, select Admin > System Settings > Event Management > Event Subscribers.
2. In **Event Subscribers** pane, select the **New** button to open the Create Event Subscriber dialog.
3. Create a new Event Subscriber called **GenerateDocument** with settings shown in the following figure.
4. Click the drop-down arrow to select the Event and Event Handler you created earlier.

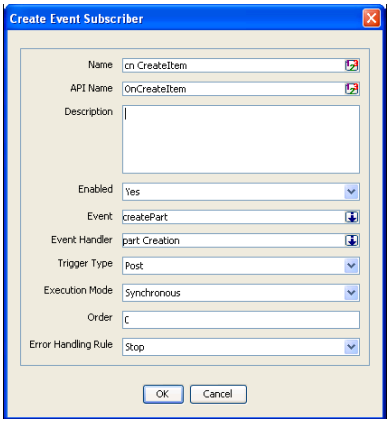


Figure 50 Event Subscriber settings for DataGenerationPX

5. Click **OK**.

Properties File Settings for DocumentGenerationJavaPX

Values set in the Oracle-Supplied Properties file are shown in the shaded region of the following illustration. Make sure these values conform to Java Client Admin settings for this PX..

```
# API_NAME                        = API name of the class in the object number
# CLASSNAME                       = name of the class to the object number
# CLASSID                         = class id to the object number
# REV_NUMBER                      = rev number of item.


#report type.. Ex pdf,html...
ATT_DOCUMENT_TYPE                 =1271

#Attribute which has templete
ATT_TEMPLATEHOLDER                =1302

#name of the filter
ATT_FILTER                        =1301

#Genenrated document file.
DOCUMENT_FILENAME                 = OBJECT_NAME + "_" + REV_NAME

#Genenrated document file desc.
DOCUMENT_FILENAME_DESC            = "Document of the Item" + OBJECT_NAME + " Change " + REV_NAME

#Document template subclass.
TEMPLATE_SUBCLASS_API_NAME        = DocumentTemplate

#target object, where report will be saved
TARGET_DOCUMENT_NAME              = "DOCUMENT " + OBJECT_NAME + "_" + REV_NAME


#to enable logging
loging                            = true

****************************
Settings for JavaOpen PX
****************************

OPEN_REPORT                       = true

IS_CHANGE_OBJECT                  = true

FILESERVER_URL                    =http://<systempath>:8089/URLPX/PX
```

Figure 51 Properties file settings for DocumentGeneration Java PX and JavaOpen PX

Output Generated by DocumentGenerationJavaPX

When the PX is triggered upon the release of an ECO, it will:

1. Load the Affected Items Tab of the change and calls the SDK to get the data for the Affected Item
2. Use the settings in the Properties file for the following P2 Item attributes:
   - TemplateID
   - ACS filter name
   - Document Type
3. Load the template using the P2 Document Number attribute for the Item.
4. Call the SDK to load the data for BOM items.
5. Call BI publisher and pass the data Template to generate the document.
6. Save the document along the naming convention for the generated file in the Attachments Tab. The PX creates a separate document object and attaches the output file to this object.

The naming convention for the generated file is <ObjectSubclassName> :<ObjectName> :<Rev>:<templateID><documentSuffix>.<documentType>.

These attributes are defined as follows:

- **ObjectSubClassName** – This is the name of the Subclass. For example, Documents.
- **ObjectName** – This is the instance of the Object. For example, D000001.
- **Rev** – This is the Revision name/number.
- **TemplateID** – This is the template name.
- **DataSuffix** – This is set by the user in the Properties file.
- **DocumentType** – This is the format of the output file. Options are PDF, EXCEL, HTML, RTF and PowerPoint.

When creating Events, Event handlers and Event subscribers, you must enable the Event by clicking the **enable** button in Java Client to see the Events in their respective actions. If Events are disabled, you cannot see the Events under their respective actions.

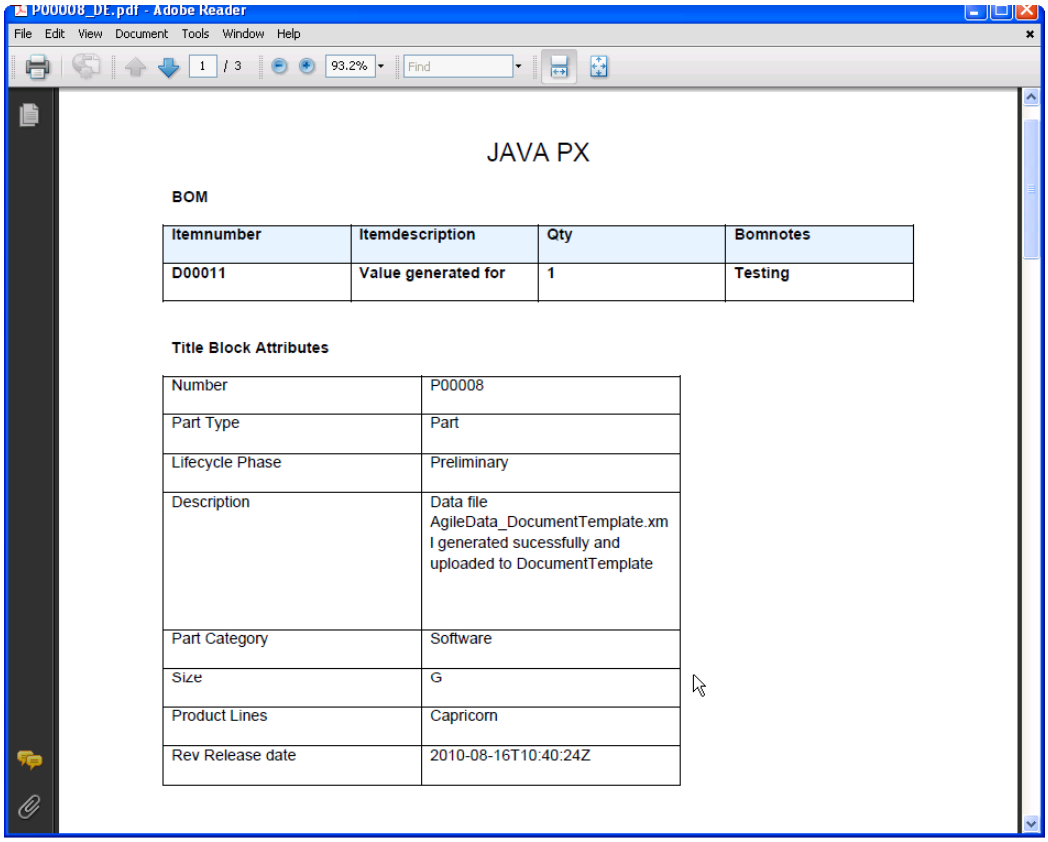


Figure 52 DocumentGenerationPX output in PDF format

Configuring DocumentGenerationJavaOpen (URL PX)

The DocumentGenerationJavaPxOpen or URL PX, instead of saving the document as an attachment, displays the output generated by the DocumentGenerationJavaPX in the URL that you specified in Java PX's Properties file, or in Script PX's Groovy script. For procedures, see To setup the URL PX:.

To setup the URL PX:

1. Unzip URLPX.zip to tomcat directory at tomcat\webapps.
   After unzipping, you will see the URLPX directory in tomcat\webapps.
2. Edit the tomcat\webapps\URLPX\WEB-INF\ web.xml by changing the http://shahdesk-dgx520.agile.agilesoft.com:8888/web to your application server hostname.
3. Type the correct value for FILESERVER_URL in Tomcat\webapps\URLPX\WEB-INF\classes\samples\DocumentGeneration\DocumentGenerationJavaPxOpen\ DocumentGeneration.properties.
4. In Java Client, with Admin privileges, select Admin > Data Settings > Process Extensions.
   The Process Extension Library panel opens.
5. In Process Extension Library, select the New button to open and configure the Add Process Extension dialog as shown in the following figure.
   The Address field should point to the Filemanager. For example, http://<filemgerHost>:<FilemgerPort>/URLPX/PX.

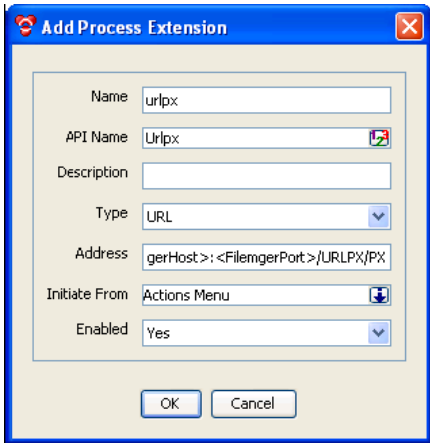


Figure 53 URL PX settings in Add Process Extension dialog

6. Click **OK**.

Configuring the Properties file for DocumentGenerationJava OpenPX

1. Navigate to the respective object, for example, **Items > Documents class**.
2. Navigate to Process Extensions tab of that Document class and add this URL PX which is already created.
3. Use these steps for other objects of interest.

DocumentGenerationJavaPxOpen Output Sample

When the PX is invoked from the Actions Menu, it will open the document in the specified URL in HTML format.

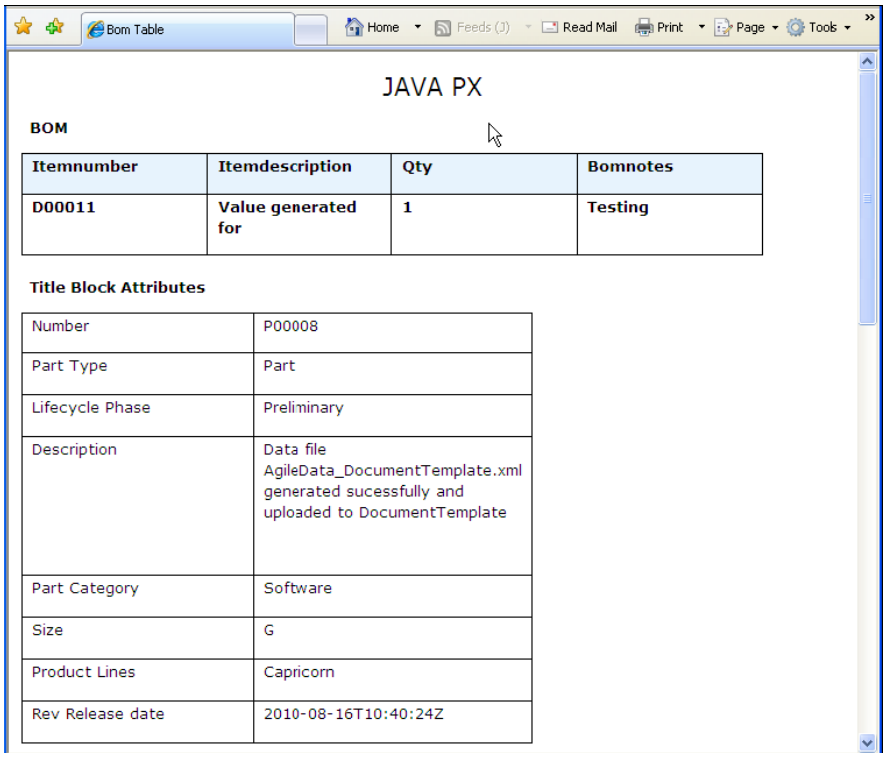


Figure 54 Output generated by URL PX

Triggering DocumentGenerationPX

On releasing the trigger (for example, a change), this PX generates a report with the file extension defined in DocType and attaches it to the object specified in TemplateHolder. Because it is a prerequisite, it requires assigning an existing object number for the TemplateHolder attribute. The required settings are defined in "Performing Agile PLM Administrator Configurations" and "Configuring DocumentGenerationJavaPX".

Modifying DocumentGenerationPX

The Sample creates a Document object and then attaches the output file to the new document. Oracle recommends attaching the output file to the source object, especially with processes such as Problem Reports. Be sure to specify the correct location of the Template because getting the Template retrieves the first file from the specified object.

To modify the PX, for example, to change the name of the document from Document name, to Document and Object name, you must modify the PX's Properties file as shown below.

```
Public ITem getTargetObject(IAgileSession session, Item object)throws
Exception{
    private static final String TEMPLATE_SUBCLASS_API_NAME =
    "documentTemplate";
    private static final String TARGET_DOCUMENT_NAME = "DOCUMENT +
    OBJECT_NAME";
```

Triggering the Event and Creating the Output File

To trigger the Event and generate the sample document do as follows:

1. Make sure the object that you want to process has the correct configuration for the Template, Filter, and output file type in Script PX Code, or Java PX Properties file. For the example, the settings are as shown below.
   - Output Type = PDF
   - ACS Filter = Itemstabs
   - Template Holder = D-00004
2. Trigger the Event (release the ECR).

Figure 55 Releasing the ECO to trigger the Event

3. Open the generated document and view the output file.

Figure 56 Generating the output file

Configuring the DocumentGeneration WebService PX

The purpose of this Document generation PX is to programmatically generate documents using Document Publication engines such as BI Publisher. This PX gets the necessary dats from Agile PLM and generates the document in Agile Java PX using Agile bundled BI Publisher (engine).

Steps in the document generation process:

1. Get current object data from Agile PLM.
2. Generate document in PX using BI Publisher Web Services APIs.
3. Add generated documents to the object attachment table.

Properties File Settings for DocumentGeneration WebService PX:

```
# API_NAME                              = API name of the class in the object number
# CLASSNAME                     = name of the class to the object number
# CLASSID                               = class id to the object number
# REV_NUMBER                    = rev number of item.


#report type.. Ex pdf,html...
ATT_DOCUMENT_TYPE               =1271

#Attribute which has templete
ATT_TEMPLATEHOLDER              =1302

#name of the filter
ATT_FILTER                                      =1301

#Genenrated document file.
DOCUMENT_FILENAME               = OBJECT_NAME + "_" + REV_NAME

#Genenrated document file desc.
DOCUMENT_FILENAME_DESC          = "Document of the Item" + OBJECT_NAME + " Change " + REV_NAME

#Document templete subclass.
TEMPLATE_SUBCLASS_API_NAME  = DocumentTemplate

#target object, where report will be saved
TARGET_DOCUMENT_NAME            = "DOCUMENT " + OBJECT_NAME + "_" + REV_NAME

#to enable logging
loging                          = true

#Username to loging to BI server
BI_SERVER_LOGIN_USERNAME    =Administrator

#password to loging to BI server
BI_SERVER_LOGIN_PASSWORD        =Administrator

#Report absolute path.
REPORT_ABSOLUTE_PATH            =/Boilerplates/chen932/chen932.xdo

# BI server URL
BI_SERVER_URL                   = http://dineshp.agile.agilesoft.com:9704/xmlpserver/services/PublicReportService

OPEN_REPORT                     = true

IS_CHANGE_OBJECT                =false

FILESERVER_URL                  =http://blr2230078.agile.agilesoft.com:8080/Filemgr
```

Creating RTF Templates and Updating Data Fields

This example shows how to create and format Word RTF templates using Oracle BI Publisher and accessing PLM from Word to update the document that uses this template.
This example makes the following assumptions:

- You are using Oracle BI Publisher Version 10.1.3.4, can log in to BI Publisher, and have the necessary privileges to use its Template management commands.
- You have performed all Administrator and Server configurations described earlier.

Logging In to BI Publisher and Loading the XSD and XML Files

In Word 7, do as follows:

1. Click **Add-Ins > Oracle BI Publisher > Log On**.

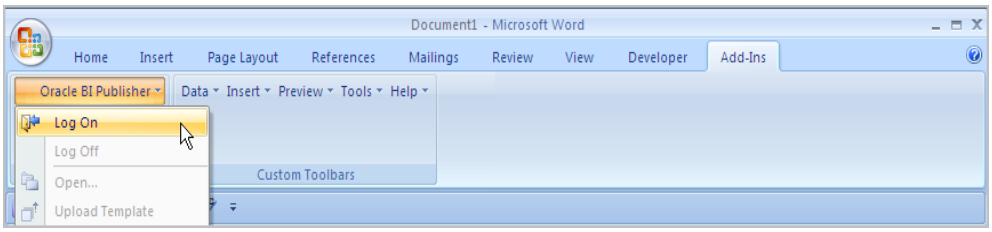


Figure 57 Accessing BI Publisher from Word Add-Ins

The Oracle BI Publisher 10.1.3.4 Login Screen opens in MS Word.

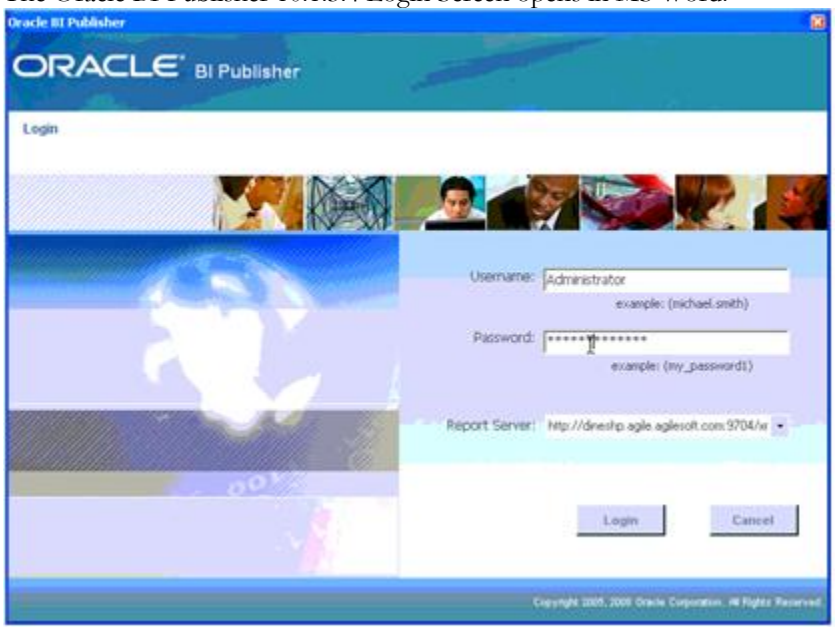
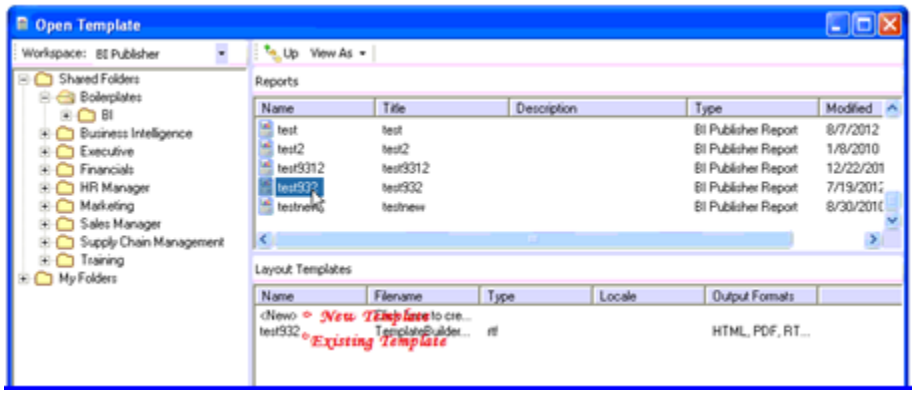


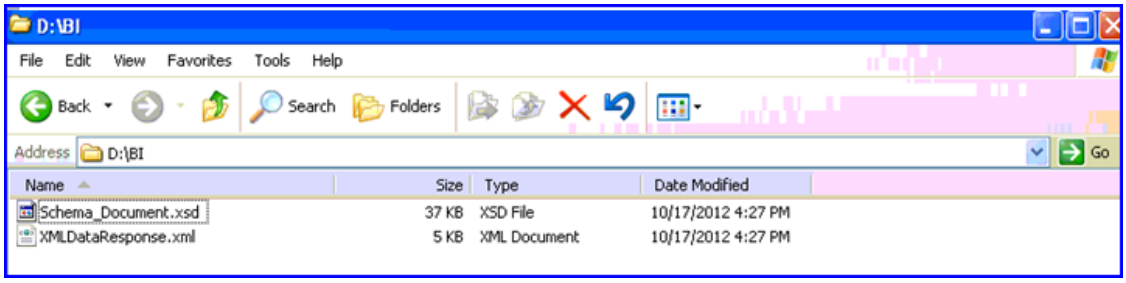
Figure 58 Oracle BI Publisher Login dialog

Log in using your credentials and the **Open Template** dialog appears.

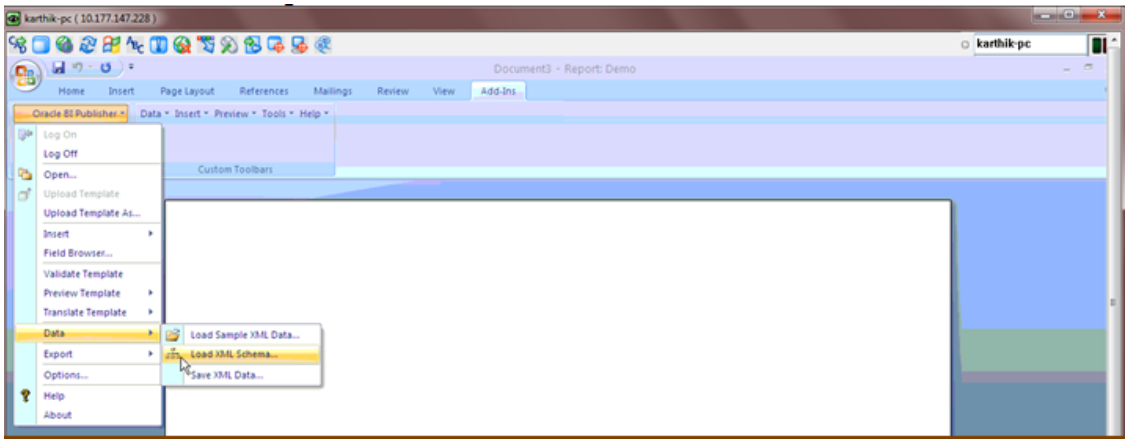
2. In Open Template dialog, select **Shared Folders > Boilerplates** and right-click an existing template. For example, **test 932**. Note that you have the option to open the existing RTF template and edit the template if you choose to do so, or create a new template by selecting **<New>**.



3. Create a folder called **BI**, copy the XSD file generated by `SchemaGenerationPX` and the xml file generated by `DataGenerationPX` and change the name of the xml file to `XMLDataResponse.xml`.



4. In Oracle BI Publisher drop-down menu, select **Data > Load XML Schema** (which invokes the Web Service API **loadXMLSchema**) to load the Schema file.



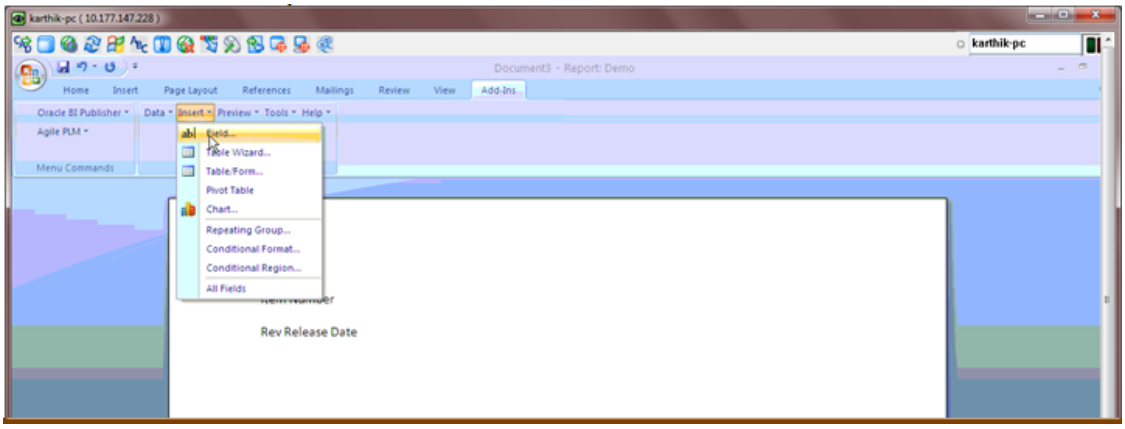
When loaded successfully, the following Word verification message appears.

Creating and Configuring Templates in Oracle BI Publisher

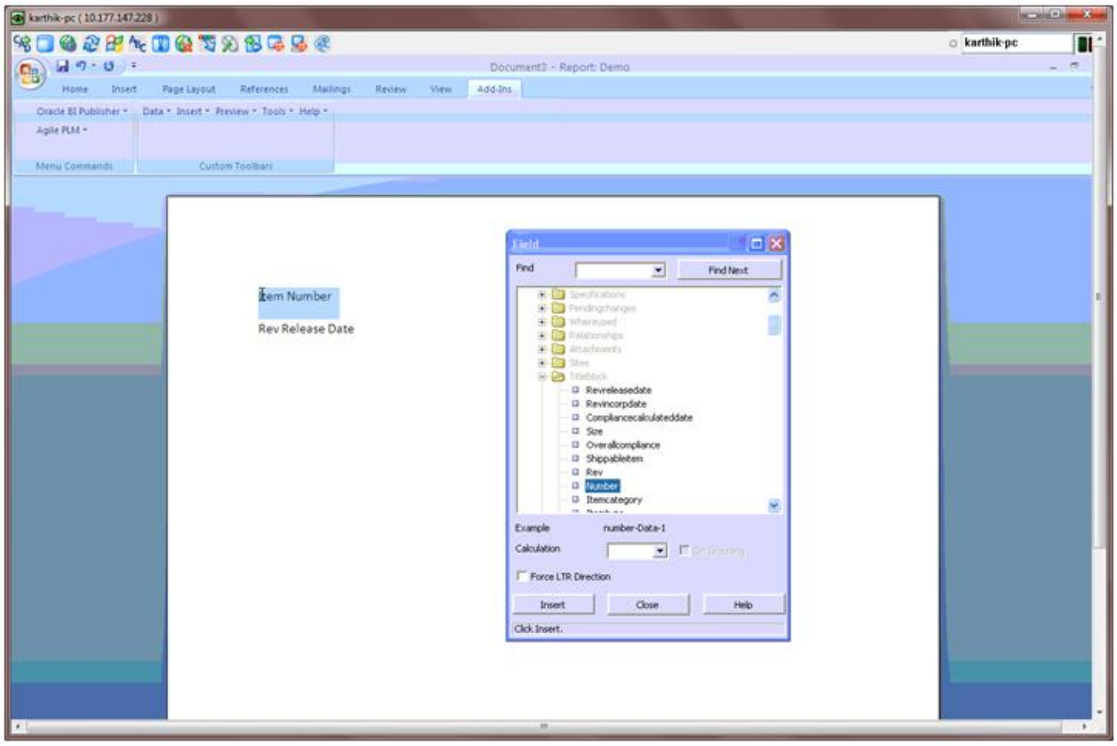
This section describes using BI's **Insert > Field** and BI's **Insert > Table Wizard** commands to map PLM data attributes into documents and format reports.

To map attributes using BI's Insert > Field:

1. In Word, select **Add-Ins > Insert > Field** as shown below to open BI's **Field** dialog.



2. In BI's Field dialog, select the applicable field and then click Insert.

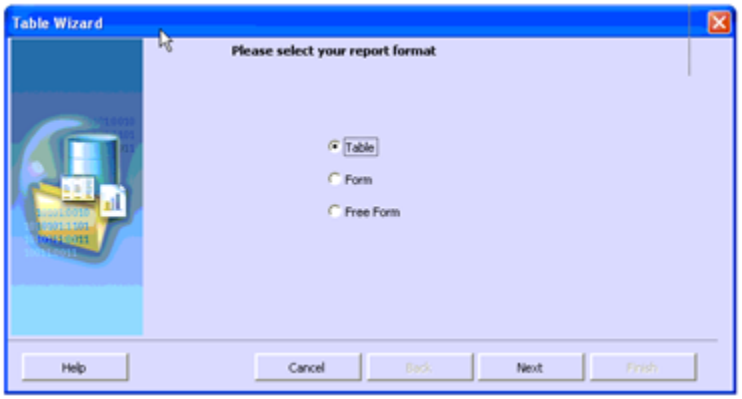


3. Repeat step 2, until you have included all the defined PLM data fields and then save the RTF template file.

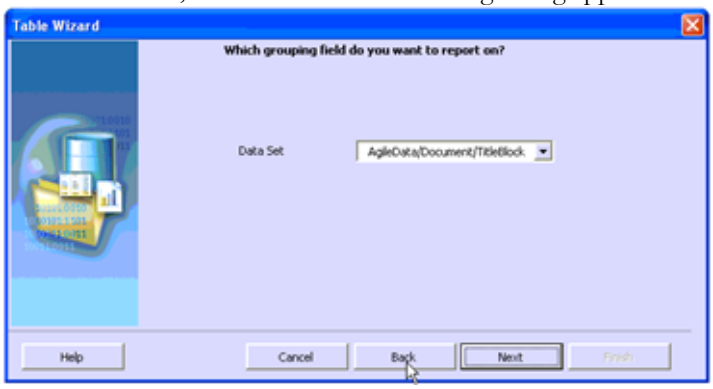
To map attributes using BI's Insert > Table Wizard's Table option:

1. In Word, select **Add-Ins > Insert > Table Wizard**.

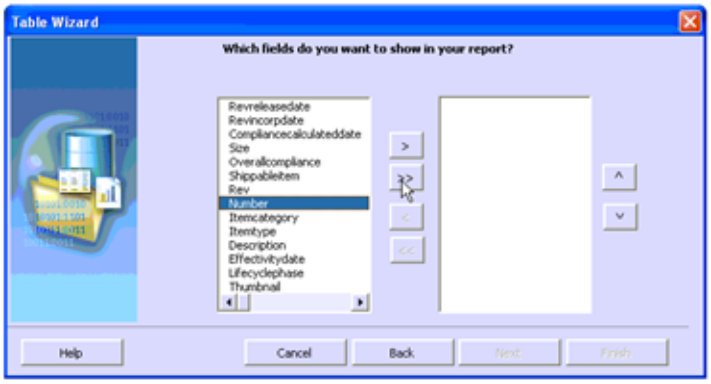
   The Table Wizard dialog appears. Note that you have the option to select Free Form, Form, and Tabular formats.



2. In Table Wizard, select **Table**. The following dialog appears.

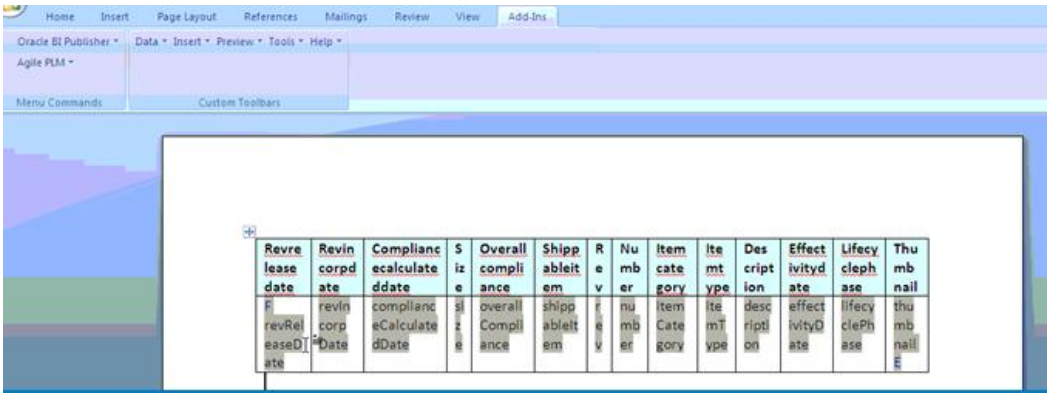


3. In the **Data Set** drop-down list, select the applicable data set. For example, **AgileDataDocument/TitleBlock**. Table Wizard displays available fields in the Data Set.



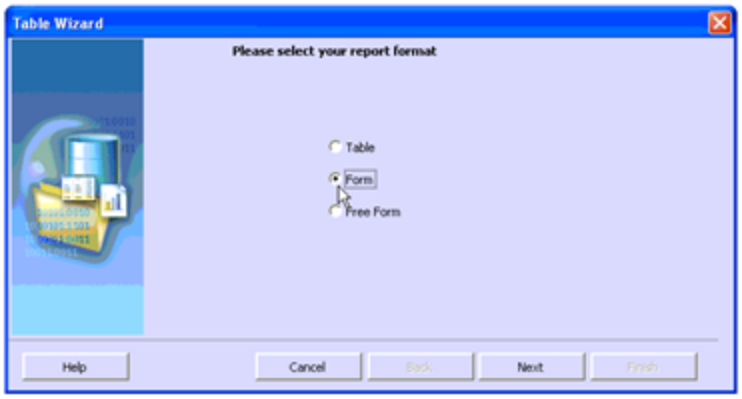
4. Select the required fields from the available list and using the **>** button, move them to the selected list.

5. In Word, use **Add-Ins > Oracle BI Publisher > Preview Template > RTF** to view the RTF template file.

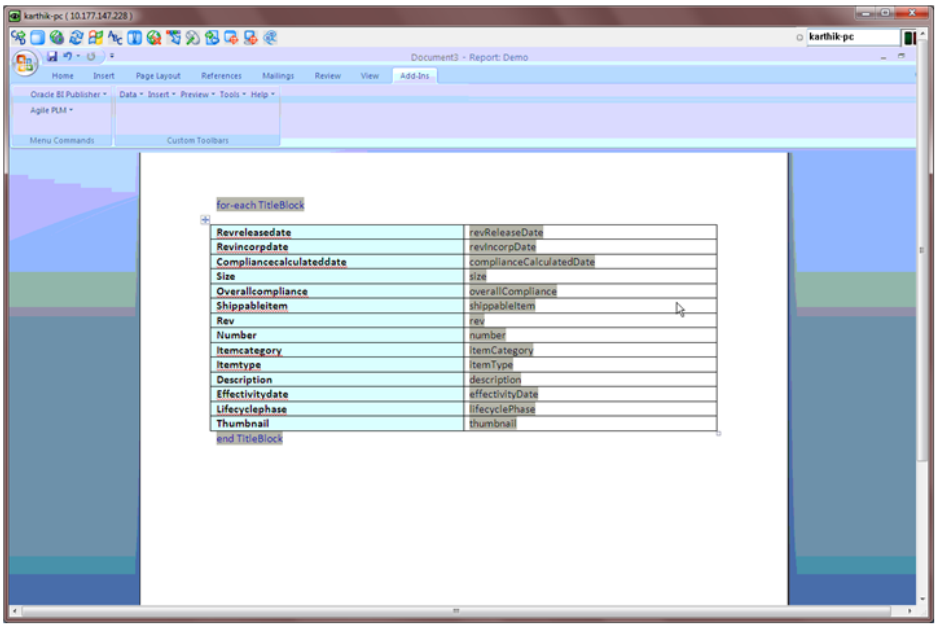


To map attributes using BI's Insert > Table Wizard into a Form option:

1. In Word, select **Add-Ins > Insert > Table Wizard** and then select **Form** in Table Wizard.

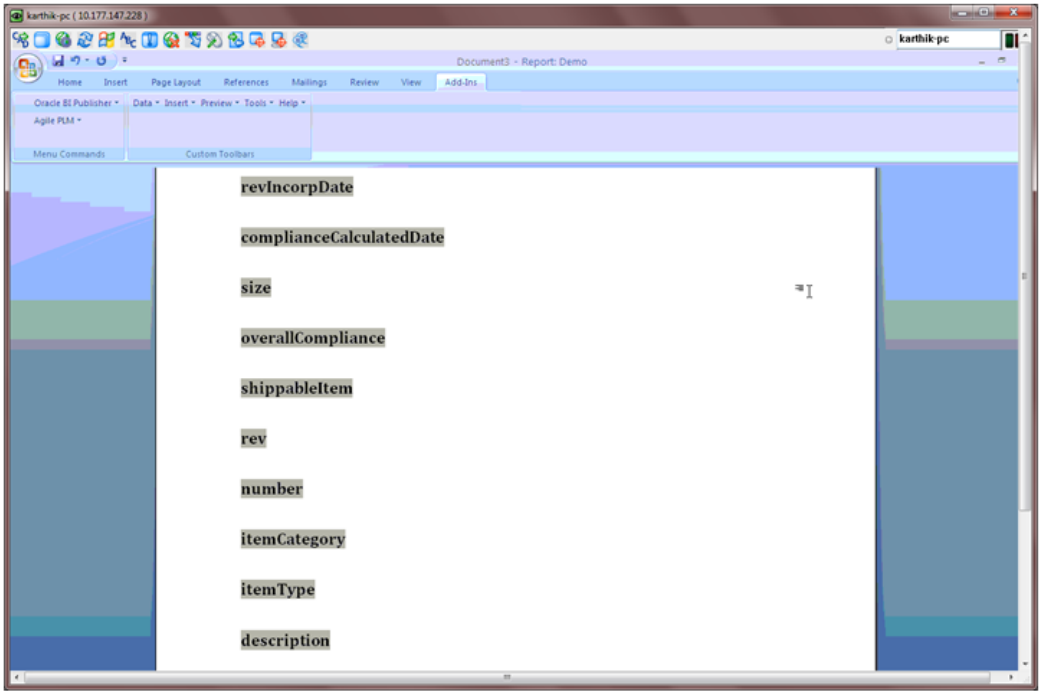


2. Select the required fields as you did in To map attributes using BI's Insert > Table Wizard's Table option: and view the Template by selecting **Add-Ins > Oracle BI Publisher > Preview Template > RTF**.



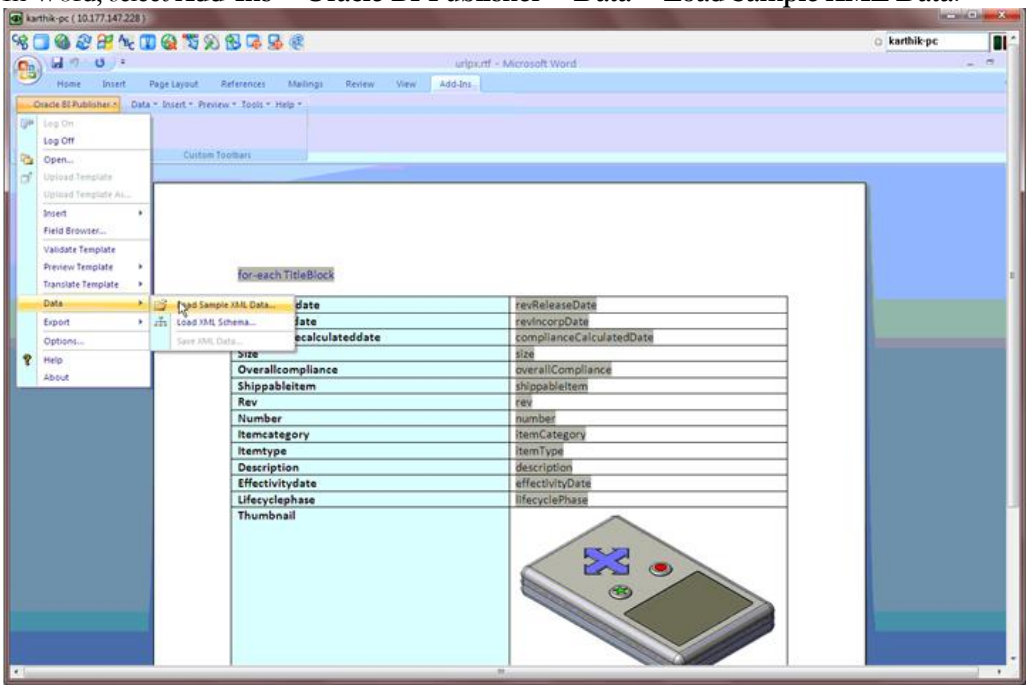
To map attributes using BI's Insert > Table Wizard into a Free Form option:

1. In Word, select **Add-Ins > Insert > Table Wizard** and then select **Free Form** in Table Wizard
2. Select the fields as you did in To map attributes using BI's Insert > Table Wizard's Table option: and view the Template selecting **Add-Ins > Oracle BI Publisher > Preview Template > RTF**.

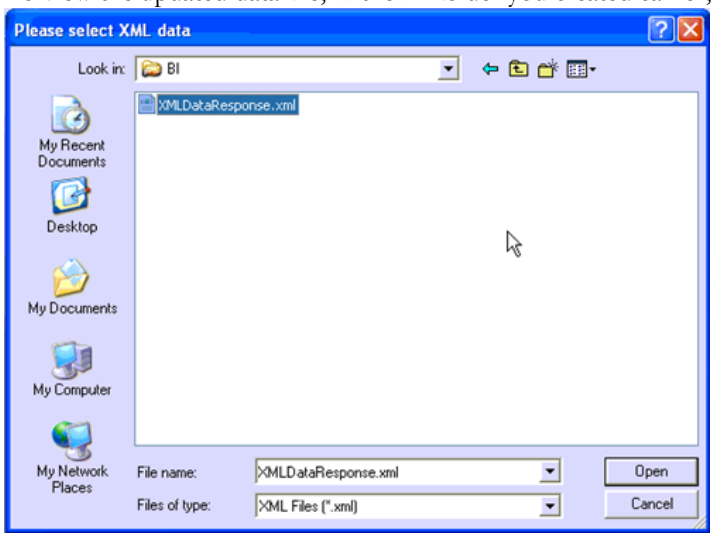


Loading Sample XML Data

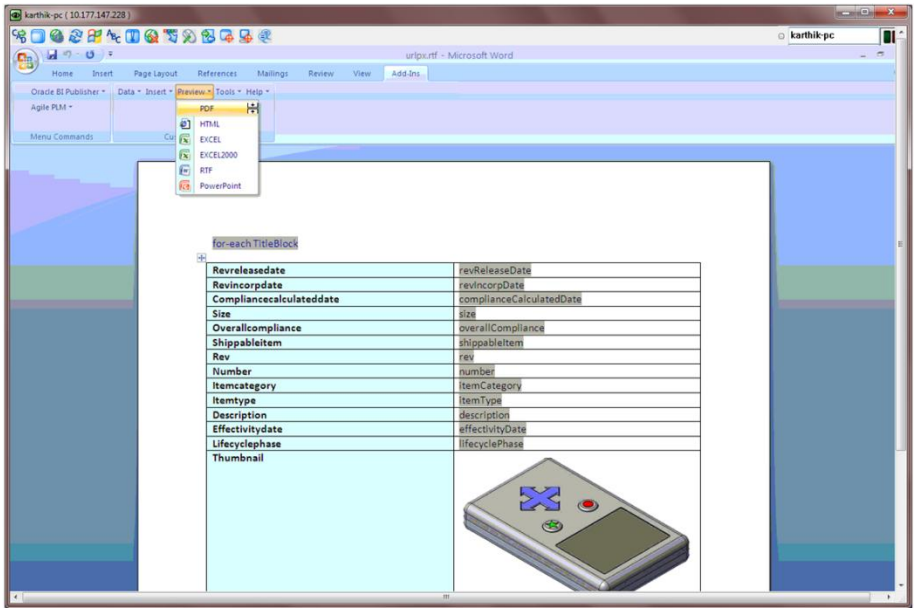
In Word, select **Add-Ins > Oracle BI Publisher > Data > Load Sample XML Data**.



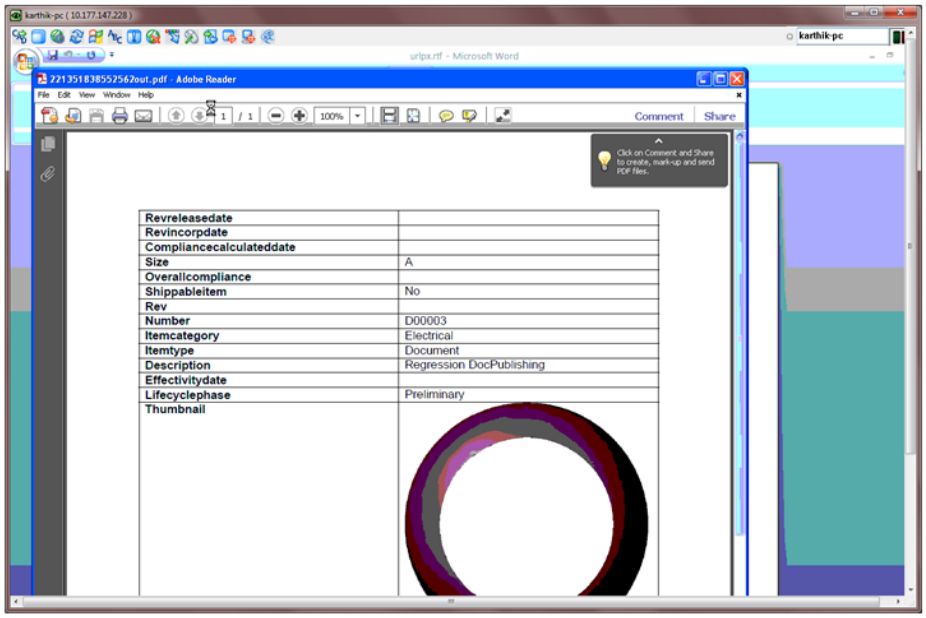
To view the updated data file, in the BI folder you created earlier, open **`XMLDataResponse.xml`**.



Save the template and open it in a supported format, for example PDF.



Afterwards, the following appears.



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

This page is blank.