



Agile Product Lifecycle Management

Agile PLM Document Publishing Solution

v9.3.1

Part No. E18342-01

January 2011

Oracle Copyright

Copyright © 1995, 2011, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS

Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications which may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

Oracle and Java are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

This software and documentation may provide access to or information on content, products and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third party content, products and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third party content, products or services. The RMW product includes software developed by the Visigoth Software Society.

CONTENTS

Oracle Copyright.....	ii
About this Whitepaper	1
Content and Organization	1
Intended Audience	1
Cited References.....	2
SDK Samples Folder and Document Publishing Examples.....	2
Dynamic Document Publishing	3
Overview.....	3
Solution Architecture	4
Operating Environment	4
Oracle Agile PLM	5
Oracle BI Publisher	5
Microsoft Word 2003/2007 and BI Publisher	5
Process Outline	6
Understanding the Process	7
Datasheet Configuration in Agile PLM	8
Generating the Data XML File	9
Creating and Testing the Template	10
Combining XML Data with Template - Publishing the Document	12
Setting Up the Environment to Publish Sample Documents.....	13
Overview.....	13
Installing BI Publisher Desktop.....	14
Agile PLM Administrator Configurations	14
Create the DocumentTemplate Subclass.....	15
Configuring Agile Content Services Filters	17
Defining Page 2 Fields for the Object.....	17
Defining Agile Content Services Filter for XML Data	18
Agile PLM Server Configurations	20
Understanding Process Extensions and Events Framework.....	20
Using Oracle Supplied Document Publishing Samples.....	21
Working with Script and Java PX Handlers	21
Getting Started with the Sample	23
Task Sequence.....	23
Configuring TemplateManagementStructureCreationPX.....	24
Configuring Event Components for TemplateManagementStructureCreationPX	25

TemplateManagementStructureCreationPX.....	27
Results from Running the TemplateManagementPX	28
Configuring SchemaGenerationPX	29
Configuring Event Components for SchemaGenerationPX.....	29
Properties File Settings for SchemaGenerationPX	31
Results from Running the SchemaGenerationPX	31
Generating Schema XSD and Data XML files	33
Configuring DataGenerationPX.....	34
Configuring Event Components for DataGenerationPX	34
Properties File Settings for DataGenerationPX.....	37
Modifying the DataGenerationPX Script.....	38
Results from Running the DataGenerationPX.....	38
Building BI Publisher Templates	39
Copying XSD and XML Files to the Local Drive	41
Loading Schema XSD and Data XML Using BI Publisher Plugin.....	41
Selecting Agile PLM Data Fields and Formatting the Template	42
Inserting Agile PLM Data Fields in the Template	43
Inserting and Formatting Tables.....	45
Inserting Images and Charts in Templates.....	45
Loading the Template into Agile PLM.....	47
Configuring DocumentGenerationPX.....	48
Configuring DocumentGenerationJavaPX.....	48
Configuring Event Components for DocumentGenerationJava PX	48
Properties File Settings for DocumentGenerationJavaPX	50
Results from Running the DocumentGenerationJavaPX	51
Configuring DocumentGenerationJavaOpen (URL PX)	53
Properties File Settings for DocumentGenerationJavaOpen PX	54
DocumentGenerationJavaPxOpen Output Sample - Keep or Remove?	54
Triggering DocumentGenerationPX	54
Modifying DocumentGenerationPX	55
Triggering the Event and Creating the Output File.....	55

Preface

Oracle's Agile PLM documentation set includes Adobe® Acrobat PDF files. The [Oracle Technology Network \(OTN\) Web site](http://www.oracle.com/technetwork/documentation/agile-085940.html) <http://www.oracle.com/technetwork/documentation/agile-085940.html> contains the latest versions of the Agile PLM PDF files. You can view or download these manuals from the Web site, or you can ask your Agile administrator if there is an Agile PLM Documentation folder available on your network from which you can access the Agile PLM documentation (PDF) files.

Note To read the PDF files, you must use the free Adobe Acrobat Reader version 9.0 or later. This program can be downloaded from the [Adobe Web site](http://www.adobe.com) <http://www.adobe.com>.

The [Oracle Technology Network \(OTN\) Web site](http://www.oracle.com/technetwork/documentation/agile-085940.html) <http://www.oracle.com/technetwork/documentation/agile-085940.html> can be accessed through Help > Manuals in both Agile Web Client and Agile Java Client. If you need additional assistance or information, please contact My Oracle Support (<https://support.oracle.com>) for assistance.

Note Before calling Oracle Support about a problem with an Agile PLM manual, please have the full part number, which is located on the title page.

TTY Access to Oracle Support Services

Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, 7 days a week. For TTY support, call 800.446.2398. Outside the United States, call +1.407.458.2479.

Readme

Any last-minute information about Agile PLM can be found in the Readme file on the [Oracle Technology Network \(OTN\) Web site](http://www.oracle.com/technetwork/documentation/agile-085940.html) <http://www.oracle.com/technetwork/documentation/agile-085940.html>.

Agile Training Aids

Go to the [Oracle University Web page](http://www.oracle.com/education/chooser/selectcountry_new.html) http://www.oracle.com/education/chooser/selectcountry_new.html for more information on Agile Training offerings.

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

About this Whitepaper

This chapter includes the following:

▪ Content and Organization.....	1
▪ Intended Audience.....	1
▪ Cited References.....	2
▪ SDK Samples Folder and Document Publishing Examples	2

Note This document 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*. The purpose of this document is to introduce Dynamic Document Publishing and is not intended as a User or Developer Guide, and will be augmented with these documents in the future.

Content and Organization

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

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

Intended Audience

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

Cited References

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

Oracle Agile PLM

- Agile PLM Readme
- Agile PLM SDK Developer Guide
- 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

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

Oracle BI Publisher

- Oracle BI Publisher 10g

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

SDK Samples Folder and Document Publishing Examples

Oracle provides Document Publishing configuration examples for PLM server and PLM Administrator. Server examples include PXs and related Java and properties files. Client configurations are included in a recording created by WebEx Recorder in .WRF format.

You can find these files in the Doc-Publishing folder, in SDK_samples.zip. The SDK_samples.zip folder which also contains the API HTML reference files and Sample applications, is maintained on the Oracle Agile PLM 9.3.1 Event and Web Services Samples Web site at <https://codesamples.samplecode.oracle.com/servlets/tracking/id/S614>. For more information and procedures to access its contents, contact your system administrator, or refer to your *Agile PLM Installation Guide*.

Dynamic Document Publishing

This chapter includes the following:

▪ Overview.....	3
▪ Solution Architecture.....	4
▪ Operating Environment.....	4
▪ Process Outline	5

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 publication of documents such as product data sheets, Parts List, or service manuals with embedded PLM data.

To support this solution, Agile PLM 9.3.1 provides two new Web services APIs for XML publishing (see [Oracle Agile PLM](#) on page 5). These APIs return an XML package with the object's schema and another with 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.

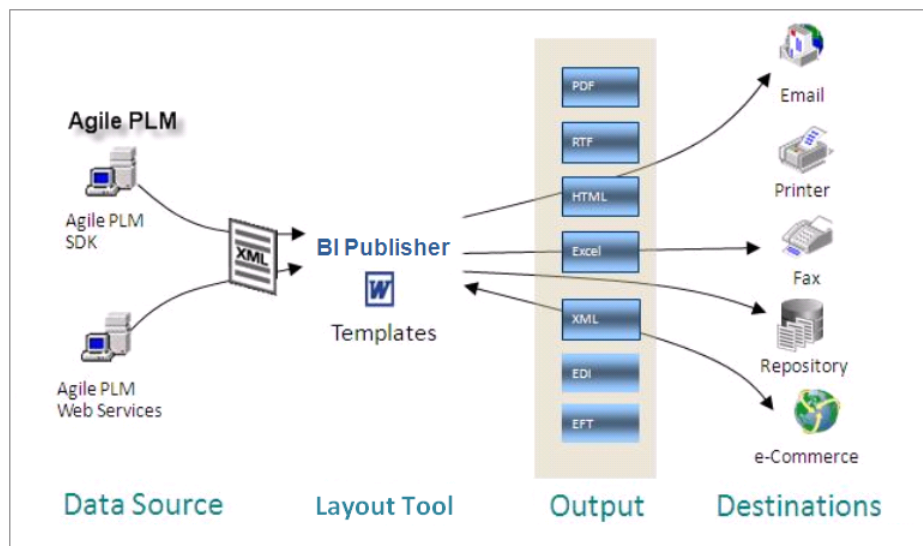
The Dynamic Document Publishing of product information can be used by Industrial, Retail, Life Sciences, Pharmaceutical, and High Tech industries to:

- 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
- Modify the content that is shared by other documents already stored in PLM
- Update documents that reference content that was modified

Solution Architecture

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

Figure 1: Document Publishing architecture



Operating Environment

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

The operating environment includes:

- Oracle Agile PLM
- Oracle BI Publisher
- Microsoft Word

Oracle Agile PLM

Oracle Agile PLM components are:

- Agile PLM Release 9.3.1 (Server and databases)
- Agile PLM Release 9.3.1 File Manager
- Agile PLM Release 9.3.1 SDK (Template Management Java and Script PXs)
- Agile PLM Release 9.3.1 Web Services APIs – The following APIs support Dynamic Document Generation:
 - **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 ECO, it will tell BI Publisher all of the possible attributes for ECOs. This is useful to be able to work with all potential attributes for 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.

Note For information on these APIs, refer to *Agile PLM WebServices 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 with 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 - The PLM Embedded version or a separate server version)

Note A license is necessary for Oracle BI Publisher. BI Publisher Enterprise is required if there is a need to publish from multiple versus a single source of data. For a list of Agile PLM and BI Publisher components that you must install and configure to enable Dynamic Document Publishing, see *Installing and Configuring BI Publisher and Word Plugins*.

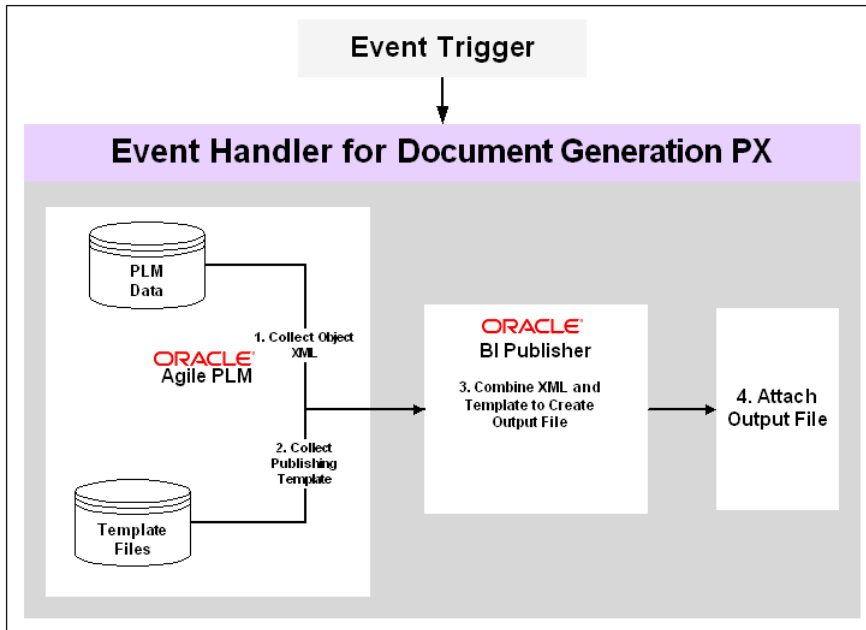
Microsoft Word 2003/2007 and BI Publisher

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

Process Outline

The following illustration is a streamlined view of the Document Publishing process. That is, a "trigger" invokes a "handler" and then steps #1, 2, 3, and 4 happen automatically.

Figure 2: Dynamic Document Publishing Process and Steps



The process is triggered by various Event subscriptions that are constructed in Agile PLM's Event Management framework. A user with Administrator privileges configures the Event subscriptions, which are detailed further on in this document. Typical Event Triggers include Release an ECO, Update Title Block for an Item, or any other user action that can generate a document in Agile PLM.

The Event subscription is used to call the Event Handler that performs the following sequence of tasks:

1. Collect Object XML – Object XML is dynamically retrieved from the object that is being processed
2. Collect Publishing Template – Template is a predefined BI Publisher RTF file stored in Agile PLM. The PX checks a field in the Object to determine which attachment in Agile PLM contains the BI Publisher Template
3. Combine XML and Template in BI Publisher to create output file – The sample PX passes the XML package and Template to the BI Publisher program running with Agile PLM which generates an output file and informs the PX of the file location
4. Attach output file – The sample PX uses Agile PLM SDK commands to attach the file to an Agile PLM business object

This process is illustrated with the aid of the following example, which updates a Datasheet with PLM data.

Understanding the Process

Consider the following Datasheet. It provides information about an assembly part that is not yet a PLM object. When this object loaded into the PLM, applicable information about this PLM object is maintained in the object's attributes. You can see some of these attributes in [Datasheet Configuration in Agile PLM](#) on page 8.

Because of recent business activities, there is a need to update some of these attributes, such as Name or Address, and then publish the updated Datasheet. The next few paragraphs summarize the Dynamic Document Publishing process to update and publish this document.

Figure 3: Datasheet before it is loaded into Agile PLM


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.

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 to convert these attributes to a Data XML file for BI Publisher processing.

ASM-00166
Preliminary
 Assembly • Data Sheet Object for Sun Server M4000 Demo Feb 11, 2010
 Unincorporated

Site: ALL Rev: Introductory
Navigator Actions

Title Block Changes SC Assy BOM Subclasss Assembly * Mfr Subclass Assembly Sites Prices Quality Compliance Supp

[Page Two](#) | [Doc Management Publishing](#) | [More Attributes](#) | [Security](#) | [Data Sheet Parameters](#)

DS Title : SUN SPARC ENTERPRISE
M4000 SERVER
Demo Feb 11, 2010

DS Introduction: 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.

DS Title 01 MT: Investment Protection, Scalability, Reliability, and Flexibility

DS Section 01: 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. For more flexibility, the Sun SPARC Enterprise M4000 server supports up to two Dynamic Domains, with a high level of granularity: CPU board-level domains for large, mission-critical workloads requiring maximum isolation, and single-socket-level domains for finer granularity with high isolation. For maximum

Generating the Data XML File

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

Figure 4: Creating and loading the Data XML file

```

1 |<?xml version="1.0" encoding="UTF-8"?>
2 |<AgileData xmlns:xsd="http://www.w3.org/2001/XMLSchema">
3 |
4 |  <Assembly>
5 |    <TitleBlock>
6 |      <number>ASM-00166</number>
7 |      <itemType>Assembly</itemType>
8 |      <lifecyclePhase>Preliminary</lifecyclePhase>
9 |      <description>Data Sheet Object for Sun Server M4000 Demo Feb 11, 2010</description>
10 |      <productLineS><Value>Automotive - Components</Value></productLineS>
11 |      <shippableItem>No</shippableItem>
12 |      <excludeFromRollup>No</excludeFromRollup></TitleBlock>
13 |    <Attachments>
14 |      <filename>AgileData_Assembly.xml</filename>
15 |      <fileDescription>Agile Data for ASM-00166</fileDescription>
16 |      <fileSize>5441</fileSize>
17 |      <fileType>xml</fileType>
18 |      <folderNumber>FOLDER0001017</folderNumber>
19 |      <folderVersion>2</folderVersion>
20 |      <modifiedDate>2010-02-13T06:25:23Z</modifiedDate>
21 |      <lastViewDate>2010-02-13T06:25:22Z</lastViewDate>
22 |      <checkinUser>Deron Johnstone</checkinUser></Attachments>
23 |    <PageThree>
24 |      <DSTitle>SUN SPARC ENTERPRISE
25 |        M4000 SERVER
26 |        Demo Feb 11, 2010</DSTitle>
27 |      <DSIntroduction>Companies can't afford to have business-critical services go offline. To meet t
28 |      <DSTitle01MT>Investment Protection, Scalability, Reliability, and Flexibility</DSTitle01MT>
29 |      <DSSection01>With the <b>Sun SPARC Enterprise M4000</b> servers, you can protect your IT
30 |      Mainframe-class RAS features come standard in the Sun SPARC Enterprise M4000 server, including autor
31 |      For more flexibility, the Sun SPARC Enterprise M4000 server supports up to two Dynamic Domains, with
32 |      <DSTitle02MT>Solaris: The World's Most Advanced Operating System</DSTitle02MT>
33 |      <DSSection02>The foundation of the Sun SPARC Enterprise M4000 server is the Solaris 10 OS, which cor
34 |      <DSKeyFeatures>MAINFRAME-CLASS RELIABILITY, AVAILABILITY, AND SERVICEABILITY IN A VALUE-PRICED SERV
35 |      <ul><li>Mix and match the boards with earlier versions of the SPARC processor in a singl
36 |      <li>Binary compatibility with earlier versions of your applications</li>
37 |      <li>Scalable, mainframe-class computing for the open systems market</li>
38 |      <li>Advanced virtualization technologies, methodologies, and services, making Sun SPARC Enterp
39 |      <li>Up to four quad-core SPARC64 VII or dual-core SPARC64 VI processors</li>

```

Creating and Testing the Template

Template is an RTF file created and formatted using Word, BI Publisher, and object's attributes in Agile PLM. For information and procedures to create Templates, see [Building BI Publisher Templates](#) on page 39 and [Cited References](#) on page 2. Assuming you have created the Template file, running the loadXMLSchema API, again, from MS Word and selecting Add-Ins > Data > Load XML Data will generate the following XSD file showing the selected attributes.

Figure 5: The Datasheet Schema XML file

```

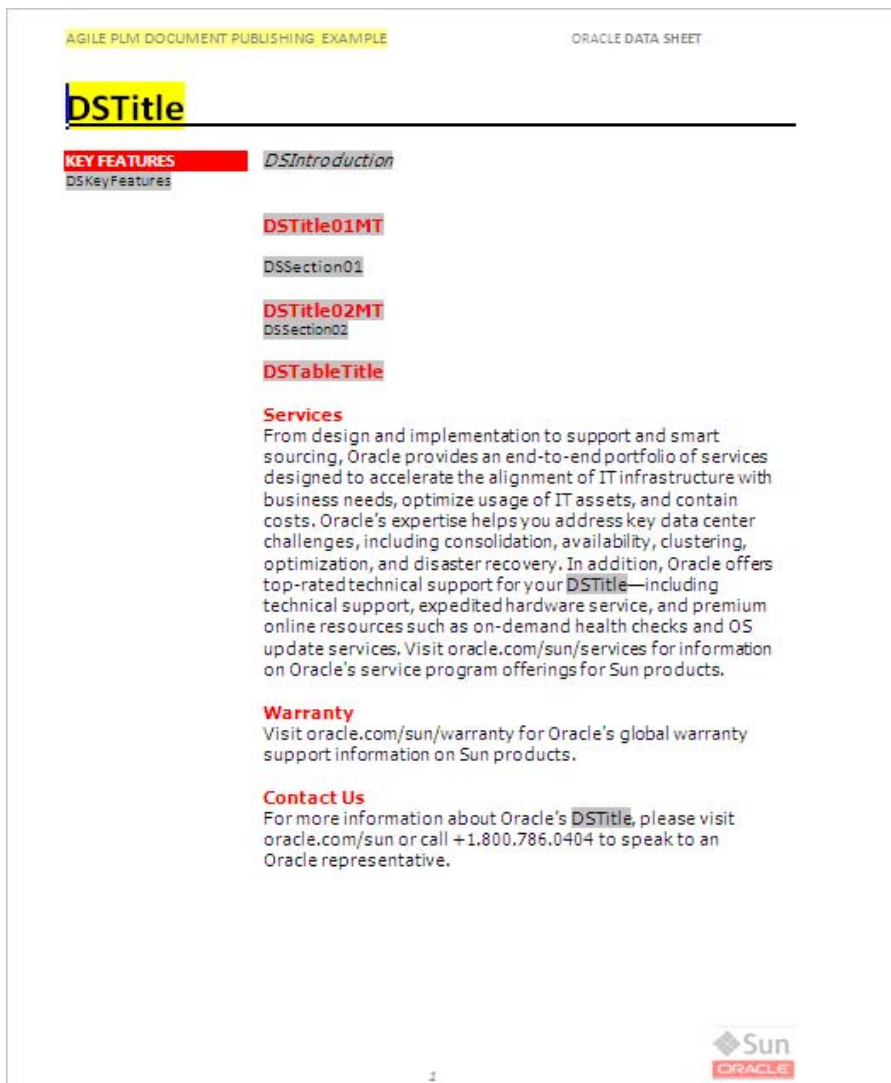
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
3   <xs:element name="AgileData" type="AgileDataType"/>
4
5   <xs:complexType name="AgileDataType">
6     <xs:sequence>
7       <xs:element name="Assembly" type="AssemblyType" minOccurs="0" maxOccurs="unbounded" nillable="true"/>
8     </xs:sequence>
9   </xs:complexType>
10
11   <xs:complexType name="AssemblyType">
12     <xs:sequence>
13       <xs:element name="BOM" type="AssemblyBOMType" minOccurs="0" maxOccurs="1" nillable="true"/>
14       <xs:element name="TitleBlock" type="AssemblyTitleBlockType" minOccurs="0" maxOccurs="1" nillable="true"/>
15       <xs:element name="Attachments" type="AssemblyAttachmentsType" minOccurs="0" maxOccurs="1" nillable="true"/>
16       <xs:element name="Manufacturers" type="AssemblyManufacturersType" minOccurs="0" maxOccurs="1" nillable="true"/>
17       <xs:element name="Specifications" type="AssemblySpecificationsType" minOccurs="0" maxOccurs="1" nillable="true"/>
18       <xs:element name="Relationships" type="AssemblyRelationshipsType" minOccurs="0" maxOccurs="1" nillable="true"/>
19       <xs:element name="Quality" type="AssemblyQualityType" minOccurs="0" maxOccurs="1" nillable="true"/>
20       <xs:element name="Sites" type="AssemblySitesType" minOccurs="0" maxOccurs="1" nillable="true"/>
21       <xs:element name="Prices" type="AssemblyPricesType" minOccurs="0" maxOccurs="1" nillable="true"/>
22       <xs:element name="PendingChanges" type="AssemblyPendingChangesType" minOccurs="0" maxOccurs="1" nillable="true"/>
23       <xs:element name="WhereUsed" type="AssemblyWhereUsedType" minOccurs="0" maxOccurs="1" nillable="true"/>
24       <xs:element name="PendingChangeWhereUsed" type="AssemblyPendingChangeWhereUsedType" minOccurs="0" maxOccurs="1" nillable="true"/>
25       <xs:element name="Compositions" type="AssemblyCompositionsType" minOccurs="0" maxOccurs="1" nillable="true"/>
26       <xs:element name="Substances" type="AssemblySubstancesType" minOccurs="0" maxOccurs="1" nillable="true"/>
27       <xs:element name="PageTwo" type="AssemblyPageTwoType" minOccurs="0" maxOccurs="1" nillable="true"/>
28       <xs:element name="ChangeHistory" type="AssemblyChangeHistoryType" minOccurs="0" maxOccurs="1" nillable="true"/>
29       <xs:element name="Suppliers" type="AssemblySuppliersType" minOccurs="0" maxOccurs="1" nillable="true"/>
30       <xs:element name="QCRs" type="AssemblyQCRsType" minOccurs="0" maxOccurs="1" nillable="true"/>
31       <xs:element name="History" type="AssemblyHistoryType" minOccurs="0" maxOccurs="1" nillable="true"/>
32       <xs:element name="Instances" type="AssemblyInstancesType" minOccurs="0" maxOccurs="1" nillable="true"/>
33       <xs:element name="PageThree" type="AssemblyPageThreeType" minOccurs="0" maxOccurs="1" nillable="true"/>
34     </xs:sequence>
35   </xs:complexType>
36
37   <xs:complexType name="AssemblyBOMType">
38     <xs:sequence>
39       <xs:element name="BOMRow" type="AssemblyBOMRowType" minOccurs="0" maxOccurs="unbounded" nillable="true"/>

```


Testing/Viewing the Template

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

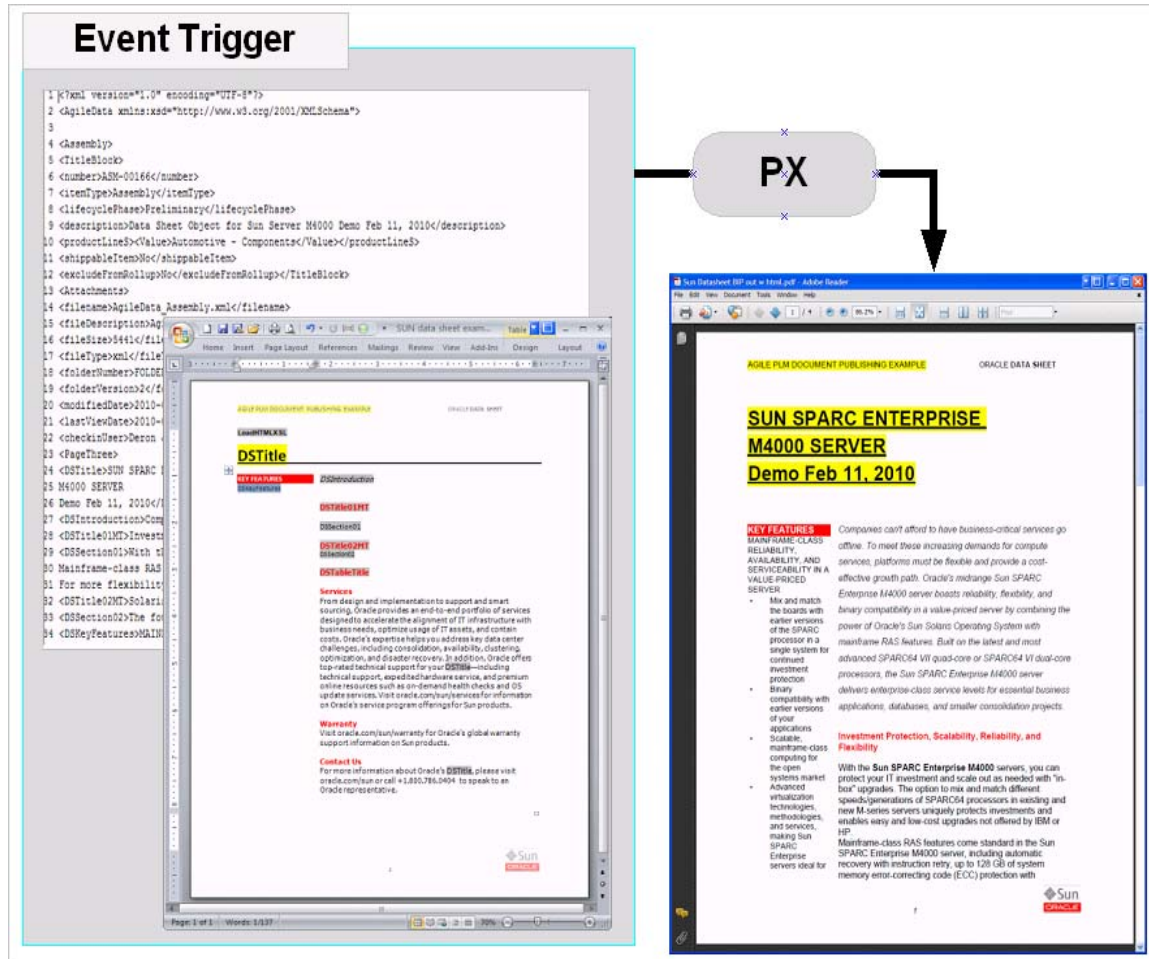
Figure 6: Sample Template output



Combining XML Data with Template - Publishing the Document

When the appropriate Event subscription which is set up in advance is triggered, XMLData 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](#) on page 23.

Figure 7: Combining Data XML and Template to generate the document



Setting Up the Environment to Publish Sample Documents

This chapter includes the following:

▪ Overview.....	13
▪ Installing BI Publisher Desktop.....	14
▪ Agile PLM Administrator Configurations.....	14
▪ Agile PLM Server Configurations.....	19

Overview

Setting up the environment to publish the sample, requires the following installations and Agile PLM configurations.

Prerequisites

Installing BI Publisher Install 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 content of form selected fields

Configuring Agile PLM Administrator

These configurations include:

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

Configuring Agile PLM Server

This involves creating and configuring the following Process Extensions (PXs):

- Template Management Structure Creation PX
- SchemaGeneration PX
- DataGeneration PX
- Document Generation PX

Getting Started with Document Publishing

These tasks address generating the following:

- Agile PLM structure and placeholders using the Oracle supplied PXs
- Templates with BI Publisher

Note If you are installing Agile PLM 9.3.1, or using an installed copy, make sure File Manager and SDK are selected or are already installed. For installation procedures, refer to documents listed under "Install/Upgrade Guides" at http://download-llnw.oracle.com/docs/cd/E17287_02/otn/docset.html.

Installing BI Publisher Desktop

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

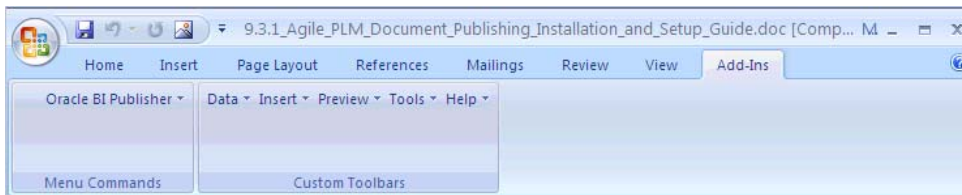
To install BI Publisher Desktop:

1. Download and install BI Publisher Desktop 11g or 10g from OTN at:
<http://www.oracle.com/technetwork/middleware/bi-publisher/documentation/xmlpdocs-084437.html>.

Note Installing Desktop 11g requires downloading all four installation files from OTN.

2. Verify Add-Ins is available in MS Word.

Figure 8: BI Publisher Add-In in MS Word



Agile PLM Administrator Configurations

These are the Agile PLM Administrator configurations and documented in the following paragraphs:

- 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


Note For complete documentation of the Event Management framework, refer to the *Agile PLM Administrator Guide*.

Create the DocumentTemplate Subclass

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

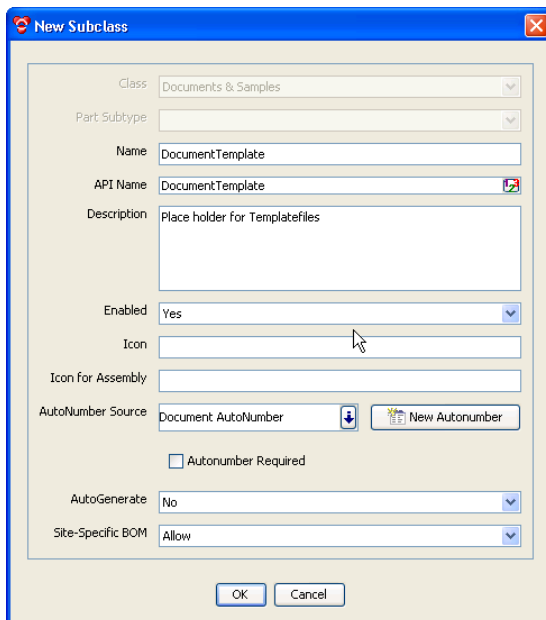
Note This is a onetime configuration and provides a placeholder for all Template files.

To create the DocumentTemplate Subclass:

1. Log in to Agile PLM Java Client as an administrator.
2. Select Admin > Classes > Documents Class to open the Class: Documents Class dialog.
3. In Class: Documents Class 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.

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.

Figure 9: Configuring the DocumentTemplate Subclass



The 'New Subclass' dialog box is shown with the following configuration:

- Class:** Documents & Samples
- Part Subtype:** (empty)
- Name:** DocumentTemplate
- API Name:** DocumentTemplate
- Description:** Place holder for Templatefiles
- Enabled:** Yes
- Icon:** (empty)
- Icon for Assembly:** (empty)
- AutoNumber Source:** Document: AutoNumber
- AutoNumber Required:** ☐
- AutoGenerate:** No
- Site-Specific BOM:** Allow

5. Click OK.

The DocumentTemplate subclass General Information page appears.

Figure 10: DocumentTemplate General Information tab

Subclass:DocumentTemplate

General Information | User Interface Tabs | Lifecycle Phases | Process Extensions | Event Subscribers | History

Name: DocumentTemplate

API Name: DocumentTemplate

Description:

Enabled: Yes

Autonumber Required: No

AutoGenerate: No

Icon:

Icon for Assembly:

Site-Specific BOM: Allow

Failure Mode: Failure Mode List - [DocumentTemplate]

Class: Documents & Samples

AutoNumber Source:

Save Refresh Close

- To continue, see [Defining Page 2 Fields for the Object](#) on page 17.

To set the Title Block Number fields:

- In Agile PLM Java client select the Admin tab.
- Select Classes > Document Class > User Interface tabs > TitleBlock > Attributes:Title Block > Number field.

The Attributes:Number page appear.

Figure 11: Setting Page 2 Attributes for the object

Attributes:Number

Name: Number

API Name: number

Description:

Visible: Yes

Include Characters: All

MaxLength: 75

Max System Length: 75

Order: 1

Required: Yes

Available for Subscribe: No

Enable for Search Criteria: Yes

Attribute: ITEM.ITEM_NUMBER

Type: Text

Input Width: Long

Change controlled: No

Save Refresh Close

- Set MaxLength field 75 and Include Characters field to All.
- Click Save.

This completes configuring DocumentTemplate subclass attributes.

Note You can rename this subclass if you modify the configuration of the PX. For example, changing the `TEMPLATE_SUBCLASS_API_NAME=DocumentTemplate` in `ManagementStructure.properties` file.

Configuring Agile Content Services Filters

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

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

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

Defining Page 2 Fields for the Object

The required fields for the Sample are a header field, two Text fields, and one List field. The *Base IDs* are for later use.

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

Note The selected fields provide flexibility for the sample and may not be necessary in a production implementation.

To define Object Page 2 fields:

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

The following Attributes dialog appears.

Figure 12: text field attributes

Attributes:List11

Name List11

API Name list11

Description Output Type

Visible Yes

List Doc Publishing Output Type List [New List](#)

List ID 2475954

Cascade List No

List Detail Box [View Detail](#)

DefaultValue PDF

Required Yes

Available for Subscribe No

Enable for Search Criteria No

Attribute PAGE_TWO.LIST11

Type List

Input Width Long

Change controlled No

[Save](#) [Refresh](#) [Close](#)

2. Set the fields and then click Save.
3. In Java client, select Admin > Classes > Documents Class > User Interface Tabs > Page 2, and configure the remaining Text and Header fields as shown in the following figure.

Figure 13: Object Page 2 fields for Document Publishing sample

Name	API Name	Type	Y...	List	Default Value	Attribute	Base ID
Doc Management Publishing	heading02	Heading	Yes	N/A	N/A	PAGE_TWO.HEADING02	12258
Filter	text12	Text	Yes	N/A	Itemtabs ... 50 50 ...	PAGE_TWO.TEXT12	1302
Template Holder	text11	Text	Yes	N/A	... 50 50 ...	PAGE_TWO.TEXT11	1301
Output Type	list11	List	Yes	DocType	PDF	PAGE_TWO.LIST11	1271


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

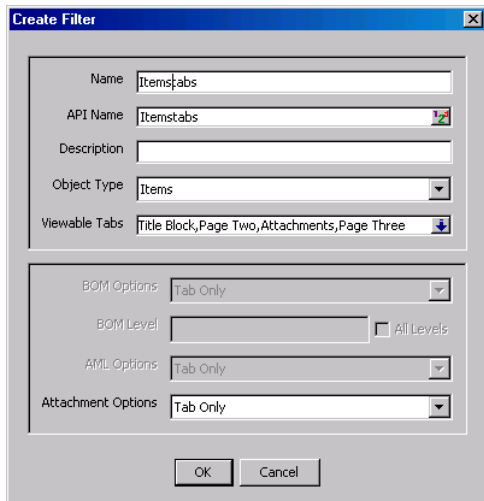
Defining Agile Content Services Filter for XML Data

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

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

To create the ACS filter:

1. Log in to Agile PLM Java Client as administrator.
2. Select Admin > System Settings > Agile Content Service > Filters  to open the Create Filter dialog.



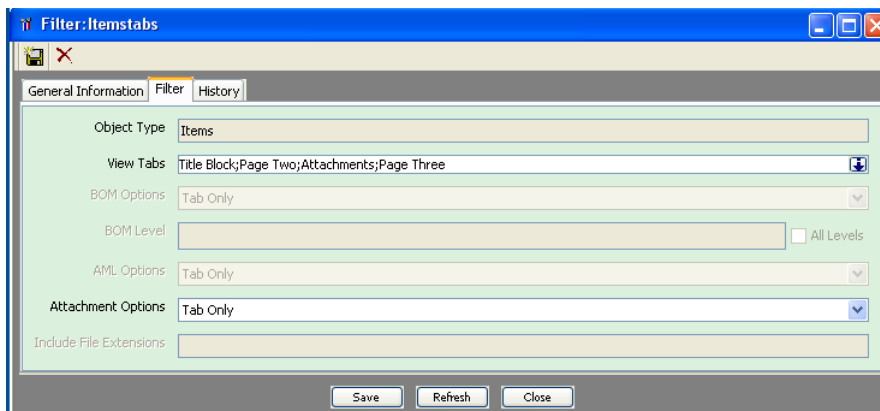
The 'Create Filter' dialog box contains the following fields and options:

- Name:** Itemstabs
- API Name:** Itemstabs
- Description:** (empty text box)
- Object Type:** Items (dropdown menu)
- Viewable Tabs:** Title Block, Page Two, Attachments, Page Three (text box with a dropdown arrow)
- BOM Options:** Tab Only (dropdown menu)
- BOM Level:** (empty text box) with an ☐ All Levels checkbox
- AML Options:** Tab Only (dropdown menu)
- Attachment Options:** Tab Only (dropdown menu)
- Buttons:** OK and Cancel

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

3. Create a new filter called Itemstabs with a minimum of Title Block, Page Two, Page Three, and Attachments tabs.
4. Click OK and select the Filter tab to view the new filter.
5. Select the settings shown in the following figure, and then click Save.

Figure 14: ACS Filter configuration interface



The 'Filter: Itemstabs' configuration interface shows the following settings:

- Object Type:** Items
- View Tabs:** Title Block; Page Two; Attachments; Page Three
- BOM Options:** Tab Only
- BOM Level:** (empty text box) with an ☐ All Levels checkbox
- AML Options:** Tab Only
- Attachment Options:** Tab Only
- Include File Extensions:** (empty text box)
- Buttons:** Save, Refresh, Close

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 enable creating Templates and generating the schema and document files. These configurations make use of PXs described in [Using Oracle Supplied Document Publishing Examples](#) on page 21. For more information on Event components, refer to *Agile PLM SDK Developer Guide* and *Agile PLM Administrator Guide*.

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* 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](#) (on page 19).

Using Oracle Supplied Document Publishing Samples

Configurations described in setting up the PLM server use the following Java and Script PXs (Event Handlers):

- TemplateManagementStructureCreation
- SchemaGeneration
- DataGeneration
- DocumentGeneration

The Doc-Publishing folder contains the Java, Properties, and Resources files that you need to set up and deploy the sample Java and Script PXs. You can find these PXs and other components in the Doc-Publishing folder in SDK_samples.zip. See SDK Samples Folder and Document Publishing Examples.

Figure 15: Oracle-supplied PXs

Name	Size	Type
DataGeneration		File Folder
DocumentGenerationJava		File Folder
DocumentGenerationJavaOpen		File Folder
DocumentGenerationWSPX		File Folder
GlobalSchemaGeneration		File Folder
SchemaGeneration		File Folder
TemplateCreation		File Folder
WordPlugin		File Folder
build.bat	1 KB	MS-DOS Batch File
build.sh	1 KB	SH File
custom.property	1 KB	PROPERTY File

Before using Java PXs, you must deploy them first. For procedures, see [Working with Script PX and Java PX Handlers](#) on page 21. There is no need to deploy the Groovy equivalent Script PXs. For information on deploying Script PXs, refer to *Agile PLM Administrator Guide*.

Working with Script and Java PX Handlers

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

To deploy the Script PXs, refer to *Agile PLM Administrator Guide*. To deploy the Oracle supplied Java PXs, you must first create the JAR files as shown below.

To create JAR files:

1. In Doc-Publishing, open the custom.property file and specify drive name and path for the Agile PLM server ias.deploy, and the Application server (OAS, or WLS) inias.home.

Figure 16: Configuring the Custom Properties file for JAR files

```
ias.deploy.dir=D:/build/agile931/agileDomain/applications/APP-INF/lib
ias.home=D:/OC4J10133
px.deploy.loc=D:/build/agile931/integration/sdk/extensions
ANT_HOME=C:/apache-ant-1.7.1
```

2. In px.deploy, specify the location the PXs will reside in your environment.

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

Note You can use these PXs to implement the Dynamic Document Publishing capabilities by creating the Event Handler and Event Subscriber to trigger the Event. For details see [Template Management PX](#) on page 24, [Schema Generation PX](#) on page 29, [Data Generation PXs](#) on page 34, and [Document Generation PXs](#) on page 48. Alternatively, you can use the information to develop your own Java Client and server configurations.

Getting Started with the Sample

This chapter includes the following:

▪ Task Sequence.....	23
▪ Configuring TemplateManagementStructureCreationPX	24
▪ Configuring SchemaGenerationPX.....	29
▪ Generating Schema XSD and Data XML files	33
▪ Configuring DataGenerationPX	34
▪ Building BI Publisher Templates.....	39
▪ Configuring DocumentGenerationPX	48
▪ Triggering the Event and Creating the Output File	55

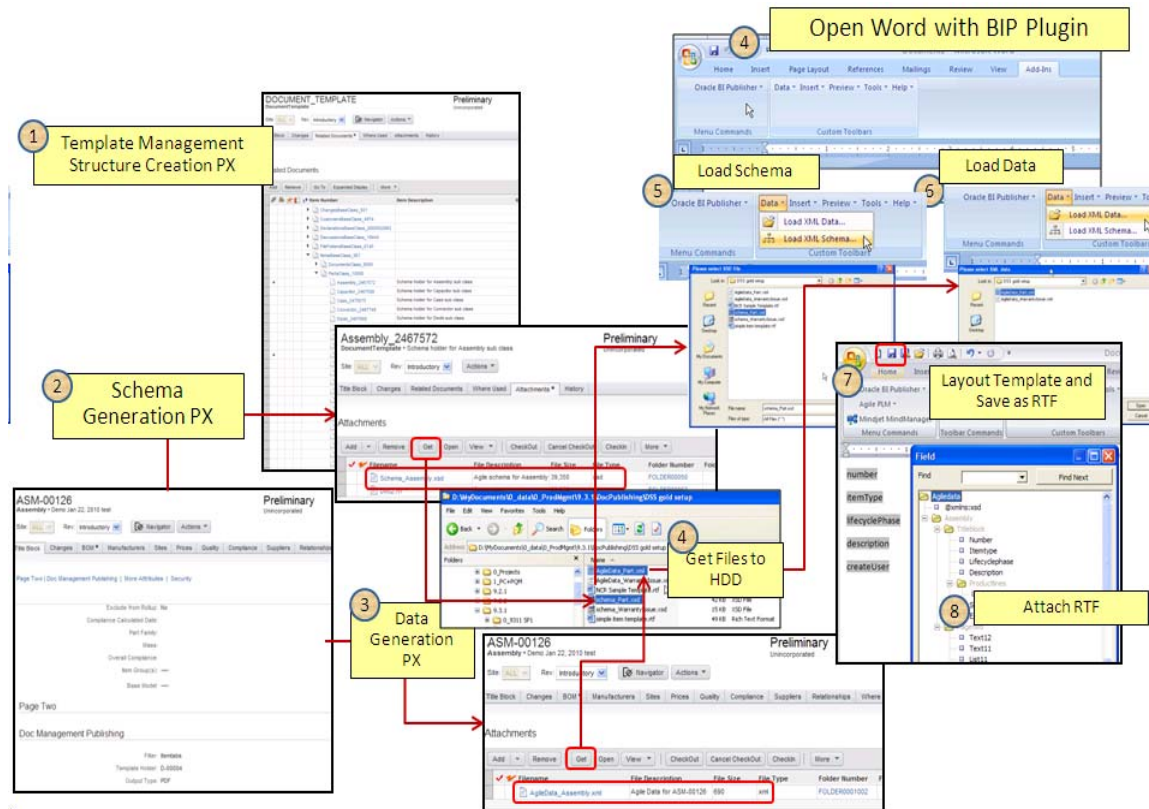
Task Sequence

Publishing the Sample requires the completion of the following tasks:

1. Setting up, configuring, and running the TemplateManagementStructureCreation PX
2. Setting up, configuring, and running the SchemaGeneration PX
3. Setting up and configuring the DataGeneration PX
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 and saving the Word file in RTF format
8. Uploading the Template into Agile PLM
9. Triggering the Document Generation Event to create the output file

The task sequence is summarized in the following illustration.

Figure 17: Publishing the Sample



Configuring TemplateManagementStructureCreationPX

The TemplateManagementStructureCreationPX creates a 3 level Bill of Material (BOM) for Base Classes, Classes, and Subclasses defined in [Configuring DocumentTemplate Subclass](#) on page 15. This is a placeholder for all future .RTF template files.

Note This PX is run once only and is not necessary if the Schema Structure is not needed

You can find this Script or Java PX in SDK Samples Folder and Document Publishing Examples. Depending on the Application Server (OAS, WLS) running in your installation, the path to the Script PX and Java PX with its .JAR and Properties files are:

- Script PX for WLS – 931_wls_sdk\samples\Doc-Publishing\TemplateCreation\TemplateCreationScript.groovy

- Script PX for OAS – 931_oas_sdk\samples\Doc-Publishing\TemplateCreation\TemplateCreationScript.groovy
- Java PX for WLS – 931_wls_sdk\samples\Doc-Publishing\TemplateCreation\samples\TemplateManagementStructureCreation\TemplateManagementStructureCreationPX.java
- Java PX for OAS – 931_oas_sdk\samples\Doc-Publishing\TemplateCreation\samples\TemplateManagementStructureCreation\TemplateManagementStructureCreationPX.java

Configuring Event Components for TemplateManagementStructureCreationPX

These configurations involve creating the necessary Event masks, Handler masks, and Subscriber masks.

To create Event mask and set Event Type:


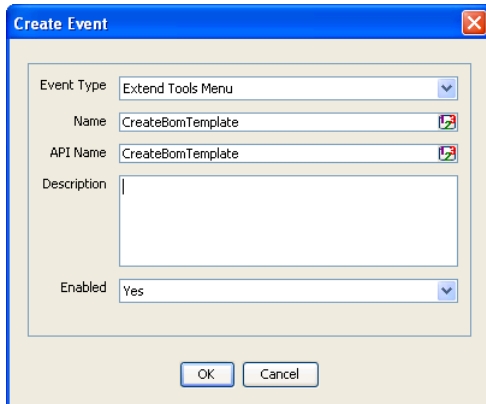

1. Log in to Java Client with Admin privileges.
2. In Java Client, select Admin > System Settings > Event Management > Events.
3. In Events page, select the New  button to open the Create Event dialog and define an Event mask called CreateBomTemplate for Object Type Parts with the following settings:

Figure 18: Create Event mask for TemplateManagementStructureCreationPX

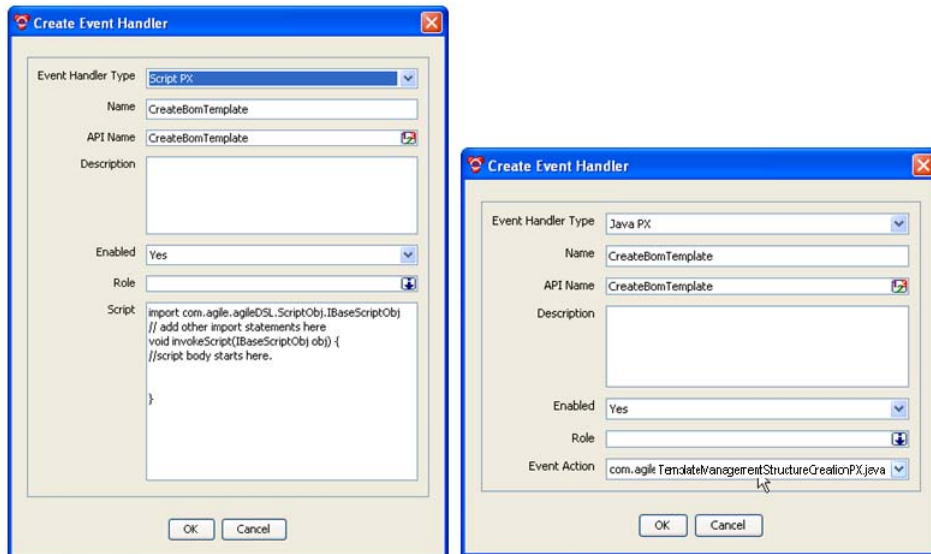


4. Click OK.

To set up the Event Handler mask:

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.
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 Script PX, or Java PX. You can find the Oracle supplied Script and Java PX in

Doc-Publishing folder in SDK_samples.zip described in [Template Management PX](#) on page 24.



5. To configure the Script or Java PX Handler, do as follows.


▫ For Script PX Event Handler:

In Create Event Handler, paste the contents of TemplateCreationScript.groovy 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*. You can also find information on configuring Script PXs in *Agile PLM Administrator Guide*.

▫ For Java PX Event Handler:

1. Make sure the Event Action for this Java PX is deployed. See [Creating JAR Files and Deploying PXs \(Event Handlers\)](#).
2. Check the values in TemplateManagementStructureCreation.properties file and make sure they conform to Java Client Admin settings shown in [Properties File Settings for TemplateManagementStructureCreationP](#) (on page 27)X.
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 = Synchronous
 - Error Handling Rule = Stop

- Click the drop-down arrow to select the Event and Event Handler you created earlier.

Figure 19: Template Management Event Subscriber

- Click OK.

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.

Figure 20: Properties file settings for Template Management 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
```

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.

Results from Running the TemplateManagementPX

The Script PX or Java PX 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
2. Create a 3-level BOM with Level 1 for all Agile Base classes in the system

Figure 21: Template Management Outputs

Output

D00840 AGILECLASSES

DocumentTemplate

Site: ALL Rev: Introductory Navigator Actions

Title Block Changes BOM Sites Quality Relationships Where Used Attachments History

BOM

Add Remove Go To Expanded Display More

Item Number	Item Description	Item Rev
3+ D00841 __CHANGES		
D00842 __MANUFACTURER_ORDERS		
D00843 __MCO		
D00844 __CHANGE_ORDERS		
D00845 __ECO		
D00846 __CHANGE_REQUESTS		
D00848 __DEVIATIONS		
D00850 __PRICE_CHANGE_ORDERS		
D00852 __STOP_SHIPS		
D00854 __SITE_CHANGE_ORDERS		
D00856 __CUSTOMERS		
D00859 __DECLARATIONS		

Configuring SchemaGenerationPX

The purpose of SchemaGenerationPX is to programmatically generate XML schema files using the Agile Java API for a given object. Document templates and document generation PXs, using BI Publisher Java API reside in Agile PLM.

It is necessary to run this PX for each Subclass 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 – 931_wls_sdk\samples\Doc-Publishing\TemplateCreation\SchemaGenerationScript.groovy
- Script PX – 931_oas_sdk\samples\Doc-Publishing\TemplateCreation\SchemaGenerationScript.groovy
- Java PX – 931_wls_sdk\samples\Doc-Publishing\SchemaGeneration\samples\SchemaGenerationPX.java
- Java PX – 931_oas_sdk\samples\Doc-Publishing\SchemaGeneration\samples\SchemaGenerationPX.java

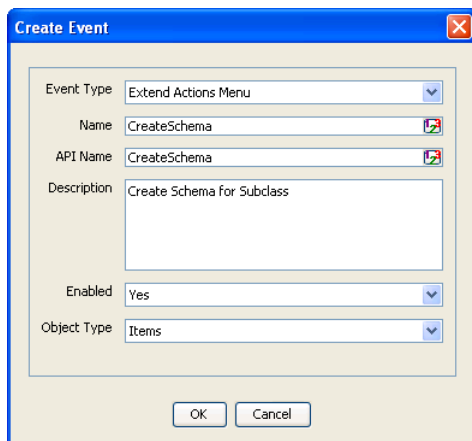
Configuring Event Components for SchemaGenerationPX

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:

1. Follow the steps in [Configuring Event Components for TemplateManagementStructureCreation PX](#) on page 25 and define an Event mask called CreateSchema for Object Type Items with the following settings:

Figure 22: Create Event mask for SchemaGeneration PX



2. Click OK.

To set up the Event Handler mask:

1. Use the information in [Configuring Event Components for TemplateManagementStructureCreation PX](#) on page 25 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 PX in Doc-Publishing folder in SDK_samples.zip described in [Configuring Event Components for TemplateManagementStructureCreation PX](#) on page 25.
3. To configure your Script PX 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*. You can also find information on configuring Script PXs in *Agile PLM Administrator Guide*.
 - For Java PX Event Handler:
 - a. Make sure the Event Action for this Java PX is deployed. See Deploying PXs (Event Handlers).
 - b. Check the values in SchemaGeneration.properties file and make sure they conform to settings defined in Java Client Admin in [Properties File Settings for Schema Generation PX](#) on page 31.
4. Click OK.

To configure the Event Subscriber mask:


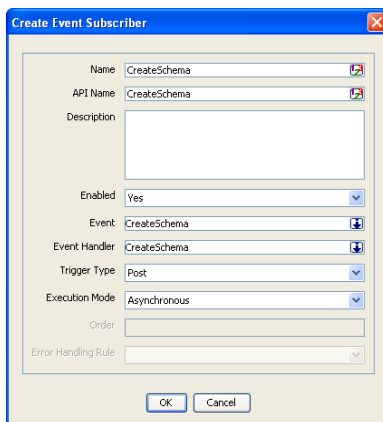
1. In Java Client with Admin privileges, select Admin > System Settings > Event Management > Event Subscribers.
2. In Event Subscribers pane, select the New  button to open the Create Event Subscriber dialog.
3. Create a new Event Subscriber called CreateSchema with settings shown in the following figure.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

Figure 23: Create Event Subscriber for CreateSchema Event


5. Click OK.

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.

Figure 24: Properties file settings for Schema Generation 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		

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.

Results from Running the 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 [Agile PLM Administrator Configurations](#) on page 14 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 defined as follows:

- 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 25: Event type settings for GlobalSchemaPX

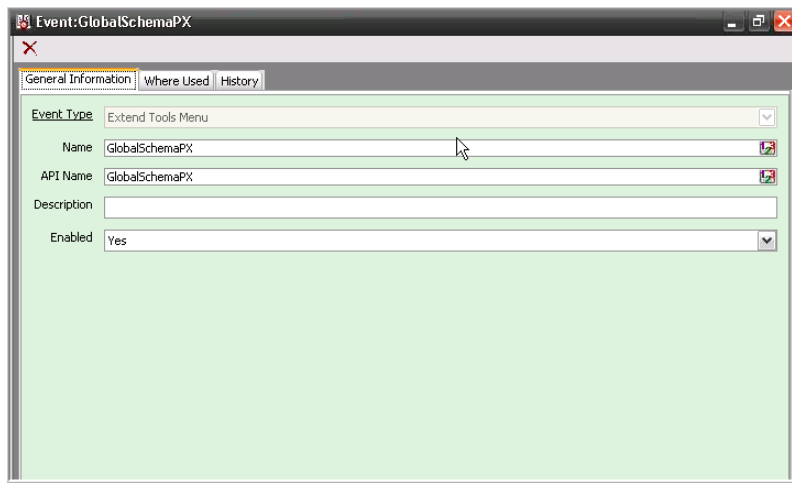


Figure 26: Output of SchemaGeneration PX

```
<?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>
```

Note In this output, Document is the Class name and Documents is name of the Subclass of Document.

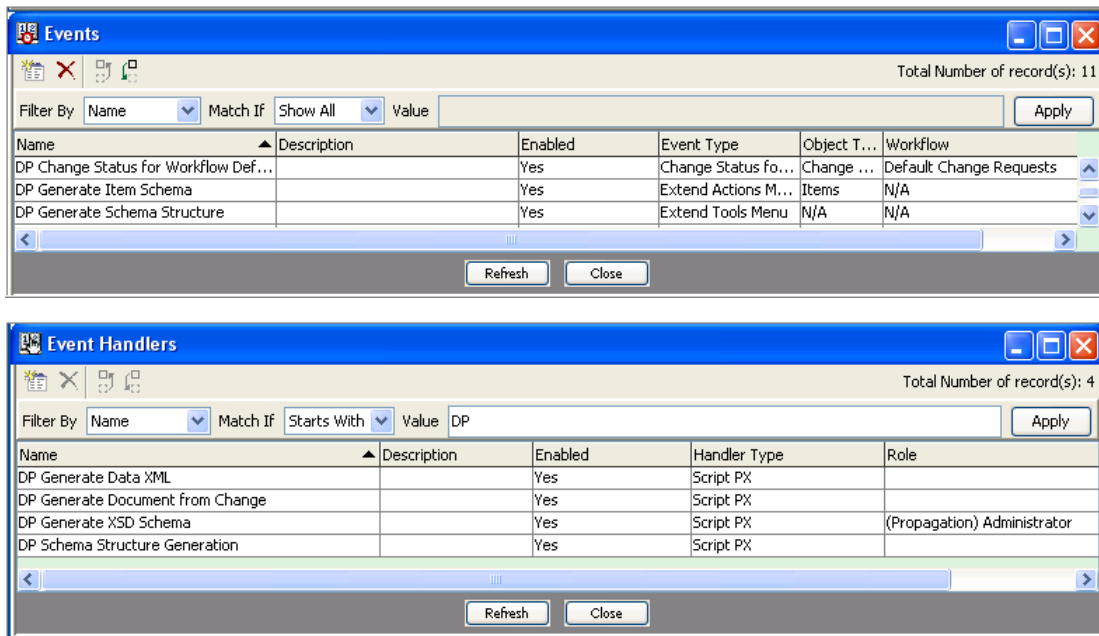
Generating Schema XSD and Data XML files

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 BaseClasses/Classes/SubClasses 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 respective Page Two attributes. In this case, DocType, Filter and TemplateHolder. These prerequisites for this PX were defined in [Agile PLM Administrator Configurations](#) on page 14 and [Configuring DataGeneration PX](#) on page 34.

Note For convenience and to facilitate search, in the Sample, names used for the PX Handlers start with DP for Document Publishing. These names are shown in the following illustration.

Figure 27: 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.

Note The PX does not rely on the Filter and generates the entire schema.

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 the all Document Publishing attributes are correct before triggering this PX

This 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.

Configuring DataGenerationPX

The overall aim 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 the document authoring tool. When run, the PX creates and loads the XML file into the authoring tool (MS Word) to test the Template with BI Publisher.

As indicated in Creating JAR Files and Deploying PXs (Event Handlers) this PX requires creating the Item – Part/Document and setting the respective Page Two attributes DocType, Filter and TemplateHolder.

Note This sample PX binds the Event to update the Title Block, which 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,

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 Components for TemplateMgtStructureCreationPX](#) on page 25 and define an Event called Create Object for Object Type Part and the following settings:

Figure 28: Create Event mask for DataGenerationPX

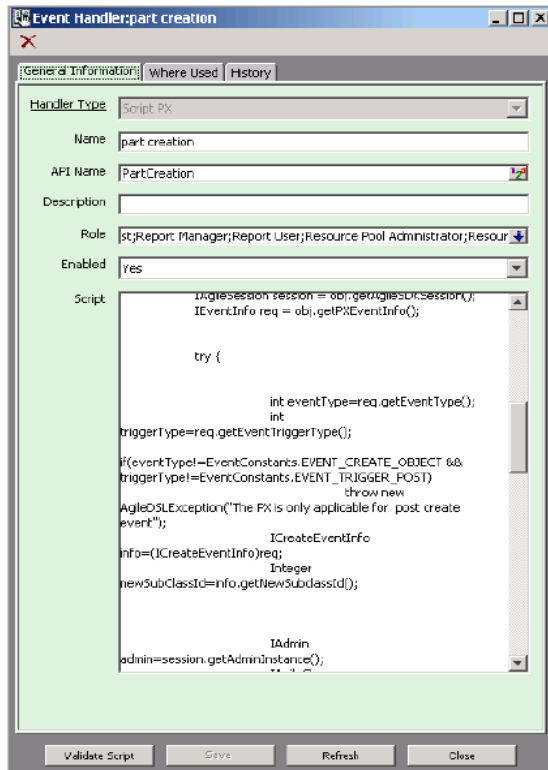
2. Click OK.

To set up the Event Handler mask:

1. Use the information in [Configuring Event Components for TemplateMgtStrucCreationPX](#) on page 25 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 PX in Doc-Publishing folder in SDK_samples.zip.
3. To configure your Script PX or Java PX Handler Type do as follows.
 - For Script PX Event Handler:


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*. You can also find information on configuring Script PXs in *Agile PLM Administrator Guide*. Following is an example of a Script PX Handler.

Figure 29: Creating a Script PX Handler for DataGenerationPX



- For Java PX Event Handler mask:
 1. Make sure the Event Action for this Java PX is deployed. See [Creating JAR Files and Deploying PXs \(Event Handlers\)](#).
 2. Check the values in DataGeneration.properties file and make sure they conform with Java Client Admin settings shown in [Properties File Settings for DataGenerationPX](#) on page 37.
- 1. 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 item creation with settings shown in the following figure.

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

Figure 30: Creating an Event Subscriber for DataGenerationPX

- 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.

Figure 31: Properties file settings for Data Generation 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 will hold
# the XML data in its attachment tab. Incase no value is specified for this property, the
# XML data will be added as an attachment to the object from which the "Data Generation"
# action is triggered.
# ACS_FILTER_ATTRIBUTE ----- This property defines the name of the ACS(Agile Content Service) filter to be used while
# generating 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
```

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 "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="";
```

Results from Running the 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 32: Document:D00006:A:Data.XML

```
<?xml version="1.0" encoding="UTF-8" ?>
- <AgileData>
- <Document>
- <TitleBlock>
  <number>D00006</number>
  <itemType>Document</itemType>
  <lifecyclePhase>Preliminary</lifecyclePhase>
  <description>TESTING BUILD#38</description>
  <itemCategory>Electrical</itemCategory>
  <size>A</size>
- <productLineS>
  <Value>Leo</Value>
  <Value>Pisces</Value>
</productLineS>
  <shippableItem>Yes</shippableItem>
  <thumbnail>http://DTP-VKOTHAND-WF.agile.agilesoft.com:8080/Filemgr/DownloadServlet?
    token=A0C64F834DA0E035C926E07C85F558A1C089674EF843B10137B261C4A08A3130DB6820E3D6A5ECBA0C887A723C33E6AB9326FE481B686C3728D0
  </thumbnail>
- <Attachments>
  <thumbnail>http://DTP-VKOTHAND-WF.agile.agilesoft.com:8080/Filemgr/DownloadServlet?
    token=9CE34EBDE5870CF43DCE3C365E0D104F5B30338632684A3BBAD019C9342183BDEF6157E8A9A78C64C3F02267D2B79C48520E6BD538872BD0F917
  <filename>Blue hills.jpg</filename>
  <fileSize>28521</fileSize>
  <fileType>jpg</fileType>
  <folderNumber>FOLDER00016</folderNumber>
  <folderVersion>1</folderVersion>
  <modifiedDate>2010-04-27T04:54:33Z</modifiedDate>
  <lastViewDate>2010-04-27T04:54:33Z</lastViewDate>
  <checkinUser>Administrator, admin (admin)</checkinUser>
</Attachments>
<BOM />
- <PageTwo>
  <DocType>HTML</DocType>
  <Filter>Itemtabs</Filter>
  <TemplateHolder>D00004</TemplateHolder>
</PageTwo>
</Document>
</AgileData>
```

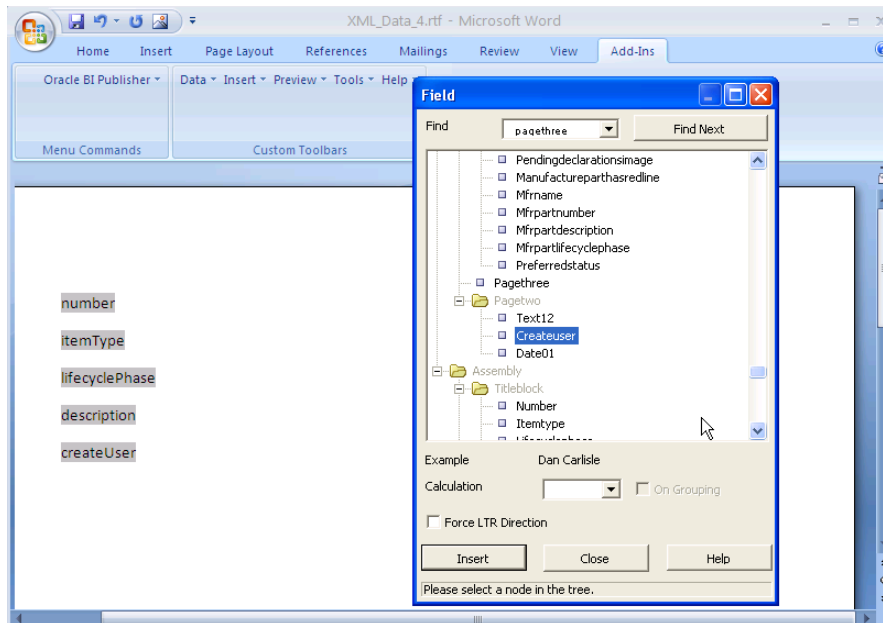
Building BI Publisher Templates

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 of these inserted tags.

Figure 33: 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.

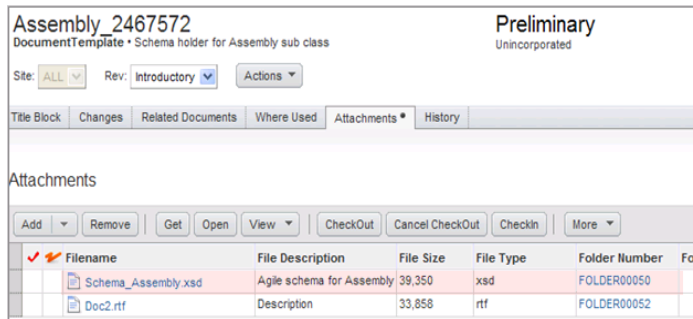
Copying XSD and XML Files to the Local Drive

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

To copy XSD and XML files to the local drive:

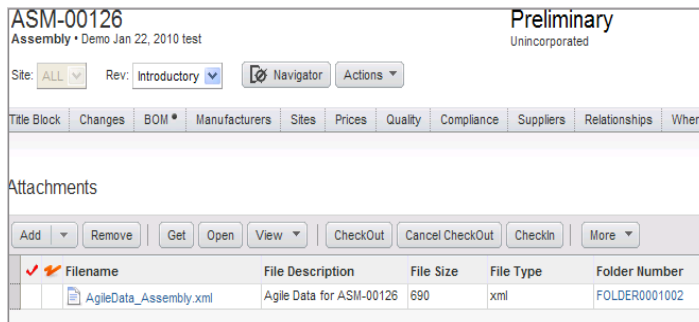
1. Select the XSD file and either click Get, or double-click on the file.

Figure 34: 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 file.

Figure 35: Copying XML File to the Local Drive



Loading Schema XSD and Data XML Using BI Publisher Plugin

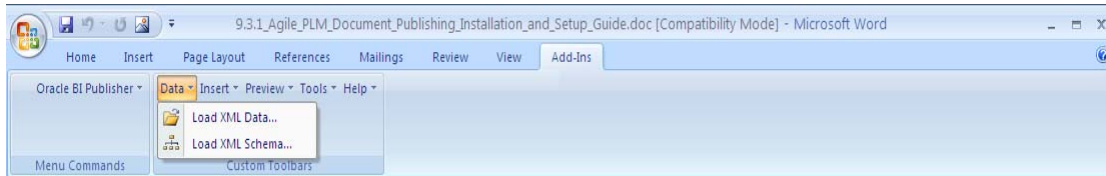
The following procedure assumes that you have already:

- Installed BI Publisher Desktop on your system
- Ran the SchemaGenerationPX and DataGenerationPX and created the Schema XML and Data XML files.

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 36: 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

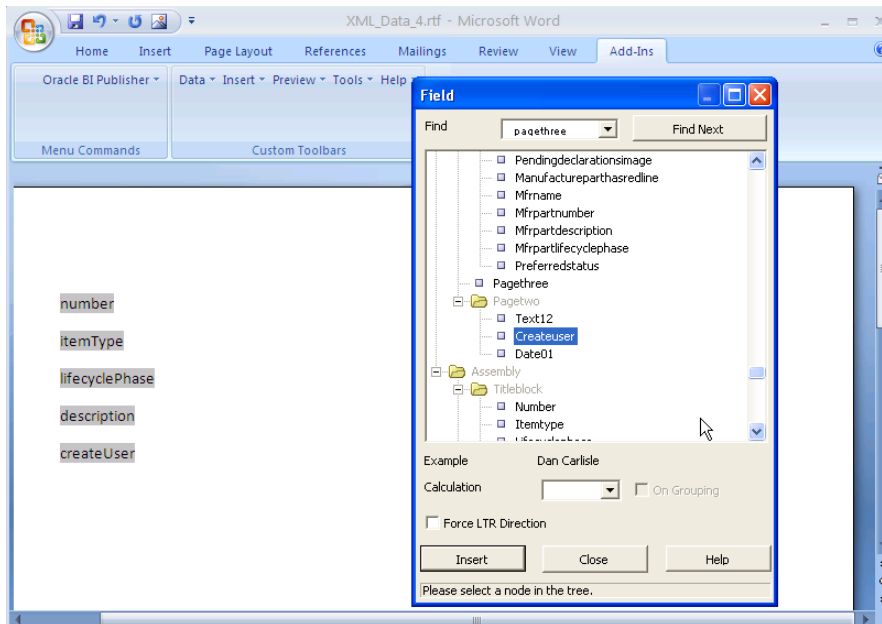
BI Publisher facilitates selecting Agile PLM data fields and provides extensive facilities to format the data and output document.

To configure the template

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.

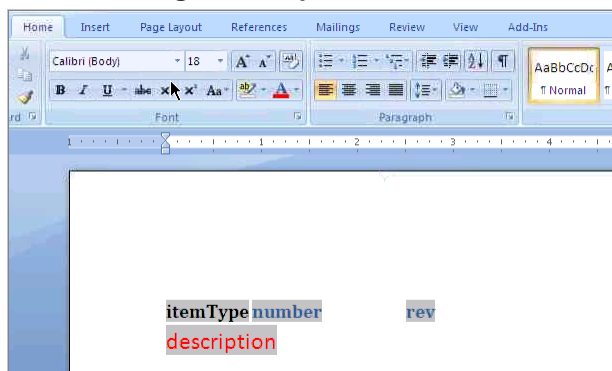
- Using Word features, customize the fonts and other formats of these inserted tags.

Figure 37: Building the BI Publisher Template



- When you complete the layout, save your Template as an RTF file.
- From the BI Publisher menus, select Preview Template > PDF (or any format) to preview the Data formatted in your Template.

Figure 38: Viewing the Template



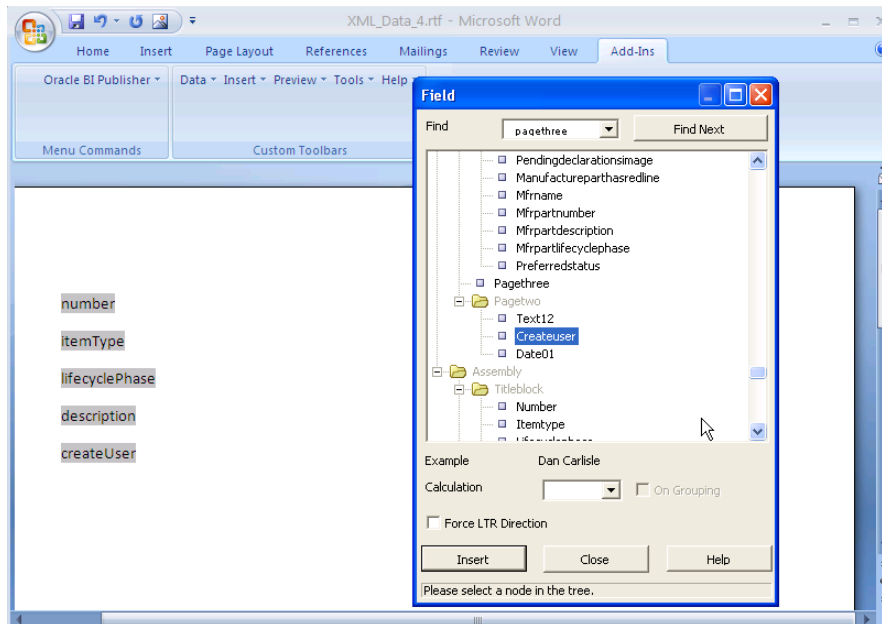
Inserting Agile PLM Data Fields in the Template

To insert Agile PLM data in the template:

- 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.
- 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.

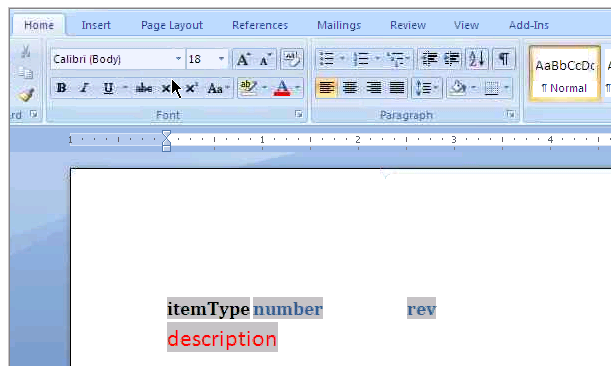
3. Scroll through the list, or use Find Next to select fields, for example, CreateUser.
4. Using Word features, customize the fonts and other formats of these inserted tags.

Figure 39: Building the BI Publisher Template



5. When you complete 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 40: Viewing the Template



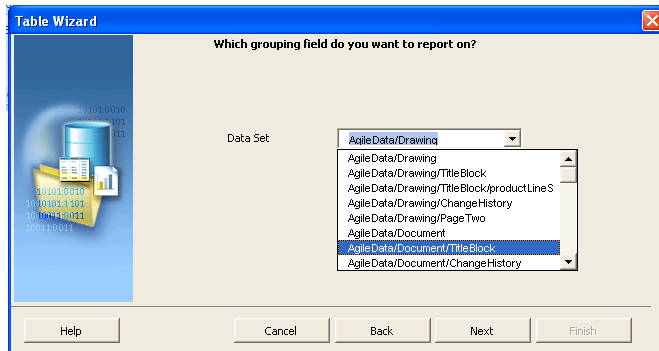
Inserting and Formatting Tables

Using BI Publisher, you can insert and represent Agile PLM fields in a tabular form. BI Publisher's Table formatting to 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](#) on page 2.

To present Agile data in tabular form:

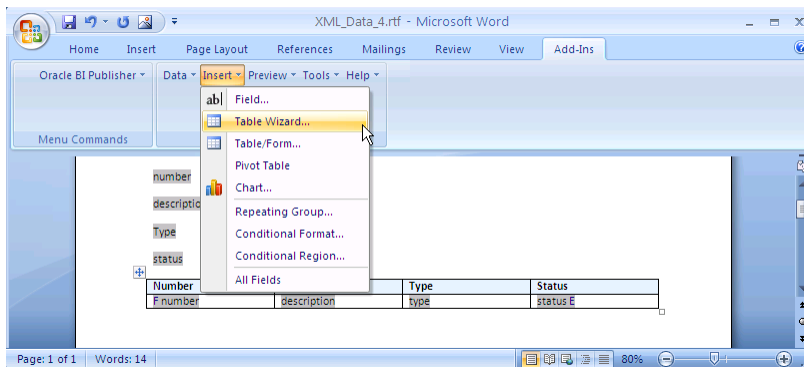
1. In Word with BI Publisher Desktop, select Add-Ins > Insert > Table Wizard. The Wizard prompts you to select the grouping fields that you want to report on.

Figure 41: Inserting and formatting tables



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

Figure 42: Represent Agile data in a 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 you are using do as follows to open the Alternative text box:
 - In Word 2007, right click the image and select Size > 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 43: Referencing a URL in earlier versions of Word

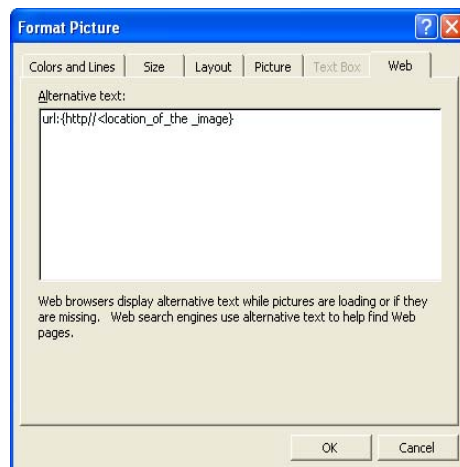
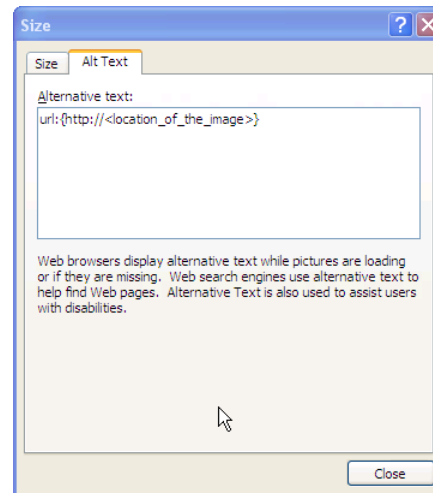


Figure 44: Referring URLs in Word 2007



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

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.

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:


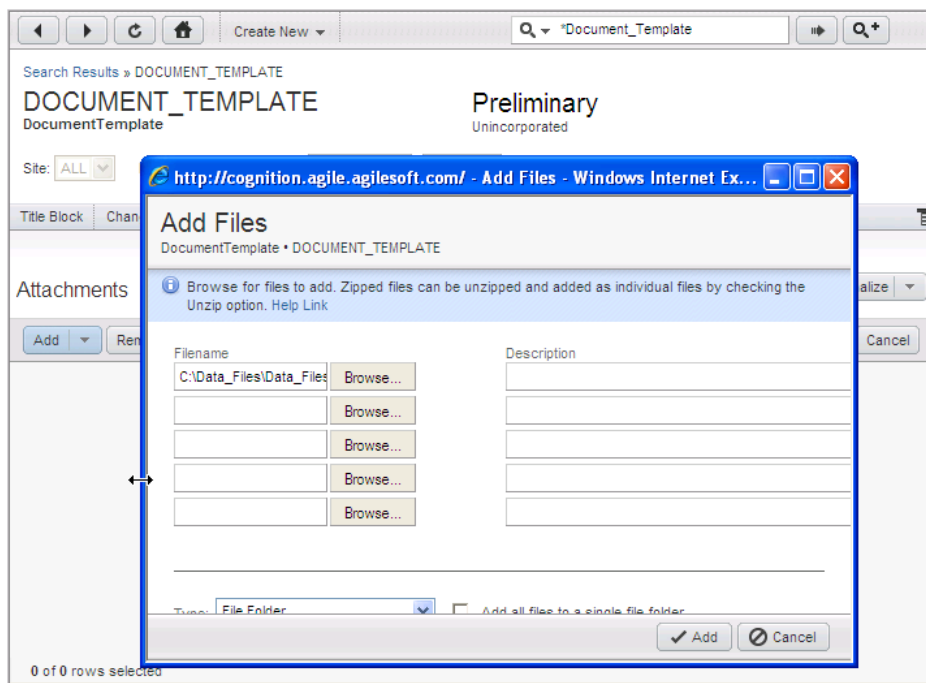
1. Log in to Agile PLM and select the folder you want to load the file into. In this case, DOCUMENT_TEMPLATE that you defined earlier.
2. Select Attachments > Add. The Add Files dialog opens.
3. In Add Files dialog, use Browse to locate the file on the local drive and then click the  button.

Figure 45: 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

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 Components 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 Components for TemplateMgtStrucCreationPX](#) on page 25 and define an Event called CreateDocument for Object Type Change Requests with the following settings:

Figure 46: Configuring Event type for DocumentGenerationJavaPX

The screenshot shows a 'Create Event' dialog box with the following fields and values:

- Event Type:** Change Status for Workflow (dropdown)
- Name:** CreateDocument (text field)
- API Name:** CreateDocument (text field)
- Description:** Create Document on release of change (text area)
- Enabled:** Yes (dropdown)
- Workflow:** Default: Change Requests (dropdown)
- Object Type:** Change Requests (dropdown)
- Status - From:** CCB (dropdown)
- Status - To:** Released (dropdown)

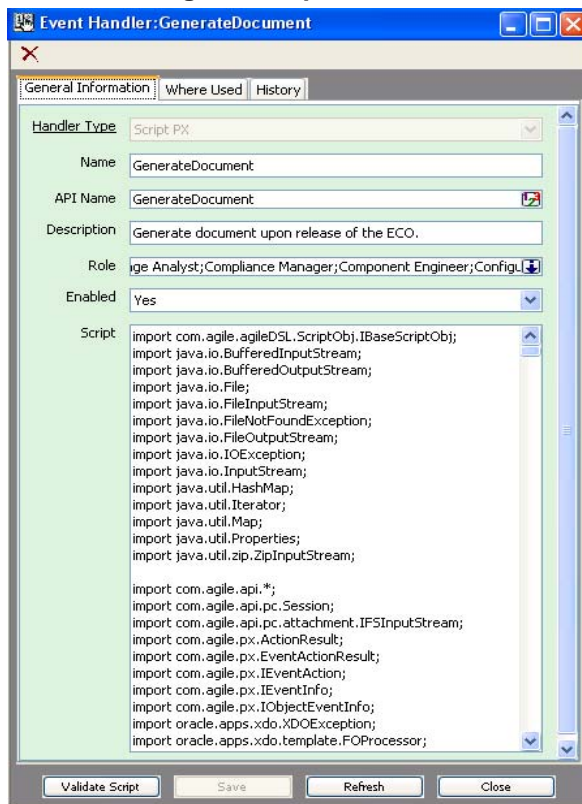
At the bottom are 'OK' and 'Cancel' buttons.

2. Click OK.

To set up the Event Handler mask:

1. Use the information in [Configuring Event Components for TemplateManagementStructureCreation PX](#) on page 25 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. 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*. You can also find information on configuring Script PXs in *Agile PLM Administrator Guide*. Following is an example of a Script PX Handler.

Figure 47: Creating the Script PX Handler for DocumentGenerationPX

- For Java PX Event Handler mask:
 - a. Make sure the Event Action for this Java PX is deployed. See Deploying PXs (Event Handler masks).
 - b. Check the values in DocumentGeneration.properties file and make sure they conform with Java Client Admin settings shown in [Properties File Settings for DocumentGenerationJavaPX](#) on page 50.
1. Click OK.

To configure the Event Subscriber mask:


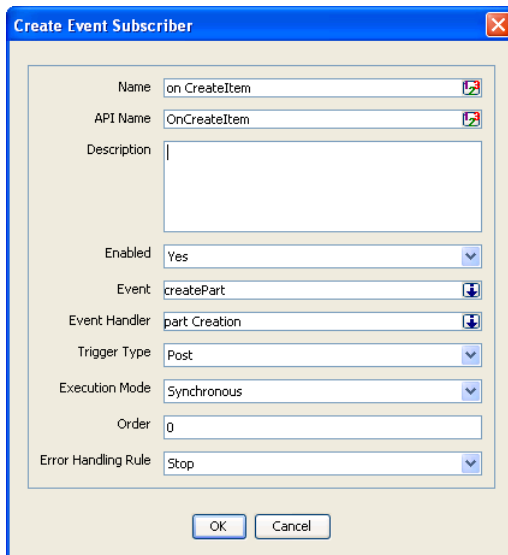
1. In Java Client with Admin privileges, select Admin > System Settings > Event Management > Event Subscribers.
2. In Event Subscribers pane, select the New  button to open the Create Event Subscriber dialog.
3. Create a new Event Subscriber called GenerateDocument with settings shown in the following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

Figure 48: Creating an Event Subscriber 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.

Note These Properties file settings also includes the ones for DocumentGenerationJavaPxOpen.

Figure 49: Properties file settings for Document Generation Java and JavaOpen 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

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

#Generated 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
logging              = true

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

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

Results from Running the 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

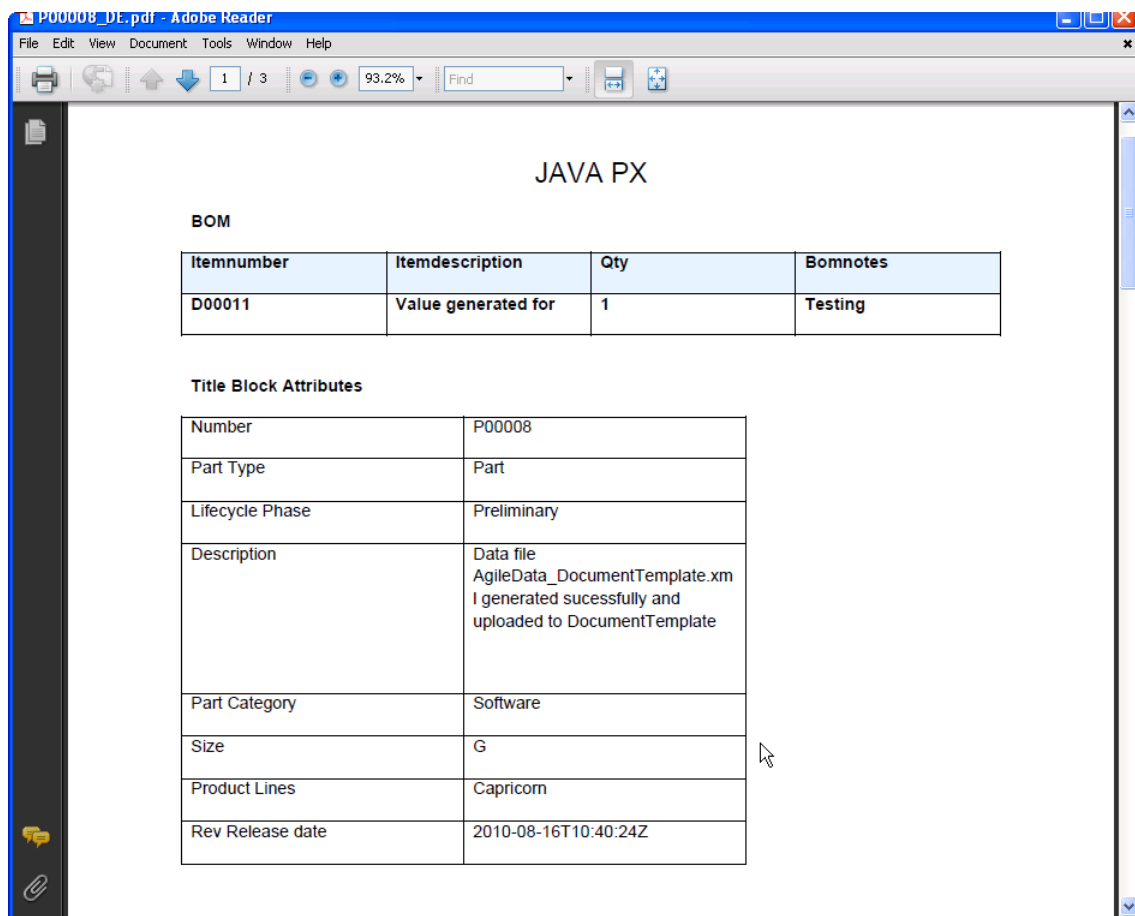
Note The PX creates a separate document object and attaches the output file to this object.

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.

Figure 50: DocumentGeneration PX output in PDF format




Note When creating Events, Event handlers and Event subscribers, you must enable the Event by clicking on 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.

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.

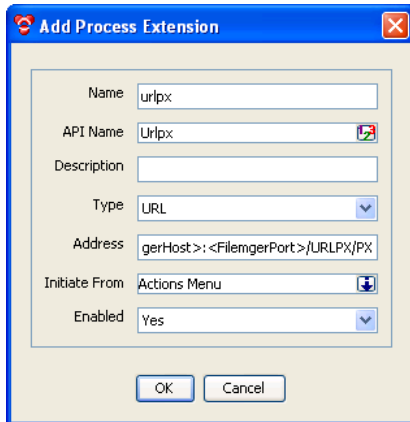
Use the following procedure to setup the URL PX.

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.

Note The Address field should point to the Filemanager. For example, <http://<filemgerHost>:<FilemgerPort>/URLPX/PX>.

Figure 51: Creating a URL PX



6. Click OK.

Properties File Settings for DocumentGenerationJavaOpen PX

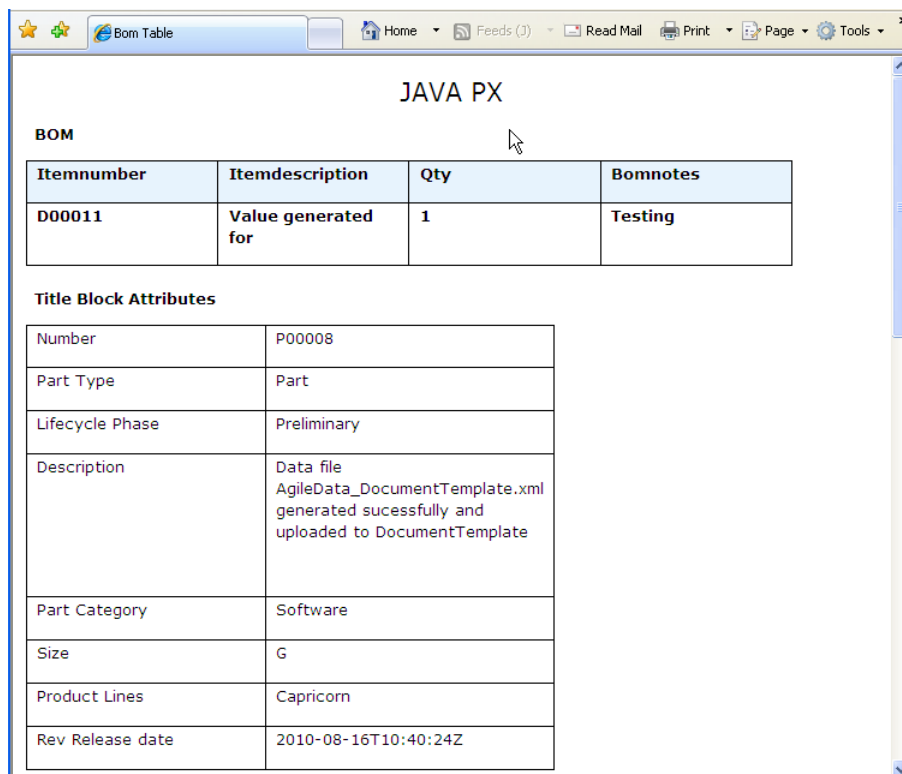
To configure file settings:

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 has been created.
3. Use these steps for other objects of interest.

DocumentGenerationJavaPxOpen Output Sample - Keep or Remove?

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

Figure 52: Output of the URL PX



JAVA PX			
BOM			
Itemnumber	Itemdescription	Qty	Bomnotes
D00011	Value generated for	1	Testing
Title Block Attributes			
Number	P00008		
Part Type	Part		
Lifecycle Phase	Preliminary		
Description	Data file AgileData_DocumentTemplate.xml generated sucessfully and uploaded to DocumentTemplate		
Part Category	Software		
Size	G		
Product Lines	Capricorn		
Rev Release date	2010-08-16T10:40:24Z		

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. As prerequisite, it requires assigning an existing object number for the TemplateHolder attribute.

Note The required prerequisites are defined in [Agile PLM Administrator Configurations](#) on page 14 and [Configuring DocumentGeneration PX](#) on page 48.

Modifying DocumentGenerationPX

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.

Note 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, modify the PX's Properties file as shown below.

Demonstrate how to alter the script to change the attachment locations

```
Public IItem 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:

1. Make sure the object to be processed has the correct configuration with Template, Filter, and output file type in Script PX Code, or Java PX Properties file. In this case, the settings are as shown below.
 - Output Type = PDF
 - ACS Filter = Itemstabs
 - Template Holder = D-00004
2. Trigger the Event (release the ECR).

Figure 53: Releasing the ECO to trigger the Event

The screenshot shows the Oracle ECR interface for item R00007. The status is 'Released'. The 'Affected Items' table lists the following item:

Item Number	Item Description	Revision	LifeCycle
ASM-00126	Demo Jan 22, 2010 test		

3. Open the resulting document.

Figure 54: Creating the output file

