

Document Number:

Date: 2013-11-05

Version:

Document Type:

Document Status of this readme file:

Document Status of DSP content within DSP2023: DMTF Standard

Document Language:

Copyright Notice

Copyright © 2008-2013 Distributed Management Task Force, Inc. (DMTF). All rights reserved.

DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems management and interoperability. Members and non-members may reproduce DMTF specifications and documents, provided that correct attribution is given. As DMTF specifications may be revised from time to time, the particular version and release date should always be noted.

Implementation of certain elements of this standard or proposed standard may be subject to third party patent rights, including provisional patent rights (herein "patent rights"). DMTF makes no representations to users of the standard as to the existence of such rights, and is not responsible to recognize, disclose, or identify any or all such third party patent right, owners or claimants, nor for any incomplete or inaccurate identification or disclosure of such rights, owners or claimants. DMTF shall have no liability to any party, in any manner or circumstance, under any legal theory whatsoever, for failure to recognize, disclose, or identify any such third party patent rights, or for such party’s reliance on the standard or incorporation thereof in its product, protocols or testing procedures. DMTF shall have no liability to any party implementing such standard, whether such implementation is foreseeable or not, nor to any patent owner or claimant, and shall have no liability or responsibility for costs or losses incurred if a standard is withdrawn or modified after publication, and shall be indemnified and held harmless by any party implementing the standard from any and all claims of infringement by a patent owner for such implementations.

For information about patents held by third-parties which have notified the DMTF that, in their opinion, such patent may relate to or impact implementations of DMTF standards, visit <http://www.dmtf.org/about/policies/disclosures.php>.

CONTENTS

[Foreword 4](#_Toc350287399)

[Acknowledgments 4](#_Toc350287400)

[Introduction 5](#_Toc350287401)

[1 Setting up the MRP development environment 6](#_Toc350287402)

[Step 1: Obtaining and installing the DSP2023 archive 6](#_Toc350287403)

[Step 2: Installing the CIM-XML files 6](#_Toc350287404)

[Step 2.1 (optional): Customizing the location of the cim-xml directory 8](#_Toc350287405)

[Step 3: Selecting and setting up a Web browser 8](#_Toc350287406)

[Step 4: Selecting and installing XML editor and build environment 9](#_Toc350287407)

[Step 4.1: Installing the make-based build environment 10](#_Toc350287408)

[Step 4.2: Installing Eclipse 10](#_Toc350287409)

[Step 4.3: Installing IBM Rational Software Architect 13](#_Toc350287410)

[Step 5: Registering the XML Catalog with the XML editor 14](#_Toc350287411)

[Step 6: Installing Python 15](#_Toc350287412)

[Step 7: Installing a PDF generator 15](#_Toc350287413)

[Step 8: Testing the build environment 16](#_Toc350287414)

[Step 8.1: Testing the make-based build environment 16](#_Toc350287415)

[Step 8.2: Testing the Eclipse-based build environment 17](#_Toc350287416)

[2 How to develop MRP XML profiles 19](#_Toc350287417)

[ANNEX A Content of DSP2023 archive 21](#_Toc350287418)

[A.1 xmp<NNNN> Directories: Sample profiles 21](#_Toc350287419)

[A.2 resources Directory: MRP tooling referenced by MRP XML profiles 22](#_Toc350287420)

[A.3 cim-xml Directory: CIM-XML files of underlying schemas 22](#_Toc350287421)

[A.4 localcopy Directory: MRP tooling redirected via an XML catalog 22](#_Toc350287422)

[A.5 bin Directory: MRP tooling in command line path 23](#_Toc350287423)

[ANNEX B Adding to PATH 24](#_Toc350287424)

[ANNEX C Change log 25](#_Toc350287425)

Foreword

This document is "readme" information that is part of (DSP2023). DSP2023 and this document were prepared by the of DMTF.

DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems management and interoperability. For information about the DMTF, see <http://www.dmtf.org>.

Acknowledgments

The DMTF acknowledges the following individuals for their contributions to DSP2023:

Andreas Maier, IBM (editor)

Michael Johanssen, IBM

Bob Blair, AMD

Cathi Biagi, Brightleaf Group

George Ericson, EMC

Steve Hand, Symantec

Introduction

DSP2023 is a ZIP archive that supports the development of management profiles in XML (also called "machine readable profiles" (MRP), or "MRP XML profiles"). The DSP2023 archive contains this document, MRP tooling needed for the development environment, and sample MRP XML profiles.

This document describes how to set up a development environment for MRP XML profiles using the tools provided in the DSP2023 archive, and how to use that environment for editing MRP XML profiles and for generating PDF files from the MRP XML profiles for subsequent publishing.

The target audience for this document is editors of MRP XML profiles.

# Setting up the MRP development environment

This clause describes how to set up a development environment for MRP XML profiles using the tools provided in the DSP2023 archive. Using the tools provided in the DSP2023 archive is not generally a hard requirement for editing MRP XML profiles, but so far it is the only choice.

The subclauses of this clause are steps that need to be taken (mostly in order) to get to the end goal of a development environment for MRP XML profiles. If you want to do less than editing MRP XML profiles, a subset of the steps may be sufficient. Also, some steps offer choices.

Step 1: Obtaining and installing the DSP2023 archive

This document is contained in the DSP2023 archive, so it is likely that you have already performed this step. In case you got hold of this document without unpacking the DSP2023 archive, this subclause describes how to obtain and unpack it.

This version of this document has been written for version 1.1 of MRP. The corresponding DSP2023 archive can be downloaded from the public part of the DMTF site using this link:

<http://www.dmtf.org/standards/published_documents/dsp2023_1.1.zip>

This link specifies only major and minor version number and resolves to the latest published update version of the DSP2023 archive for that major and minor version.

**Download the DSP2023 archive** to an empty directory that you want to use for MRP profile development, for example: "C:\DMTF\MRP" on Windows or "~/dmtf/mrp" on Linux.

A note on blanks in file and directory names: The Eclipse-based build environment supports having blanks in the path name of the DSP2023 unpack directory and in the names of profile files and directories in the subtree below it. The make-based build environment also supports having blanks in the path name of the DSP2023 unpack directory, but does not support having blanks in the names of profile files and directories in the subtree below it.

**Unpack the archive** using any UNZIP utility, into that same directory. It is important that the subdirectory structure is maintained when unpacking the archive; most UNZIP utilities do that by default. This unpack directory is referred to as the "DSP2023 unpack directory" and its full path name as "<DSP2023-unpack>" in the following subclauses. See ANNEX A for a description of the content of the DSP2023 archive.

**Add** the "<DSP2023-unpack>/bin" path name to the PATH (see ANNEX B for instructions).

Step 2: Installing the CIM-XML files

The DSP8029 XSLT stylesheet that is used to convert MRP XML profiles to HTML needs a definition of the CIM schema that contains the CIM classes used by the MRP XML profile. That schema definition needs to be provided in CIM-XML format, with one CIM class per CIM-XML file, in a well-defined subdirectory structure that is described below.

The CIM classes may be part of the CIM Schema published by DMTF, or may be part of other schemas defined by vendors or other standards organizations.

The DSP2023 archive does not contain any such CIM-XML files.

The CIM-XML files of the CIM Schema published by DMTF can be downloaded from the public CIM Schema page of the DMTF site, starting at this link:

<http://dmtf.org/standards/cim/schemas>

Navigate to the CIM Schema release or releases that you want to use; decide whether you want to use the experimental or the final schema (or both); and use the ZIP archives for individual XML classes. In newer CIM Schema releases, the download link is labeled "Zip archive of individual Final/Experimental XML Classes"; in older releases, the link is labeled with the file name, e.g. "cim\_schema\_2.29.0Final-XMLClasses.zip".

The CIM Schema page of DMTF provides versions back to 2.21. If earlier CIM Schema versions are needed, they can be downloaded from the DMTF Architecture WG page from a folder with this link:

<http://members.dmtf.org/apps/org/workgroup/technical/dmtf-arch/documents.php?folder_id=1521>

The ZIP files in the Architecture WG folder cover CIM Schema versions 2.10 to 2.29, and already contain the required subdirectory structure for the cim-xml directory subtree. These ZIP files typically contain a consecutive range of CIM Schema versions, where the lowest and the highest version number is stated in the file name.

**Download the desired ZIP files** to any directory on your system, but do NOT unpack them yet.

**Create a subdirectory tree** under the cim-xml directory, for each final and experimental schema version you want to use, as follows:

<DSP2023-unpack>/profiles/resources/cim-xml/<schema>/<version>

Where:

<schema> is the schema name of the schema, for example "CIM".

<version> is the major.minor version of the schema, followed by nothing for the final schema, and by "+" for the experimental schema. The update version indicator is not part of that version, in order to support an automatic upgrade of the update version. The assumption is that you have the latest published update version in that directory. For example, "2.22+" for 2.22 experimental.

Examples:

cim-xml/CIM/2.35 Directory for CIM Schema 2.35 final

cim-xml/CIM/2.35+ Directory for CIM Schema 2.35 experimental

cim-xml/ACME/1.0 Directory for ACME vendor schema 1.0 final

cim-xml/ACME/1.0+ Directory for ACME vendor schema 1.0 experimental

These directories then each need to contain the CIM-XML files for the respective schema version.

**Unpack the downloaded ZIP files** as follows:

ZIP files downloaded from the CIM schema page of the DMTF site directly contain the CIM-XML files without further subdirectories, so they need to be unpacked into the version subdirectories described above.

ZIP files downloaded from the folder of the Architecture WG contain the subdirectories below cim-xml in the required format, so they need to be unpacked into the cim-xml directory.

If you are unsure as to which CIM Schema versions you need, start with the versions you will need for the sample MRP XML profiles in DSP2023. These are:

2.10 final

2.19 final

2.22 experimental

2.22 final

As you work with MRP XML profiles over time, you may need to install further schema versions. Simply repeat the steps in this subclause for the new schema version.

Step 2.1 (optional): Customizing the location of the cim-xml directory

The default location of the cim-xml directory works as described above; and this step is not normally needed.

If you have a need to change its location for some reason (e.g., sharing it among multiple team members), this subclause describes how to customize its location. You may want to change the location in all the places described below.

When MRP XML profiles are opened with a Web browser, the location of the cim-xml directory is determined by the "cim-xml-root-url" parameter in the DSP8029 XSLT stylesheet, which is the file:

<DSP2023-unpack>/profiles/resources/dsp8029\_\*.xsl.

This parameter has a description that provides further details, such as the format to be used.

When MRP XML profiles are converted to HTML using the mrp2html.py script, the location of the cim-xml directory is specified via its command line option --cimxmlurl. The script then passes that directory path on to the DSP8029 XSLT stylesheet. The directory name to be used when invoking the script can be customized for both build environments (Eclipse-based and make-based) via the cimxml\_dir property in the following file:

<DSP2023-unpack>/profiles/resources/profiles.properties

Step 3: Selecting and setting up a Web browser

A Web browser can be used to open the MRP XML file (without first converting it to HTML), and the browser will then display human readable HTML. The conversion from MRP XML to HTML is performed by the browser and includes XSLT stylesheet processing, XPATH processing within the XSLT stylesheet, CSS stylesheet processing, interpreting image files specified using Data URIs, and JavaScript execution. Some Web browsers do not support all of these features sufficiently.

Using a Web browser to view the HTML file generated by the mrp2html.py script reduces the requirements on browser features somewhat, but not significantly. Also, when editing an MRP XML file, it is most convenient to look at the human readable result by opening that same MRP XML file in a Web browser and occasionally refreshing the page. For this reason, it is recommended to use a Web browser that supports displaying the MRP XML format without restrictions.

The following list shows Web browser versions that have been tested to work; some have restrictions as explained below. In this step, you should make sure that you use one of the Web browsers that supports opening the MRP XML files directly, without restrictions:

Firefox 10.0 and later. Extra setup required; see note 1).

Opera 11 and later

Safari 5 and later

Google Chrome 24. Restrictions apply; see note 2).

Internet Explorer 8 and later. Restrictions apply; see note 3).

Note 1) Extra setup for Firefox:

Firefox requires a particular setting to permit the use of the parent directory ("..") in relative URIs. (Relative URIs may be used for example in the xml-stylesheet processing instructions of MRP XML files, or in the cim-xml-root-url" parameter of the DSP8029 XSLT stylesheet.)

**Set the Firefox setting** "security.fileuri.strict\_origin\_policy" to the value "false". Firefox settings can be edited when directing the browser to the address about:config. For details about this setting, see <http://kb.mozillazine.org/Security.fileuri.strict_origin_policy>.

You can **verify that this works** by opening one of the sample MRP XML profiles (double check that it specifies a parent directory in its xml-stylesheet processing instruction, to be found somewhere in the first few lines of the file). If the familiar formatted title page of a profile document appears, the setting works.

Note 2) Restrictions for Google Chrome 24:

Chrome 24 displays the generated MRP HTML files with a minor flaw (partly incorrect clause numbering).

However, it does not convert the MRP XML files to HTML at all – it has no XSLT engine.

Note 3) Restrictions for Internet Explorer 8:

* 1. IE 8 does not support CSS counters and "content" attributes. Therefore, it does not display heading numbers in headings, table numbers in table captions, and figure numbers in figure captions.
  2. IE 8 has a size limit of 32 kB for Data URIs. This restriction can lead to truncated images when opening the generated HTML files (which use Data URIs for embedding the images into the HTML file). The truncation does not happen when opening the MRP XML files directly. IE 9 has lifted that size limit to 4 GB, see <http://blogs.msdn.com/b/ieinternals/archive/2010/09/15/ie9-beta-minor-change-list.aspx>.
  3. IE 8 does not support the JavaScript used to dynamically create the table of contents, table of figures and table of tables in the generated HTML. Therefore, it displays the original text "[Insert table of <toc-type> here ..." instead of the generated tables.

Internet Explorer 9 has lifted all of these restrictions (requires Windows 7 or higher).

Step 4: Selecting and installing XML editor and build environment

There is a large variety of XML editors that can be used for editing MRP XML files. The XML editor should satisfy the following requirements:

Support for XML Catalogs (see Step 5 for the benefits and why this is a hard requirement)

Support for displaying XSD annotations as user help. The MRP XSD files are written to provide usage help on each XML element and XML attribute in the form of annotations.

The following XML editors have been tested with MRP XML files:

Altova XMLSpy 2008 (Professional Edition)

XML package of Eclipse 4.3

XML package of IBM Rational Software Architect 8.5

The choice of build environment is related to the choice of XML editor. The DSP2023 archive supports two build environments:

Make-based build environment.

This command line centric environment uses the make utility to create PDF files from MRP XML files, and to package pictures from Visio drawings into a ZIP archive for easier uploading to a DMTF WG page or to some other source repository. The DSP2023 archive provides a makefile and scripts for this environment.

An XML editor would typically be a standalone tool in this environment (e.g., Altova XMLSpy).

Eclipse-based build environment.

This is an integrated environment where building can be triggered by menu actions and an XML editor is an integrated part of the environment.

The Eclipse-based build environment comes in two variants:

* + - * The Eclipse open source software
      * The IBM Rational Software Architect product. IBM provides Rational Software Architect free of charge for work related to standards development. For details, see the RSA/RSM entry on the DMTF tools page: <http://members.dmtf.org/members/tools/>

Step 4.1: Installing the make-based build environment

This step is only needed if you decide to use a make-based build environment along with a standalone XML editor.

The makefile provided in the DSP2023 archive is supported for Windows and Linux.

**Have the commands available** in the PATH that are used by the makefile provided by the DSP2023 archive. These are:

make (GNU make)

rm

cp

zip (any ZIP program)

These commands can be made available on Windows by installing CygWin 1.7 (or higher) from:

<http://cygwin.com/install.html>

When using the installer program of CygWin, the PATH environment variable gets updated automatically. Otherwise, the CygWin bin directory needs to be added to the PATH (see ANNEX B for instructions).

**Have your XML editor of choice available**.

**Test** the build environment is described in Step 8.

Step 4.2: Installing Eclipse

This step is only needed if you decide to use the Eclipse open source software as your XML editor and development environment.

This description is based on the latest Eclipse version available at this point, 4.2 or "Juno", but for purposes of editing and converting MRP XML files, there is no particular requirement for using the latest Eclipse version; earlier versions should work similarly.

This description uses Windows XP as an example install environment; Eclipse is also supported on higher Windows versions and on Linux.

Eclipse provides differently customized variants that can be installed, the simplest variant is "Eclipse Classic", and this description uses that variant. It should be sufficient to use the 32-bit version.

**Download** the 4.2 version of the "Eclipse Classic" variant from:

<http://www.eclipse.org/downloads/>

**Unpack the downloaded ZIP file** (for example: eclipse-SDK-4.2.1-win32.zip) into an empty directory that will be your permanent Eclipse installation directory, for example: "C:\DMTF\Eclipse‑4.2".

This description refers to that directory as the "Eclipse unpack directory" and its full path name as "<Eclipse-unpack>".

**Create a Windows shortcut** to the file <Eclipse-unpack>/eclipse/eclipse.exe , e.g., on your Windows desktop; this allows you to start Eclipse easily by double clicking on the shortcut icon.

**Start Eclipse** (e.g., via the shortcut).

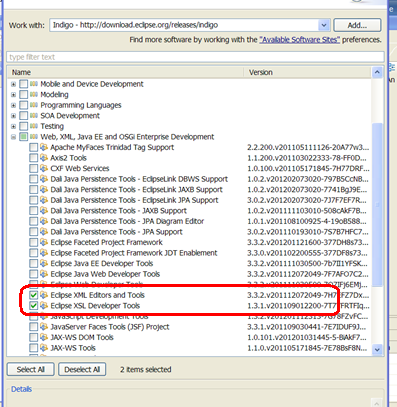
When started, Eclipse asks you to select a “workspace”. A workspace is a directory subtree where Eclipse keeps its configuration files. Eclipse projects created in context of a workspace are by default created as subdirectories in the workspace directory, but they can also be located elsewhere.

**Select** an empty directory as the Eclipse workspace (e.g., "C:\DMTF\MRPWorkspace").

In Eclipse, **install the XML and XSL extensions**, as follows:

**Open menu** “Help” -> “Install new software…”

**Select the install site** “Juno - <http://download.eclipse.org/releases/juno>” or add that install site if not yet listed. This populates the list of possible installation items.

Under the “Web, XML, Java EE and OSGi Enterprise Development” item, **select** “Eclipse XML Editors and Tools” and “Eclipse XSL Developer Tools”. This selection is shown in the screen shot below:  
  


**Confirm** the installation by pressing “Finish” and approve subsequent requests.

At the end of the installation process, **confirm** an Eclipse restart.

**Update** Eclipse via Help -> Check for Updates.

In Eclipse, **create a project** for MRP that uses the existing files in the DSP2023 unpack directory, as follows:

**Open menu:** File -> New -> Project, and then **select:** General -> Project

**Specify** a project name, for example "MRPProject".

**Specify** to use the DSP2023 unpack directory as the location for the project.

In Eclipse, **refresh the project** you just created, as follows:

In the Package Explorer, **right click** "MRPProject" and **select** "Refresh".  
This synchronizes knowledge about the DSP2023 files into Eclipse.

**Integrate** the mrpbuild.ant script provided in the DSP2023 archive into Eclipse, as follows:

**Register the Ant script** <DSP2023‑unpack>/bin/mrpbuild.ant as an external tool in Eclipse, as follows:

* + - **Open menu:** Run -> External Tools -> External Tools Configurations
    - **Right click** "Ant Build", select **New** and enter information as follows:

Name: mrpbuild  
Buildfile: ${workspace\_loc:/MRPProject/bin/mrpbuild.ant}  
Base Directory: (empty)  
Arguments: -Dmrp.xml="${selected\_resource\_loc}"  
Set an Input handler: checked

Assuming that the MRP project was named "MRPProject" (otherwise, change the path in the Buildfile field accordingly).

* + - **Finish** the registration by clicking on "Apply".

**Test** the Ant script is described in Step 8.

Step 4.3: Installing IBM Rational Software Architect

This step is only needed if you decide to use IBM Rational Software Architect (RSA) as your XML editor and development environment.

RSA is available for educational and standards-related work free of charge via the IBM Academic Initiative (formerly: IBM Scholar Program). For details related to members of DMTF and related standards organizations, see <http://members.dmtf.org/members/tools/OpenStandardsLetter-DMTF.doc>.

In order to download RSA via the IBM Academic Initiative, you must be a member of the IBM Academic Initiative. Refer to <http://www-03.ibm.com/ibm/university/academic/pub/page/membership> for how to become a member.

This description assumes you are downloading RSA via the IBM Academic Initiative. If you already have some version of RSA, skip the download and installation steps, but please double check that the XML and XSLT support is installed, as described below.

This description uses Windows XP as an example install environment; RSA is also supported on higher Windows versions and on Linux.

This description uses latest RSA version available at this point (8.5), but for purposes of editing and converting MRP XML files, there is no particular requirement for using the latest RSA version; earlier versions should work similarly. IBM Rational Software Modeler (RSM) also works.

RSA is an enhanced version of Eclipse; RSA 8.5 is based on Eclipse 3.6 ("Helios"). The installation instructions described in Step 4.2 apply, except for different download sites and for the additional requirement to install a license.

**Follow** all the steps in Step 4.2, with these exceptions:

**Download of RSA:**

**Sign in** to the "Software Download Catalog" page, using your IBM ID:

<https://www14.software.ibm.com/webapp/iwm/web/reg/signup.do?source=scholars>

**Download** "Rational Software Architect for WebSphere Software v8.5"

**Installation of XML and XSLT tools:**

**Select** the install site for "Helios":

<http://download.eclipse.org/releases/helios>

This populates the list of possible install items.

Under the “Web, XML, Java EE and OSGi Enterprise Development” item, **select** “Eclipse XML Editors and Tools” and “Eclipse XSL Developer Tools”.

**Installation of RSA license:**

Follow the instructions to download the license file, and to activate the license.

Step 5: Registering the XML Catalog with the XML editor

An XML Catalog is an XML file that follows the W3C XML Catalog format. XML Catalogs provide a means to redirect URLs that target files on the Web to local copies of these files. This provides a number of benefits, such as:

Local fixes to published files can be used

More control over which fix version is used

Using local files is faster than accessing them on the Web

Most importantly: Some websites that host XML schema files will temporarily reject the access if the same IP address requests the same file repeatedly. For example, the W3C site that owns some of the files used by MRP is known for this behavior. Using a local copy of the XSD files solves the issue by not going to the publishing website in the first place.

One can work in a local environment that has no Internet access.

The DSP2023 archive contains all XSD files needed by MRP, as well as an XML Catalog file that redirects the published URLs of these XSD files to the local copies. See A.4 for details.

XML Catalog files are utilized only by the XML editor that is used to edit the MRP XML file. The MRP conversion tools that generate HTML and PDF do not need them, nor does the Web browser used to open MRP XML files or the HTML file.

To make the XML editor use the XML Catalog file provided in the DSP2023 archive, that file needs to be registered with the XML editor.

Follow these steps, depending on the XML editor you decided to use in Step 4:

For Altova XMLSpy:

Add a nextCatalog XML element that references the MRP catalog.xml file in the DSP2023 unpack directory, for example:

<nextCatalog  
  catalog="file:///C:/DSP2023‑unpack/localcopy/catalog.xml"/>

to the end of the catalog list in the RootCatalog.xml file of XMLSpy. That file can be found in "C:\Program Files\Altova\XMLSpy2008".

Make sure that "file:" followed by three slash characters is used in that URL (XMLSpy has issues when just one slash character is used). On Windows, use forward slash characters as path delimiters in the catalog attribute shown above.

Eclipse (V4) or IBM Rational Software Architect (V8):

Open the XML Catalog settings dialog through **Window -> Preferences**, select **XML -> XML Catalog** and add a new entry of type "Next Catalog" that points to the catalog.xml file provided with DSP2023: "<DSP2023-unpack>/localcopy/catalog.xml".

If you do not see a settings subtree named "XML" in the Preferences dialog, you have not installed the XML package of Eclipse or RSA.

The registration of the XML Catalog file for MRP can be verified by opening one of the sample MRP XML files with your XML Editor when no Internet connectivity is available. A properly established XML Catalog will redirect the remote schema locations of .xsd files in that .mrp.xml file to local copies in the localcopy directory and will cause the XML editor to work without reporting any errors about missing XSD files.

Step 6: Installing Python

The MRP tooling provided in the DSP2023 archive contains Python scripts that need Python 2.6 or 2.7, and the Python package lxml 2.3.

Python is open source software and can be downloaded from:

<http://python.org>

**Download** the latest fix release of Python 2.6 (currently: 2.6.8) or of Python 2.7 (currently: 2.7.3) and install it into any directory on your system.

On Windows, the Python installer program does not add the Python installation directory to the PATH. **Add** the Python installation directory (the one containing the python.exe file, for example C:\Python27) to the PATH (see ANNEX B for instructions).

**Download and install** lxml 2.3 as follows:

For Windows, go to <http://pypi.python.org/pypi/lxml/2.3>, click the green Download button, which gets you to a list of possible downloads of lxml for Windows and MacOS. It is recommended to use the Windows Installer program for lxml, because that has the libxml2 library statically linked and thus is a standalone package without further dependencies. It is also important that the Python version of lxml matches the version of your Python installation, and that you pick the 32-bit or 64-bit version of lxml depending on what bit size your Python installation has.

Invoking "python" in the command line will tell you the version and bit size of your Python installation. You can leave the Python interpreter with "quit()".

For example, this Python invocation:

>python

Python 2.6.4 (r264:75708, Oct 26 2009, 08:23:19) [MSC v.1500 32 bit (Intel)] on win32

Type "help", "copyright", "credits" or "license" for more information.

>>> quit()

tells you that it is 32-bit Python 2.6 on Windows, and you would then select the following matching lxml download:

lxml-2.3.win32-py2.6.exe

After downloading the Windows Installer program, install lxml by invoking that program.

For Linux, go to <https://pypi.python.org/pypi/lxml/2.3.6>, and click the green Download button, which downloads the file lxml-2.3.6.tar.gz that is a source distribution that works for all Python 2.x versions. To install it, follow the instructions on the download page.

**Troubleshooting:**

If the Windows Installer program of lxml reports the error: "Python Version 2.7 required which was not found in the registry" but you have Python 2.7 installed, you may have picked the 32-bit version of lxml when you have the 64-bit version of Python installed.

Use the 64-bit version of lxml instead.

Step 7: Installing a PDF generator

A PDF generator is used to convert the HTML file generated by the mrp2html.py script into a PDF file. Standards organizations that are publishing profiles as PDF files need a high quality PDF generator that produces a PDF file that is comparable in quality to writing a document by using a text processor, such as Microsoft Word, and then converting that file to PDF. Critical areas in terms of quality are image resolution and scaling, page headers and footers, and page layout. Also, because the HTML file that is generated by mrp2html.py represents pictures as Data URIs, and contains embedded CSS stylesheet and JavaScript, the PDF generator will need to support all that.

The only PDF generator known so far that provides a sufficient quality and supports these requirements is Prince-XML. The DSP8054 CSS stylesheet at this point has some dependencies on features specific to Prince-XML.

Prince-XML is a commercial product that also has a free-of-charge community edition. The community edition is fully functional, but generates a small Prince logo on the top right corner of the title page of the PDF document. The commercial edition does not generate that logo. Using the community edition for developing standards is considered acceptable from a licensing perspective. DMTF has decided to use the community edition during development of DMTF-owned profiles, and the commercial edition for publishing the profiles.

Download the Prince-XML PDF generator from:

<http://www.princexml.com/>

At this point, Prince-XML 8.1r4 is the latest version available. That version has been tested and works fine, but runs considerably slower than the 8.0 version. The bugs fixed by the 8.1r4 version do not provide significant advantages for MRP XML profiles. Therefore, it is recommended to use the 8.0 version for now.

The install program is the same for the commercial edition and the community edition. If you want to have a commercial edition, acquire a license and install the license key.

The Prince-XML download page offers no link to the 8.0 version anymore, but 8.0 can still be downloaded from this page:

<http://www.princexml.com/download/8.0/>

Prince-XML can be installed to any directory.

The Prince-XML executable is found in both build environments (make-based and Eclipse-based) via the prince\_file property in the following file:

<DSP2023-unpack>/profiles/resources/profiles.properties

The value for this property as contained in the DSP2023 archive works unchanged for the default installation path of Prince-XML on Windows XP. On Linux, and on Windows when installing Prince-XML in some other location (for example, in the default installation directory of Windows 7), **edit this file and customize** the prince\_file property accordingly.

Step 8: Testing the build environment

Step 8.1: Testing the make-based build environment

**Run the following command** to test that the make-based build environment has been set up correctly:

make clean buildxmp

The command needs to be run from within the <DSP2023-unpack>/profiles directory.

The command removes any generated files that came with the original DSP2023 archive and then re-builds the HTML and PDF files for the sample profiles.

**Verify that**:

No error messages are displayed while executing that command (besides the few intentional "Profile Error" messages issued by DSP8029)

The PDF files for the sample profiles have been generated in their respective subdirectories

The generated PDF files have the following components:

* + - a formatted title page like published DMTF documents do (this verifies that the XSLT stylesheet has been found and executed),
    - a Table of Contents (this verifies that the JavaScript program executed properly)

The DSP2023 archive ZIP file has been generated in the current directory

The makefile has a default target "help", which prints information about the available make targets.

Step 8.2: Testing the Eclipse-based build environment

To test the Eclipse-based build environment, start Eclipse or RSA.

In the Project Explorer of Eclipse or RSA, **select** one of the sample MRP XML files unpacked from DSP2023.

The first time you run the "mrpbuild" tool, this must be done from within the External Tools Configuration dialog: **Open** menu Run -> External Tools -> External Tools Configurations, **select** the "mrpbuild" tool and **click** on "Run".

Subsequently, the "mrpbuild" tool will be available as a menu item: Run -> External Tools -> mrpbuild.

Running the "mrpbuild" tool executes the mrpbuild.ant script, which creates an HTML file and a PDF file for the selected MRP XML file in the directory of the selected MRP XML file.

Error messages issued by the "mrpbuild" tool can be seen in the Console pane of Eclipse.

**Verify** the execution process and its result as described in Step 8.1.

**Troubleshooting:**

If upon running the mrpbuild tool, you encounter the following error message (usually as a popup window):

Unable to resource a selected resource: ${selected\_resource\_loc}

the reason may be that you did not select an MRP XML file. Note that Eclipse forgets about the selection after a while, even though the file still appears to be selected in the Eclipse Explorer pane.

Select an MRP XML file and retry.

If upon running the mrpbuild tool, you encounter the error message in the Console pane:

. . . mrpbuild.ant:{NN}: Error: Selected file does not have a file  
extension .mrp.xml or \_mrp.xml: '{filename}'

the reason is that the selected file name does not have one of the supported endings (.mrp.xml or \_mrp.xml). The \_mrp.xml file name ending is supported in order to tolerate the renaming some download tools perform when downloading files that have more than one dot in their file name.

Rename the MRP XML file to end with .mrp.xml or \_mrp.xml, and retry.

If upon running the mrpbuild tool, you encounter the error message in the Console pane:

prince: . . . .pdf: error: The process cannot access the file because it is being used by another process.

then you likely have the PDF file opened in a PDF reader and the PDF generator cannot write its output.

Close the PDF file or the PDF reader and retry.

If upon running the mrpbuild tool, you encounter the error message in the Console pane:

Buildfile: . . . \mrpbuild.ant  
checkSelection:  
html:  
 [exec] Traceback (most recent call last):  
 [exec] File "../bin/mrp2html.py", line 21, in <module>  
 [exec] from lxml import etree  
 [exec] ImportError: No module named lxml  
 [exec] Result: 1

then you do not have the lxml package installed in your Python installation.

Install the lxml package as described in Step 6 and retry.

If upon running the mrpbuild tool, you encounter the error message in the Console pane:

. . . mrpbuild.ant:{NN}: Error: Prince-XML executable not found:  
'{filename}'

then (besides not having Prince-XML installed at all), you likely have it installed in a different directory than the default directory. The default directory works for Windows XP, but not for Windows 7.

Change the value of the prince\_file variable in the file  
<DSP2023-unpack>/profiles/resources/profiles.properties  
to match your Prince-XML installation (see Step 7) and retry.

If upon running the mrpbuild tool, you encounter the error message in the Console pane:

Buildfile: C:\DMTF\MRP Profiles\bin\mrpbuild.ant  
checkSelection:  
html:  
 [exec] mrp2html.py Version 1.1.0  
 [exec] Processing MRP XML file: C:\DMTF\MRP Profiles\profiles\  
dsp1033\dsp1033\_1.1.0a.mrp.xml  
 [exec] Generating HTML file: C:\DMTF\MRP Profiles\profiles\  
dsp1033\dsp1033\_1.1.0a.html  
 [exec] Cannot resolve URI file:///C:/DMTF/MRP%20Profiles/  
profiles/resources/cim‑xml/CIM/2.10/CIM\_RegisteredProfile.xml  
 [exec] Found no XSLT transformation errors.  
 [exec] Error: Found 1 errors.  
 [exec] Result: 1

then you have not installed the desired version of the CIM Schema, or you have installed it into a directory that does not follow the subdirectory structured described in Step 2.

**Install** the desired CIM Schema version using the directory structure described in Step 2 and retry.

If upon running the mrpbuild tool, you encounter the error message in the Console pane:

Buildfile: C:\DMTF\MRP Profiles\bin\mrpbuild.ant  
BUILD FAILED  
Target "Profiles\profiles\xmp1033\xmp1033\_1.1.0a.mrp.xml" does not exist in the project "mrpbuild".

then you are likely using a draft version of DSP2023 1.1 that did not yet support blanks in the name of the DSP2023 unpack directory.

**Upgrade** to the latest DSP2023 1.1 version and retry.

If upon running the mrpbuild tool on a Work in progress version, and the resulting PDF contains "Not a DMTF Standard" twice in its footer line, then you have specified "Not a DMTF Standard" in the mrp:Confidentiality element (like it is done for Word-based profiles).

An occurrence of "Not a DMTF Standard" is created automatically into the footer line, based on the document status "Work in Progress", so there is no need to add this text to the mrp:Confidentiality element, in MRP profiles.

# How to develop MRP XML profiles

After setting up the MRP development environment as described in clause 1, you are ready for developing MRP XML profiles.

To start, it is recommended that you get familiar with the MRP XML format by looking at the sample MRP XML profiles, and by trying to relate statements in the MRP XML file with text in the HTML or PDF output generated from the MRP XML file.

Note that some subclauses in the HTML and PDF output (e.g., Adaptations, Use Cases) have large percentages of generated text and tables that are constructed from machine readable data in the MRP XML file (that is the whole idea of making the profile machine readable). Conversely, subclauses like clause 6 "Description" consist of text that is entirely in the corresponding MRP XML element for that subclause.

Any rich text in an MRP XML profile is written in XHTML. In the sample MRP XML profiles, the XML namespaces are set up such that the XHTML namespace is the default namespace. That setup allows writing XHTML tags without specifying a XML namespace prefix, e.g., as "<p>", and makes it very similar to HTML. One notable difference to HTML is that in XHTML, everything needs to be well-formed XML; that is, any opening tag needs to have a closing tag and tags need to be properly nested. The syntax to have a single tag without content (that is, an opening and closing tag), is: "<br/>".

A convenient way to verify the output is to open the MRP XML file in a Web browser. This causes the Web browser to convert the MRP XML to HTML, and then display that HTML.

The Web browser can stay open and can be used to refresh / reload the file after saving it in the XML editor.

The format of the MRP XML is described in annotations of the XSD files the XML editor will be able to locate when editing an MRP XML file. XML editors typically show these annotations in a context-sensitive way, so that you can learn which XML elements and XML attributes are allowed at a particular spot, and how they are used, by reading these annotations. Try to see these annotations when editing one of the sample MRP XML files.

Make changes to the sample MRP XML files, and see how they are visible in the Web browser that is displaying the MRP XML file you are editing, by saving the XML file and refreshing the browser.

To produce PDF files for publishing and review, use the build environment you have installed.

When using the make-based build environment:

**Open** a command window and go to the <DSP2023-unpack>/profiles directory.

**Run** the following command to build the PDF files for all sample profiles in DSP2023:

make buildxmp

**Run** the following command to build the PDF files for all non-sample profiles that have a DMTF DSP number that you may have added yourself:

make build

The makefile displays a short help about the targets it supports, when running:

make help

When using the Eclipse-based build environment (including RSA):

In the Project Explorer, **select** one of the sample MRP XML files, and click menu item: Run -> External Tools -> mrpbuild

This executes the mrpbuild.ant script, which creates an HTML file and a PDF file for the selected MRP XML file in the directory of the selected file.

Content of DSP2023 archive

This annex describes the content of the DSP2023 archive.

At a high level, the DSP2023 archive contains this readme document, tooling needed for a development environment for MRP XML profiles, and sample MRP XML profiles.

Unpacking the DSP2023 archive creates the following directory structure:

+ Unpack directory

|

+-- profiles

| |

| +-- xmp<NNNN> Directories with sample MRP XML profiles xmp<NNNN>

| |

| +-- resources Directory with MRP tooling (files referenced by

| | MRP XML profiles)

| |

| +-- cim-xml Directory with MRP tooling (CIM-XML files of

| underlying schemas, not included in DSP2023)

|

+-- localcopy Directory with MRP tooling (local copies of files

| on the Web, redirected to via an XML catalog)

|

+-- bin Directory with MRP tooling (files that need to

be in the command search path)

xmp<NNNN> Directories: Sample profiles

Each of these directories contains one sample MRP XML profile. The sample profiles are mostly high fidelity representations of real profiles published by DMTF that demonstrate certain aspects of MRP.

The sample profiles are:

**XMP1000:** MRP XML profile template

This file can be used as a template for new MRP XML profiles.

**XMP1009:** MRP version of DSP1009 (Sensors Profile)

This profile demonstrates how to include boilerplate text from the Management Profiles Text Registry (DSP8008), and the use of collaborations in diagrams.

**XMP1011:** MRP version of DSP1011 (Physical Asset Profile)

This profile demonstrates how to use abstract base adaptations.

**XMP1013:** MRP version of DSP1013 (Fan Profile)

This profile demonstrates how to define indications.

**XMP1033:** MRP version of DSP1033 (Profile Registration Profile)

This profile is a very high fidelity representation of the published DSP1033 v1.0.0 profile, and demonstrates how an MRP XML profile should be written, from all aspects. Specifically, it demonstrates the use of profile references for referencing the same profile (in this case, itself) more than once.

**XMP1999:** Sample glossary in MRP format

This profile demonstrates the use of the MRP format for representing a glossary document.

Each of the xmp<NNNN> directories (for example, xmp1000) contains the following files:

**xmp<NNNN>\_<m.n.u>.mrp.xml:** MRP XML profile

This is the source file of the profile, and is in the MRP XML format.

**xmp<NNNN>\_<m.n.u>.html:** HTML file representing the profile

This file is generated from the MRP XML profile using the mrp2html.py script contained in the DSP2023 archive. It is an intermediate product in the tools chain and has been included for completeness. The HTML files in the DSP2023 archive have been created in standalone mode, which causes any figures, JavaScript and CSS stylesheets to be embedded in the HTML file.

**xmp<NNNN>\_<m.n.u>.pdf:** PDF file representing the profile

This file is generated from the HTML file using an HTML to PDF converter, and can be published (depending on the quality of the HTML to PDF converter).

**xmp<NNNN>\_<m.n.u>\_<figname>.gif:** Picture files with figures for the profile

These files are picture files containing the figures referenced by <img> tags in the MRP XML profiles. They have been created by saving Visio diagrams in GIF format. The GIF files have been saved with 240 dpi resolution (the source size does not matter).

**xmp<NNNN>\_<m.n.u>.vsd:** Visio source file for the diagrams in the profile.

resources Directory: MRP tooling referenced by MRP XML profiles

The resources directory contains files that are considered MRP tooling and that are (potentially) referenced from MRP XML profiles in some way, and for which URI resolution through an XML catalog is not supported. This includes XSLT stylesheets (DSP8029), CSS stylesheets (DSP8054), Javascript files (tocgen.js), standard message registries (DSP8008), and the cim-xml subtree. Also, it includes a properties file (resources.properties) that allows for certain customizations and is used by both the make-based and the Eclipse-based build environment.

cim-xml Directory: CIM-XML files of underlying schemas

The cim-xml directory contains a subtree with the CIM schemas that are used by any MRP XML profiles. The subbtree may include the CIM schema published by DMTF, and other schemas. The cim-xml subtree is initially empty when unpacking the DSP2023 archive, and needs to be populated with schema files, as described in Step 2.

localcopy Directory: MRP tooling redirected via an XML catalog

The localcopy directory contains files that are considered MRP tooling and for which URI resolution using an XML catalog is supported. This includes XSD files (DSP8028, …), and the XML catalog itself (catalog.xml).

bin Directory: MRP tooling in command line path

The bin directory contains files that are considered MRP tooling and that need to be in the command line path, or that are referenced for performing some function. This includes a Python script (mrp2html.py) and an Ant script (mrpbuild.ant).

Adding to PATH

This annex describes how to add path names of directories to the command line execution search path, that is, the PATH environment variable. At this point, only Windows is covered.

On Windows, the path names in the PATH environment variable are separated by using a semicolon (";") and within the path names, the directory delimiter character is the backslash ("\"). The PATH variable may contain relative and absolute path names, but this document recommends using only absolute path names for anything that is added.

To edit the PATH variable on Windows XP, go to: Control Panel -> System -> Advanced -> Environment Variables, and edit the system variable named "Path". It is similar on Windows 7.

Add the absolute path name of the directory to the beginning of the PATH variable, and make sure you have a path delimiter character between it and the next path name.

Save the new value of the PATH variable.

Note that a change in the PATH system variable as described above may not be reflected in the PATH environment variable as seen in a newly opened command shell. It may take a restart of the desktop instance of explorer.exe to see the change (or a Windows reboot).

Change log

The change log applies to the DSP2023 archive. DSP2023 for convenience contains other MRP-related documents published separately by DMTF (e.g., DSP8028). The change description in the change log covers these included files as well and thus provides a complete overview about changes related to MRP.

|  |  |  |
| --- | --- | --- |
| Version | Date | Description |
| 0.9.1 | 2008-11-06 | Released as a first Work in Progress. |
| 0.9.3 | 2009-05-31 | Released as a second Work in Progress. |
| 1.0.0a | 2010-03-16 | Released as a third Work in Progress. |
| 1.0.0b | 2010-05-31 | Released as a fourth Work in Progress. |
| 1.0.0c | 2011-02-22 | Released as a fifth Work in Progress. |
| 1.0.0d | 2011-06-21 | Released as a sixth Work in Progress. |
| 1.0.0e | 2011-08-16 | Released as a seventh Work in Progress. |
| 1.0.0f | 2011-09-13 | Released as a Work in Progress. |
| 1.0.0 | 2012-03-19 | Released as DMTF Informational. |
| 1.0.1 | 2012-08-19 | Released as DMTF Informational.  Changes in DSP2023:  Added description for using Eclipse or IBM Rational Software Architect for editing MRP XML.  Simplified the MRP XML to HTML conversion by providing a mrp2html.py Python script, and removing the files that were used by the more complicated procedure in V1.0.0: bin/xalan.bat, bin/CatalogManager.properties, profiles/resources/embed-file.awk and profiles/resources/embed-css-file.awk.  Picked up DSP8028 v1.0.1, and DSP8029 v1.0.1.  Changes in DSP8028 v1.0.1:  Fixed issue with duplicate heading id values reported by parsers that validate the uniqueness constraint of the xsd:ID type, by changing the type of mrp:HeadingLink/@id from xsd:ID to xsd:IDREF. The referenced elements are the h1, h2, etc. elements in XHTML, which define their id attribute using the xsd:ID type.  Changes in DSP8029 v1.0.1:  Added tolerance for whitespace in the CIM-XML values of the following (whitespace-tolerant) qualifiers:  Boolean-typed qualifiers: ASSOCIATION, INDICATION, ABSTRACT, KEY, REQUIRED, IN, OUT.  Integer-typed qualifiers: MIN, MAX.  String-typed qualifiers: EMBEDDEDINSTANCE.  Added missing double quote in the message text of a profile error. |
| 1.1.0 | 2013-11-05 | Released as DMTF Informational.  Changes in DSP2023:  Changed readme file to Word format and simplified it.  Added support for a Linux based build environment.  Fixed issues in Eclipse-based build environment.  Updated information about Web browsers and PDF converters to cover their latest releases, including IE9.  In MRP tooling, improved ability to debug some errors in MRP tooling.  In MRP tooling, improved support for heading links (mrp:HeadingLink element) such that the text shown on these links is now the subclause number, or "Clause {clause-number}".  In MRP tooling, added support for table links (mrp:TableLink element) and for figure links (mrp:FigureLink element).  Removed batch script that helped migrating from pre-1.0.0 versions to 1.0.0.  Added a verbose mode to mrp2html.py for better diagnostics.  Improved error handling in mrp2html.py.  Added support for both .mrp.xml and \_mrp.xml file names, in order to support renaming of some download tools when downloading files with more than one dot in their file name.  Picked up DSP8008 v1.1.0, DSP8028 v1.1.0, DSP8029 v1.1.0, DSP8050 v1.1.0, DSP8051 v1.1.0, DSP8052 v1.1.0, DSP8053 v1.1.0, DSP8054 v1.1.0.  Changes in DSP8008 v1.1.0:  Upgraded to use MRP schema from dsp8028 v1.1.  Added <xhtml:p> elements around text where possible, to improve the quality of the line number generation.  Experimental: In test fragment "Logo", replaced reference to DMTF logo image file with inline definition of the logo, to avoid dependency on logo file.  Added schema location for MRP namespace.  Removed unused namespace declaration for XMLSchema.  Changes in DSP8028 v1.1.0:  Fixed issue with not finding ID values for IDREF values in some environments by changing the type of mrp:HeadingLink/@id from xsd:IDREF to xsd:NCName.  Removed the import of the xml namespace, because it was not used (it is still imported by dsp8053 which does use it).  Added support for profile type "Pattern" (in addition to Autonomous and Component). A pattern profile is not individually discoverable using PRP and thus does not define central class, scoping class and scoping path.  Added declaration of xml namespace and its import, in order to support XML editors that don't have that built-in.  Added support for include files included via XInclude. This caused the mrp:Profile child elements that can have many child elements, to become global elements (mrp:RelatedProfiles, mrp:Features, mrp:Elements, mrp:MessageRegistries, mrp:MetricRegistries, mrp:UseCases, mrp:Actors); these are the elements that can be in include files.  Added support for referencing class adaptation names in other profiles using either the adaptation names for DSP1001 1.1 profiles, or the class uses for DSP1001 1.0 profiles (that is, class name followed by some text in parenthesis, for example: "CIM\_Managedelement (affected element)"). This affects base adaptations, class adaptation links, and class adaptations in element links.  Added property implementation and method implementation as additional was to express conditions for requirement levels Conditional and ConditionalExclusive, via elements mrp:PropertyImplementationCondition and mrp:MethodImplementationCondition.  Changed names of generic operations specified in profiles, to the new names defined in DSP0223 1.0.2, which are now aligned with the DSP0200 operation names.  Changes in DSP8029 v1.1.0:  Upgraded to use MRP schema from dsp8028 v1.1.  Fixed the bug that adaptations with association operations but no other need for subclauses had generated dangling links for non-existing operation subclauses.  Added support for showing the mrp:Confidentiality element in the document footer.  Added a check that an XHTML "img" element always has an image file specified (via its "src" attribute).  Added tolerance for omitted mrp:Logo element (it is optional).  Added tolerance for omitted mrp:RelatedProfiles element (it is optional)  Added support for table links (mrp:TableLink) and figure links (mrp:FigureLink).  Added "div" element with id="div-map" in support of heading, figure and table links with dynamically determined link text.  Added "div" element with id="div-ln" in support of line numbering. Line numbering can be controlled with a new stylesheet parameter "linenumbers", which is turned off by default.  Added profile title before Scope clause.  Improved wording of text in subclause 7.2.1 "Conventions".  Removed generated feature usage list at the end of each feature subclause, because it could have been incomplete.  Editorial changes in name of use case and state description entries in use case table (added colon to make them consistent with the respective headings).  Editorial changes in introduction text before list of use case steps.  Editorial changes in punctuation of text in subclause 7.2.1 "Conventions".  Added page break before Bibliography.  Fixed issue that some table headings in related profiles table broke into a new line without generating a hyphen. The solution is only a workaround for this particular case, and soft hyphening has been removed from other table headings that do not typically break into new lines.  Fixed the issue that the vertical spacing before operation subclauses was larger than for property and method subclauses.  Improved a profile error message.  Fixed XSLT validation errors about missing parameter "h2-number" when calling template "output-heading-H2".  Added a "verbose" parameter that controls verbose messages in DSP8029, for better diagnostics.  Added support for property and method implementation conditions (for conditional requirement levels).  Changed names of generic operations specified in profiles, to the new names defined in DSP0223 1.0.2, which are now aligned with the DSP0200 operation names.  Changed generated text "Abstract Indicator: True/False" to "Abstract: Yes/No".  Changes in DSP8050 v1.1.0:  Picked up DSP1053 1.1.  Changes in DSP8051 v1.1.0:  Picked up DSP1053 1.1.  Changes in DSP8052 v1.1.0:  Picked up DSP1053 1.1.  Changes in DSP8053 v1.1.0:  Improved the format pattern for identifiers to be closer to what DSP0004 defines.  Added a type in support of referencing class uses in DSp1001 1.0 profiles.  Changes in DSP8054 v1.1.0:  Added styles FigureRef and TableRef for links to figures and tables.  Added support for long footer lines (e.g., WiP & Confidential) by defining minimum width for version field in footer to avoid that it breaks into two lines.  Added style div.lineNumber in support of line numbering.  Added support for header/footer to be flipped on even and odd pages.  Added support for showing page numbers in Table of Content/Figures/Tables.  Changes to match DSP1000 template:   * + - Left-adjusted text in table header rows for Prince-XML.     - Added style Heading1-Title for use by profile title.     - Center-adjusted caption text of figures and tables. |