



**PathMATE Transformation Maps
for C++ & Build
Version 8.02.011**

Release Notes

August 12, 2011

www.PathfinderMDA.com
888-662-7284

Table Of Contents

1. Introduction	1
Online Documentation	1
Demo	1
2. Installation	1
3. Technical Support	1
4. Compatibility	1
5. Impact to Existing Projects	2
Version 8.02.011.....	2
Version 8.02.010.....	2
6. New Features	3
Version 8.02.011.....	3
Version 8.02.010.....	4
Version 8.02.008.....	6
Version 8.01.000.....	8
7. Defects Repaired	12
Version 8.02.011.....	12
Version 8.02.010.....	12
Version 8.02.008.....	12
Version 8.01.000.....	13
8. Known Limitations	15
Version 8.02.011.....	15

1. Introduction

Online Documentation

PathMATE help is available from the Eclipse Help system - see the PathMATE topics under the Eclipse environment Help->Help Contents.

Additional Online product documentation and technical information may be found in the doc subdirectory of the base PathMATE installation (default c:\pathmate\doc). The documentation is stored in Adobe Acrobat (PDF) format. An Acrobat reader can be downloaded free of charge from www.adobe.com.

Demo

In addition to the product demo available from www.pathfindermda.com, be sure to see the PathMATE Quick Start Guide for RSM, and the PathMATE Quick Start Guide (for Rose users), located in the product install doc directory – typically c:/pathmate/doc.

2. Installation

Remove the existing C++, Common, and Build templates to ensure a clean base. Then install the PathMATE Transformation Map for C++ and Build in the same root directory as base PathMATE so the C++ and Build Templates transformation maps in the platform models delivered with PathMATE can find their template files.

3. Technical Support

Pathfinder Solutions Technical Support
Telephone: 888-662-7284 x103
Email: support@pathfindermda.com

4. Compatibility

Requires PathMATE 8.01.001 or higher

5. Impact to Existing Projects

Version 8.02.011

As a result of BUG02225:

After updating to this version of PathMATE any realized code that utilizes PfdString and the length service will have to be updated. Previously length returned current contents buffer size() which was typically string length + 1. Now length is equal to current string length and size maps to the buffer size. Realized code needs to be updated to account for this value change.

Version 8.02.010

As a result of REQ02237:

After updating to this version of PathMATE all multi-process properties files that define load modules via ProcessID markings will need to be updated to use the more accurate and current marking ProcessType.

Additionally De2faultProcessID was also updated to DefaultProcessType.

6. New Features

Version 8.02.011

Design – C++

REQ02251 – Auto Configure a Minimal Process Config Table

To facilitate deployments to targets without files systems or without topology configuration files, the PathMATE mechanisms have been updated to allow deployments the ability to discover build their system topology table at run.

To do this, start the process specifying a pid and port via the `-pid` and `-port` command line arguments, and DO NOT specify a process config table. Other processes connecting to it will add to its internal process config table via the information provided in their connect messages.

A key limitation of this is the process cannot send messages to other processes until those processes send a message to this process first.

REQ02253 – Complete the feature to specify listener port with the “-port” argument

Specify the correct IP address when using the `-port` command line argument

REQ02254 – Create an entry in the process config table from a connect message

When a new process connects to the local process that was previously unknown to the local process create an entry in the process config table from the connect message.

REQ02296 – Hand Off Parameters Pattern - Support and Stereotype

The Handoff Pattern a.k.a. the Send and Delete pattern will deallocate serializable items after sending has been completed. This works for both inter-process and inter-task messages and requires no changes to existing `PfdSerializable` subclasses. To mark a service apply the “HandOff” stereotype in the PathMATE Extensions profile.

The Hand Off Pattern is applicable when:

- Invoking a service through the non-local dispatcher,
- Invoking a service at multiple destinations via a routed non-local dispatcher to a group of destination handles.

The Hand Off Pattern is not applicable when:

- Invoking callbacks to "Hand Off" services
- When invoking the service directly through the standard function call

Please note that with the change:

- PfdSerializable now inherits from RCOBJECT
- PfdRCSerializable class has been removed.

Version 8.02.010

Design

REQ02219 – Added DefaultProcessID System Marking

The DefaultProcessID marking can now be applied to the system, this changes the default process name from MAIN to the specified name.

REQ02220 – Add RequireDomainAssignment System Marking

When the RequireDomainAssignment System marking is set to TRUE transformation errors are generated for each Domain not specifically assigned to a process type or ANY. Default value == FALSE.

REQ02229 – Static Model Element Numbering

The elements that can be numbered include:

- Domains
- Classes
- Domain and Class Services
- Serialized Types

Example markings:

- System,HugeExternalRealizedMsg,ProcessTypeIndex/MAIN,33344
- System,HugeExternalRealizedMsg,TaskIndex/SYS_TASK_ID_MAIN,999
- Domain,HugeExternalRealizedMsg.ExternalMessages,DomainIndex,300
- Object,HugeExternalRealizedMsg.ProcessImageDirector.TestTracker,ObjectIndex,1
- DomainService,HugeExternalRealizedMsg.RemoteShutdown.GroupTest,ServiceIndex,1
- GroupType,Group<Integer>,SerializeIndex,1;2;3

Additionally, a template properties file can be generated to show the default numbering scheme to aid in the transition to a static numbering scheme. The file is

<systemname>_indices_properties.txt_gen

REQ02230 – Static Process Type and Task Index Numbering

The new C++ supports that static numbering of Process Types and Task Indices, in the associated generated enumeration. The marking to number elements is:

- System,<systemname>,ProcessTypeIndex/<processname>,#index_value

- System,<systemname>,TaskIndex/<taskidname>,#index_value

Additionally, a template properties file can be generated to show the default numbering scheme to aid in the transition to a static numbering scheme.

The generated file name is as follows:

```
<systemname>_indices_properties.txt_gen
```

The contents of the file contain property markings for all enum literate values in use in the system generated files. The user can rename it and include in the main properties.txt file to fix the values in subsequent transformations.

REQ02237 – Rename ProcessID marking to ProcessType

Given the SPMD enhancements to the domain marking ProcessID was update to the more accurate name ProcessType. Additionally DefaultProcessID was also updated to DefaultProcessType.

Design – Build

REQ02216 – Expanded System Build Markings to be applied to Domains

The following markings previously only available on the System, are now available on both the System and Domains:

- AdditionalLibraries
- AdditionalIncludes
- Defines

REQ02236 – Lean Build

The lean build feature, which reduces the code compiled into each process type load module is enabled via setting the system marking EnableLeanBuild (Default ="FALSE") to TRUE. When enabled, realized code for domains which are not specified to live on a load module are not included into the build files.

Additionally the marking RealizedTypesPath was added to system and domain elements, to allow types code, used throughout the system to be included regardless of EnableLeanBuild being set to TRUE.

The new C++ supports that static numbering of Model elements in the applicable generated enumerations through markings.

Design - C++

REQ02196 – Binary Instance Data Loading

The C++ map now supports the loading of binary instance data streams and routing to appropriate domain and class methods.

Additional features including loading from a file (REQ02222) and sending the instance data in a message (REQ02223).

REQ02217 – Add XMLServiceHandleSupport marking

The XMLServiceHandleSupport applied to the system has a default value of ENABLED. If set to DISABLED the code to support incident handle loading from an XML file is no longer generated.

REQ02222 – Binary Instance Data Loading – From a File

Binary Instance loading was extended to support the loading of data from a file. See associated Technote for more information.

REQ02223 – Binary Instance Data Loading - Via Inter-process Message

Binary Instance loading was extended to support the loading of data on a remote process in blocks via inter-process messages. See associated Technote for more information.

REQ02225 – Add Serialization Support for Long

Serialization support was added via the new put_long_in_buffer and get_long_from_buffer services, following the standard PathMATE serialization pattern.

Version 8.02.008

Design

REQ02185 – Do not Trim if Element is Externally Loaded

The PathMATE Self-Trimming feature has been updated to exclude trimming of model elements which have been marked to be loaded via C Static Instance Initialization or XML Instance Loading.

This can occur when self-trimming has been enabled for models in which place holders for new functionality have been added.

Design - Build

REQ02190 – Add NIOS IDE Support

PathMATE Build Map now supports the generation of uOS (NIOS) IDE project files. This also includes multi-process support.

REQ02199 – Add RTP support to VxWorks Workbench Projects

PathMATE Build Map now supports the generation of RTP project files. for VxWorks Workbench 6.5.

REQ02207 – Preprocessor Definition Marking on Domains

As of PathMATE 8.2 C++ preprocessor definitions can be applied to individual domains and not just the system, simplifying property management across multiple deployments.

The marking is Defines and is a semi-colon separated list, as with the System marking. For example:

```
Domain,*.MyDomain,Defines,NO_PATH_IE;PATH_STUB_MY_DOMAIN
```

Design - C++

REQ02115 – Remove legacy filesystem code conditionally compiles with PATH_NO_FILESYSTEM

In mechanisms file "pfd_os.cpp", the section of code for accessing a file system that only compiles to Win32 has been reworked to not solely utilize the PATH_NO_FILESYSTEM #define to conditionally compile the code as well as support other operating systems. Applications under Linux OS are now able to take advantage of this feature.

REQ02182 – SuppressGeneration should not require SelfTrim

As of PathMATE 8.2, the SuppressGeneration feature will no longer require that the SelfTrim feature is enabled. Requiring SelfTrim has undesirable consequences, especially in sparse models or models with place-holders.

REQ02191 – Add Single Process Multi-Deployment (SPMD) Support

As of version 8.2 of PathMATE the C++ Map will support flexible start up sequences using the Single Process Multi-Deployment (SPMD) approach. For more information see the Multi Process Deployment tech-note.

REQ02197 – Support realized external messages

PathMATE's multi-process support needs to be enhanced to support the sending and receiving of realized inter-process messages, utilizing the existing inter-process communication mechanisms.

REQ02201 – Allow Allocation of Model Elements to Multiple Process Types

The PathMATE Multi-Process feature now supports the deployment of Domains and Domain Services to multi build modules without requiring their assignment to "ANY" process.

The ProcessID marking now accepts a semi-colon separated list of process ids.

REQ02203 – Support for Group<DestinationHandles> as Routing Parameter

The PathMATE Multi-Process feature now supports groups of destination handles as routing parameters. The non-local dispatcher shall dispatch a call to the specified domain service on each of the specified target nodes.

REQ02204 – Support large socket messages

PathMATE Messaging mechanisms were expanded to support arbitrarily sized socket messages. To support this feature PdfSerializable was extended to require a getCurrentSize() method, used by PdfProcess when constructing a PdfSocket message for outbound traffic

REQ02205 – Add UDP port to process config table

The PathMATE Multi-Process feature now allows UDP ports to be optionally specified in the process config table. For more information see the Multi Process Deployment tech-note.

REQ02206 – Allow EXTERNAL process in the Process Config Table

The PathMATE Multi-Process feature now allows "non-PathMATE" process to be identified in the process config table. For more information see the Multi Process Deployment tech-note.

REQ02209 – PathMATE "Internal" XML Parser

PathMATE has added a new simple XML Parser which does not require the use of the Xerces toolkit including linking in the library and running with the dll. The parser has been designed to be platform independent, not requiring a special build for various platforms. Common and C++ templates and mechanisms have been added to support this new feature, including InternalInstanceLoader.cpp and.hpp.

However this parser provided limited error information and no pre-loading validation against the generated schema. Additionally this is now the default XML Parser while the Xerces option is still available. For more information please see the XML Instance Loading tech-note included in the DOCs folder of your PathMATE installation.

REQ02226 – Model Level Unit Testing

The C++ Map has been expanded with a set of test templates to allow unit test to be defined in the model. Features include assert style verification in addition to delayed validation for state machine verification.

This feature requires the PathMATE Testing profile is applied to the model.

REQ02227 – UnitTest++ Integration

Support for the unit test framework UnitTest++ was added. The use of this framework is specified by the addition of the following marking:

System,*,TestFramework,UnitTest++

Version 8.01.000

Design

REQ01874 - Service handles to synchronous services (with return values or output parameters) are permitted to be created, but are never invoked.

A new transformation time check issues an error message if a service handle create statement in action language references a synchronous service.

REQ02120 - Allow values of enumerated data types to be added to the model dynamically

A new marking called EnumValuesFile can be set on a user defined enumerated type. The property specifies the path to a file containing a whitespace separated list of enum literals. The templates replace

the literal values specified in the model with the values specified in the file.

A new engine extension called `addEnumerationLiteral` was added. The parameters to this engine extension are

- the qualified name of the enumerated data type
- a whitespace separated list of literals to add
- a flag taking TRUE or FALSE as a string, where TRUE replaces the literals from the model, and FALSE adds them to the set from the model

REQ02185 – Do not Trim if Element is Externally Loaded

The PathMATE Self-Trimming feature has been updated to exclude trimming of model elements which have been marked to be loaded via C Static Instance Initialization or XML Instance Loading.

This can occur when self-trimming has been enabled for models in which place holders for new functionality have been added, resulting in proper validation of correct instance populations.

Design - Build

REQ02160 – Generate VS Projects to utilize Multi-core support

PathMATE Build Map now generates models into multiple Visual Studio projects (one per domain and one for the system) to utilize the Visual Studio ability to build each project on a different core, speeding up build times on multi-core machines

Design - C++

REQ02121 - Support instance reference parameters in service handles in XML file

Instance based services can now be loaded into an incident handle attribute using an XML file. The value of the XML `_x0040_id` field of the instance can be used as a value for the `this` parameter in the incident handle.

In the example below, the `op` attribute of the `Scripting_Action` class is an incident handle. The `op` incident handle is set to the instance based print service of the `System` class in the `Scripting` domain. The `this` parameter of the incident handle determines which instance will be used when the incident is called. In this case the `this` parameter is set to 1 which refers to the `System` instance declared in the file with the `_x0040_id` value of 1.

```
<Scripting_System>
<name>TestSystem</name>
<_x0040_id>1</_x0040_id>
</Scripting_System>
<Scripting_Action>
<op>Scripting.System.print(this=1)</op>
```

```
<_x0040_id>141</_x0040_id>
</Scripting_Action>
```

REQ02123 - Support a hash map implementation of groups

A hash map implementation of groups is now supported. The integer index is the key in the hash map. Set the TypeImplementation marking on the attribute or PAL statement that creates the group. PAL operation group[index] is used to index the hash for read and write.

For example:

```
Group<Real> valGrp; { TypeImplementation="Hash" }
valGrp[0] = 12345.67890;
val = valGrp[0];
```

REQ02124 - Allow PfdTask to interface with external event queue

REQ02125 - Provide external interface to shutdown model execution

Provide a method on the generated System class to shutdown the application. This clears the event queue, calls domain services marked with the <<Shutdown>> stereotype, and cleans up instances. This does not call SW::Shutdown() or cause the program to exit.

REQ02126 – Allow implementation for derived services to be specified in the markings

The property InlineCode, attached to a DomainService or ObjectService model element, can contain an implementation macro. The parameters to the service are referenced using the order of arguments. For example, \$1\$ is the first parameter.

For example, the UT:Assert service is tagged as Derived in the model.

In PAL, it looks like:

```
UT:Assert(val, 1);
```

The properties.txt file include the line:

```
DomainService,*.UT.Assert,InlineCode,if ($1$ != $2$) { cout <<
"ERROR" << endl; }
```

So the generated code for the PAL above would be:

```
if (val != 1) { cout << "ERROR" << endl; }
```

Support for commas in the value field for a property in the properties.txt file has been added. This is required to support the InlineCode feature property, since some inline code will require commas. See the common template gen_read_properties.arc.

REQ02161 – Invalid Guard Condition should throw Transformation Error

Now invalid guard conditions result in a transformation error, as opposed to a compilation time error.

7. Defects Repaired

Version 8.02.011

Design – C++

- BUG02225** PdfString length_ not correctly set on overwrite
- BUG02246** Non-Main Task Synchronous problem
- BUG02247** Non-Main Task Single Process problem
- BUG02248** Potential Memory Leak - Deliver on a callback using advanced realized types
- BUG02249** ARTs Not Deallocated
- BUG02250** Internal instance loader caused an Assertion Failure
- BUG02255** pdfString length_ not correctly set on overwrite
- BUG02258** Preprocess Directives for getTopology in process.* not defined consistently.
- BUG02259** SPMD Issue (Process Type Markings Transformation Error)
- BUG02260** XML Truncates Reals
- BUG02263** With 8.02.010, cannot create XvWorks builds that compile and build.
- BUG02264** 8.02.010 does not compile for NIOS
- BUG02272** Instance loader does not close XML files
- BUG02284** Binary Instance Data Loading doesn't write Boolean attributes
- BUG02286** Can't mark domain service to run where domain isn't

Version 8.02.010

Design – C++

- BUG02163** There is a memory leak in C++ multi-process deployments using service handles
- BUG02210** Using Local Memory Pools causes Multi-process Deployment Crash
- BUG02218** Internal XML Loader Sporadic Issue
- BUG02228** State Based Test Overrun
- BUG02235** SPMD doesn't properly handle elements interacting of different multiple processes

Version 8.02.008

Design – C++

- BUG02165** cpp compilation error - Incorrect use of ctime_r
- BUG02166** cpp compilation error - pfd_wait_for_semaphore invalid Linux code
- BUG02167** cpp compilation error - In SAX2Instance Loader stricmp not supported
- BUG02189** XML Instance Loader – Compilation error in Service Handle Loader for Domains without Interface
- BUG02192** SPMD local call ends up remote
- BUG02193** Incomplete Serializer Utility Functions
- BUG02194** Support File System on Linux by Default.
- BUG02195** Linux build errors caused by clock_gettime used in OS_POSIX
- BUG02198** Passing NULL as a serializable ART parameter causes crash
- BUG02200** PathMATE XML Loader and SPMD Does not support 64-Bit Linux
- BUG02202** SPMD remote call ends up as a local
- BUG02208** Internal XML Parser Does not Handle Comments Correctly

Version 8.01.000**Design**

- BUG02130** Commas not allowed in properties files as values.

Design – C++

- BUG01693** PATH_NO_DOUBLE defined even when operation has Real parameter
- BUG02090** An intertask incident may fail to wake up the receiving task (POSIX).
- BUG02092** Memory leak in tasking mechanism.
- BUG02094** Wrong definition of thread_mutexattr_t in critical.cpp mechanism file
- BUG02096** Bad return in the pfd_start_task_param routine (POSIX)
- BUG02100** Service handles are not allocated from any memory pool
- BUG02108** Use of ctime is not thread-safe
- BUG02109** Cannot compare two incident handles for equality.
- BUG02116** Need to qualify substate names in transition actions and handlers.

Design – Build

- BUG02101** Generated Visual Studio project explicitly link to libcmd.lib.
- BUG02113** DirCommand Extension fails on .h file lookups
- BUG02119** PATH_NO_STREAMS is defined by default for VxWorks builds, however it should be configurable.

8. Known Limitations

Version 8.02.011

- BUG02093** Generated and mechanisms code generate copious warning messages while compiling.
- BUG02144** Groups of Incident Handles are not supported as Primitive Types. Use Group<ServiceHandle> instead.
- BUG02145** Hash Table implementation of Groups not Supported in VS2005+.
- BUG02285** Complete error reporting for Binary Instance Data Loading