

All-In-One Installation & Test Tutorial for MDHT Release M7

Author: Les Westberg



This tutorial will walk through installation of the All-In-One MDHT installation and go through the exercise of creating a new project using an example project as the basis.

Download and Install the latest MDHT All-In-One Zip File

- 1) Before installing you will need to install the Java Development Kit. I did my installation with JDK 1.6 Update 23. Make sure that the JDK can be found on the path. A common way to do that is to create a system environment variable called JAVA_HOME and give it the path to the directory where you installed the JDK. Then alter the Path environment variable to include %JAVA_HOME%\bin on the path.
 - a) NOTE: We found a problem trying to use this with the 64 bit JDK we would get errors when we tried to start Eclipse. So we had to install the 32 bit JDK even though we had 64 bit windows.
- 2) Download the latest MDHT All-In-One Zip File:
<https://www.projects.openhealthtools.org/sf/projects/mdht/downloads/index.html>
- 3) Unzip this to a directory. (Use something like 7Zip. Do NOT use the Windows Compressed Folders Extraction Wizard - it has problems with some of the large file names.)
- 4) Run Eclipse.exe and verify the environment
 - a) You will find this in the following location:
`<unzip-directory>\MDHT_CDATools_M7\eclipse\eclipse.exe`
 - b) Note that when Eclipse starts up, it should build the entire project. The following problem may occur when it builds:
 - i) "Manifest has no main section". If this occurs, the problem needs to be resolved before proceeding. Do the following steps to resolve this:
 - (1) Highlight all of the projects in the Project Explorer window
 - (2) Right-click on the highlighted projects and select "Refresh" from the context menu.
 - (a) This will cause all projects to be refreshed and rebuilt. Once they are all rebuilt, this should resolve the build problems.
 - c) Make sure that java compatibility is set to 1.5
 - i) Select "Preferences" from the "Window" menu
 - ii) Expand the "Java" item
 - iii) Select the "Compiler" item.
 - iv) Change "Compiler compliance level" to 1.5
 - v) Select "OK" and let it rebuild everything again.
 - d) Verify that the build was successful
 - i) Expand the project: "org.openhealthtools.mdht.uml.cda"
 - ii) Expand the "src" folder
 - iii) Expand the "org.openhealthtools.mdht.uml.cda.tests" package

- iv) Right click on the "Main.java" file and select "Run As" and "Java Application" from the context menu.
- (1) This will use the class library to generate and validate an instance of a CDA document. In the console window. The output in the console window will show the example XML that was generated following by the validation results. The results should state "Document is Valid".

Create an Example Project

- 1) Pull the example project into this workspace from the MDHT subversion repository. (Note: when prompted for your subversion user name and password - this will be the same as your OHT user name and password.)
 - a) Select the subversion perspective. This is done by selecting the following Subversion icon in the upper-right side of the Eclipse window. 
 - b) Expand "<https://mdht.projects.openhealthtools.org/svn/mdht>", then "trunk", then "cda", then "examples"
 - c) Right-click "org.openhealthtools.mdht.uml.cda.example" and select "Check Out"
 - d) Select the UML Modeling perspective by selecting the icon in the upper-right side of the Eclipse window 
 - i) You should now see "org.openhealthtools.mdht.uml.cda.example" as one of the projects in the workspace.
- 2) Detach the Example project from Subversion
 - a) Right-Click on the project: "org.openhealthtools.mdht.uml.cda.example" and select "Team" followed by "Disconnect" from the context menus.
 - b) Select "Also delete the SVN meta-information from the file system." option
 - c) Select "Yes"

Rename Project

- 1) Right-click on the project and select "Refactor" followed by "Rename" from the context menus.
- 2) Enter your new project name and select "OK".

Edit refactor.xml and transform.xml Ant Scripts

- 1) Expand your new project and open the "refactor.xml " script in the editor.
- 2) Update the following attributes to the specified value

Attribute	Value
basePackage	Set this to the value of your project without the final portion. So if your project name is: org.openhealthtools.mdht.uml.cda.testmodel then the

	value would be "org.openhealthtools.mdht.uml.cda".
nsURI	Set this to your new URI. If your project name is: org.openhealthtools.mdht.uml.cda.testmodel then the value would be "http://www.openhealthtools.org/mdht/uml/cda/testmodel" .
packageName	Set this to the name of the package. If your project name is: org.openhealthtools.mdht.uml.cda.testmodel then the value would be "testmodel".
prefix	Set this to the name of the package. But this must be unique from the packageName. It is case sensitive... If your project name is: org.openhealthtools.mdht.uml.cda.testmodel then the value would be "TestModel".

- 3) Save the file.
- 4) Open the "transform.xml" script in the editor.
- 5) Update the following attributes

Attribute	Value
modelName	Set this to the same value as the "packageName" in the refactor.xml script. If your project name is: org.openhealthtools.mdht.uml.cda.testmodel then the value would be "testmodel".

- 6) Save the file.

Run the refactor.xml Ant Build Script

- 1) Right-click on the "refactor.xml" file in the project window and select "Run As" followed by "Ant build..." from the context menus.
- 2) Select the "JRE" tab.
- 3) Select "Run in the same JRE as the workspace".
- 4) Select "Run"
- 5) You should see the following output in the console:

```
Buildfile:
C:\projects\mdht3\MDHT_CDATools_M6\workspace\org.openhealthtools.m
dht.uml.cda.siframework\refactor.xml

refactor:
    [move] Moving 2 files to
C:\projects\mdht3\MDHT_CDATools_M6\workspace\org.openhealthtools.m
dht.uml.cda.siframework
    [move] Moving 4 files to
C:\projects\mdht3\MDHT_CDATools_M6\workspace\org.openhealthtools.m
dht.uml.cda.siframework
```

```
[move] Moving 3 files to
C:\projects\mdht3\MDHT_CDATools_M6\workspace\org.openhealthtools.m
dht.uml.cda.siframework\src\org\openhealthtools\mdht\uml\cda
[move] Moving 2 files to
C:\projects\mdht3\MDHT_CDATools_M6\workspace\org.openhealthtools.m
dht.uml.cda.siframework\src\org\openhealthtools\mdht\uml\cda\examp
le
[move] Moving 3 files to
C:\projects\mdht3\MDHT_CDATools_M6\workspace\org.openhealthtools.m
dht.uml.cda.siframework\src\org\openhealthtools\mdht\uml\cda\examp
le

BUILD SUCCESSFUL
Total time: 1 second
```

- a) Note that even though it successfully ran the refactor.xml script, you will notice that the project shows build errors. You have to regenerate the library artifacts to resolve this issue. Follow the steps in the next section.

Create the Ecore Artifact from the UML File

- 1) In the project, right-click on the "transform.xml" file and select "Run As" followed by "Ant Build..." from the context menus.
- 2) Select the "JRE" tab.
- 3) Select "Run in the same JRE as the workspace".
- 4) Select "Run".
 - a) Note: If you do not set to have this run in the same JRE as the workspace, you will get an error that implies that it cannot find the model based on the modelName that was set. If this occurs, return to step 1 in this section and make sure that step 2 and 3 is done.
 - b) When successful you will see the following output:

```
Buildfile:
C:\projects\mdht3\MDHT_CDATools_M6\workspace\org.openhealthtools.
mdht.uml.cda.siframework\transform.xml

all:

transformModel:
  [cdatools] Loaded model: siframework
[transformToEcoreModel] Saving model:
file:/C:/projects/mdht3/MDHT_CDATools_M6/workspace/org.openhealtht
```

ools.mdht.uml.cda.siframework/model/siframework_Ecore.uml

refresh:

BUILD SUCCESSFUL

Total time: 2 seconds

Generate Model Classes from the Ecore File

- 1) Right-click on the <ModelName>.genmodel file in the model folder and select "Reload" from the context menu.
- 2) Select "UML model" and then select "Next".
- 3) Select "next" again. (This tells it to pull the information from the <ModelName>_Ecore.uml file that was just created from the transform.
- 4) Select "Finish"
 - a) When this is finished, it will open the <ModelName>.genModel in the eclipse editor. It will have created new files for the <ModelName>.ecore and <ModelName>.genmodel files.
- 5) In the eclipse editor, right-click on the <ModelName> node (in the editor window and not the project window) and select "Generate Model Code" from the context menu.
 - a) Note: If a compiler occurs do the following steps to fix them.
 - i) Expand the "src" folder in the project.
 - ii) Expand the <package>.util package.
 - iii) Delete the <ModelName>AdapterFactory.java and <ModelName>Switch.java files.
 - iv) Generate the model code again by Right-clicking on the <ModelName> node in the Eclipse editor (The editor window and not the project window) and selecting "Generate Model Code" from the context menu. This will regenerate those two files and rebuild the project. The build errors should now be gone.

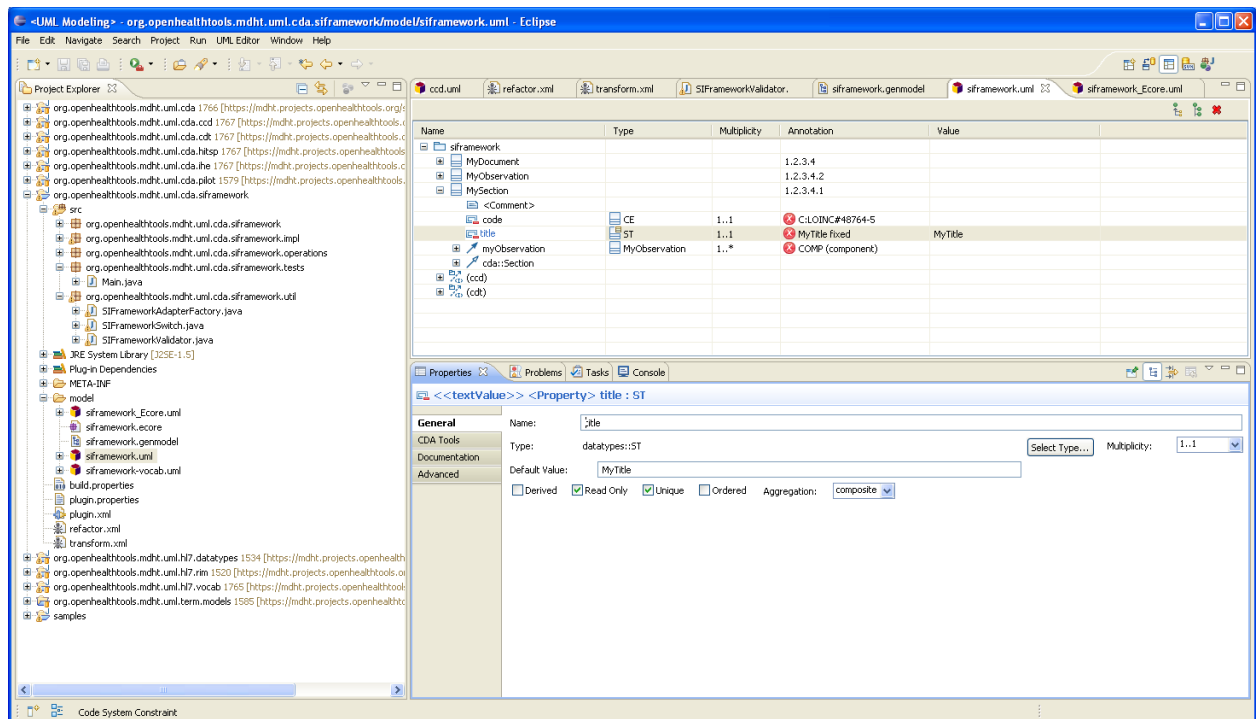
Verify that the New Project is Functional

- 1) In the projects window, expand the new project.
- 2) Expand "src" followed by <project>.tests package
- 3) Right-click on "Main.java" and select "Run As" followed by "Java Application". This will create an example XML and validate the example. Note that the example that is generated

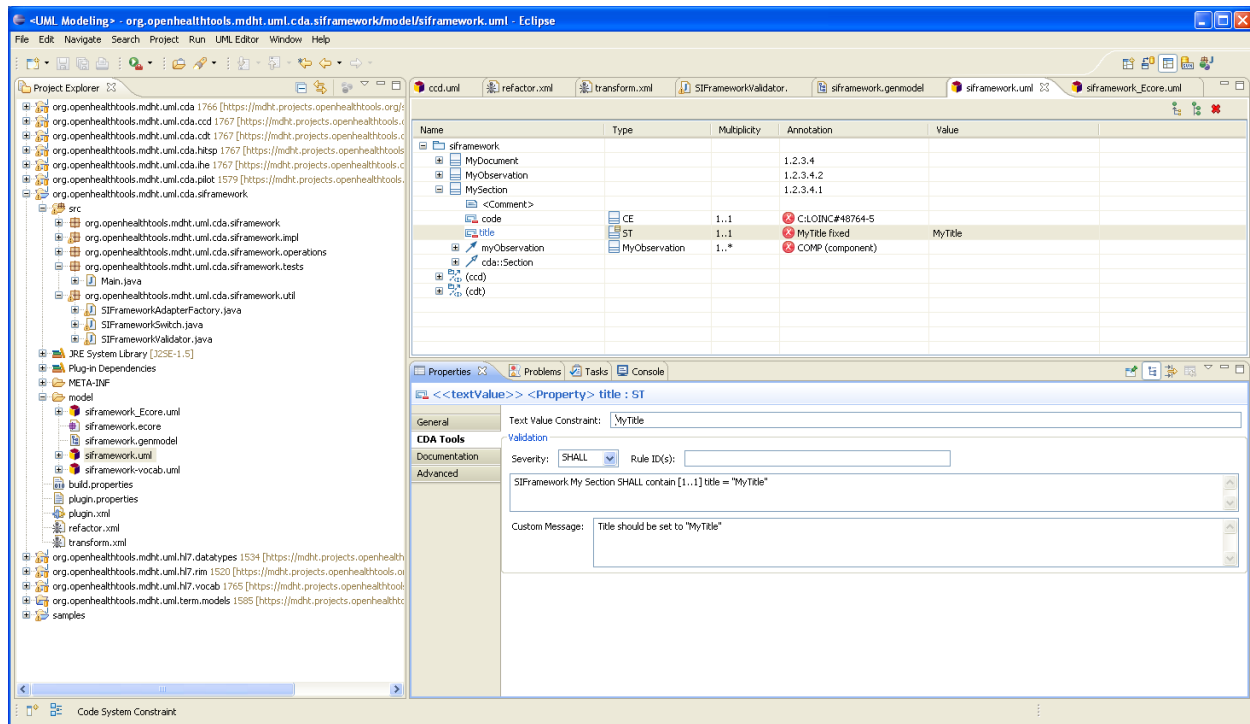
will say "Document is invalid" and show the errors and warnings about the instance of the document. This will show that the class library is working correctly. If you inspect the rest of the lines in the console window you will see which fields are invalid according to the constraints and the reason they are invalid.

Change the UML File

- 1) Open and Change the uml file.
 - a) In the projects window, expand your project, then "model".
 - b) Open <ModelName>.uml in the Eclipse editor. (Note this is NOT the <ModelName>_Ecore.uml file. It is the other one.)
- 2) Expand <ModelName>, followed by "MySection" in the eclipse editor.
- 3) Select "title" and make sure that the "Properties" tab is showing.
- 4) Change the "Default Value" to "MyTitle".
- 5) Check "Read Only".
- 6) Your screen should look like this:



- 7) Select the "CDA tools" tab.
- 8) Enter the "MyTitle" into the Text Value Constraint field.
- 9) Enter "Title should be set to MyTitle" in the "Custom Message" field
- 10) Save the file.
- 11) Your screen should look like this:



Create the Ecore Artifact from the UML File

- 1) In the project, right-click on the "transform.xml" file and select "Run As" followed by "Ant Build..." from the context menus.
- 2) Select the "JRE" tab.
- 3) Select "Run in the same JRE as the workspace".
- 4) Select "Run".
 - a) Note: If you do not set to have this run in the same JRE as the workspace, you will get an error that implies that it cannot find the model based on the modelName that was set. If this occurs, return to step 1 in this section and make sure that step 2 and 3 is done.
 - b) When successful you will see the following output (with your model name):

```
Buildfile:
C:\projects\mdhtr7\MDHT_CDATools_M7\workspace\org.openhealthtools.mdht.
uml.cda.siframework\transform.xml

all:

transformModel:
    [cdatools] Loaded model: siframework
    [transformToEcoreModel] Saving model:
file:/C:/projects/mdhtr7/MDHT_CDATools_M7/workspace/org.openhealthtools
.mdht.uml.cda.siframework/model/siframework_Ecore.uml

refresh:
BUILD SUCCESSFUL
Total time: 1 second
```

- c) Open the <ModelName>_Ecore.uml file in the Eclipse editor. This file is located in the model folder of the project.
- d) Expand "MySection".
- e) Notice that "MySectionTitle" constraint now shows the change. The value should look as follows:

```
not self.title.oclIsUndefined() and self.title.getText() = 'MyTitle'
```

Generate Model Classes from the Ecore File

- 6) Right-click on the <ModelName>.genmodel file in the model folder and select "Reload" from the context menu.
- 7) Select "UML model" and then select "Next".
- 8) Select "next" again. (This tells it to pull the information from the <ModelName>_Ecore.uml file that was just created from the transform.
- 9) Select "Finish"
 - a) When this is finished, it will open the <ModelName>.genModel in the eclipse editor. It will have created new files for the <ModelName>.ecore and <ModelName>.genmodel files.
- 10) In the eclipse editor, right-click on the <ModelName> node (in the editor window and NOT the project window) and select "Generate Model Code" from the context menu.
 - a) Note: This will generate the code and compile it. If you get compiler errors when it builds, do the following steps to fix them.
 - i) Expand the "src" folder in the project.
 - ii) Expand the <package>.util package.
 - iii) Delete the <ModelName>AdapterFactory.java and <ModelName>Switch.java files.
 - iv) Generate the model code again by Right-clicking on the "SIFramework" node in the Eclipse editor and selecting "Generate Model Code" from the context menu. This will regenerate those two files and rebuild the project. The build errors should now be gone.

Run Test to Verify the Generated Code

- 1) Expand the "src" folder in the project.
- 2) Expand <package>.tests package.
- 3) Right-click on "Main.java" and select "Run As" followed by "Java Application" from the context menu.
 - a) This will re-run the test and validate the generated XML. You can validate that the XML now contains the title with a default value of "MyTitle" in the console window. The following shows an example of the XML that was generated... Note that the section now contains the "<title>" tag. Also note that the last line will state that the "Document is invalid". This is OK. It is supposed to state that.

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



```
xmlns="urn:hl7-org:v3" xsi:schemaLocation="urn:hl7-org:v3 CDA.xsd">
  <realmCode code="US"/>
  <templateId root="2.16.840.1.113883.10.20.3"/>
  <templateId root="1.2.3.4"/>
  <component>
    <structuredBody>
      <component>
        <section>
          <templateId root="1.2.3.4.1"/>
          <code code="48764-5" codeSystem="2.16.840.1.113883.6.1"
codeSystemName="LOINC" displayName="Summary Purpose"/>
          <title>MyTitle</title>
          <entry>
            <observation classCode="OBS" moodCode="EVN">
              <templateId root="2.16.840.1.113883.10.20.1.28"/>
              <templateId root="1.2.3.4.2"/>
              <code codeSystem="2.16.840.1.113883.6.96"
codeSystemName="SNOMEDCT"/>
              <statusCode code="completed"/>
            </observation>
          </entry>
        </section>
      </component>
    </structuredBody>
  </component>
</ClinicalDocument>
```