Guide to the DLRL WhiteListed MessageBoard example

Version 2

Date: 10 November 2010

Table of Content

1 Introduction

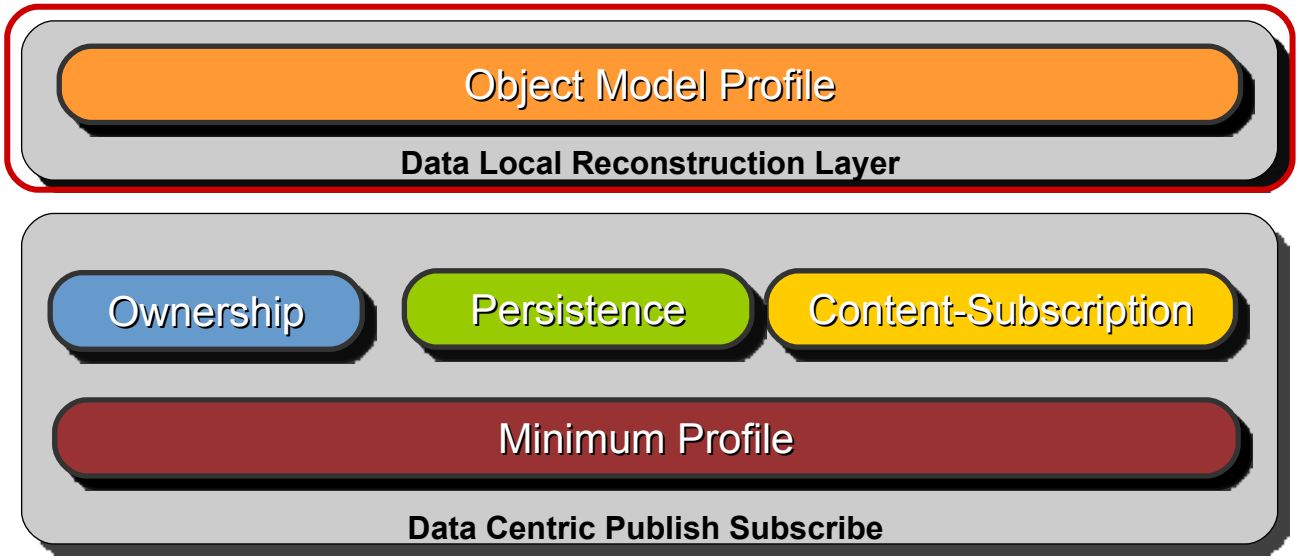
This document contains a small guide to the OpenSpliceDDS DLRL tutorial examples, explaining the purpose of the various applications within the example and providing some additional helpful information.

2 DLRL in a nutshell

The DDS specification consists of multiple profiles, one of those profiles is the Object Model profile, represented by the DLRL. OpenSpliceDDS also implements this profile and thus offers the benefits of this profile. The OpenSpliceDDS DLRL implementation allows the software developer to use the power of OpenSpliceDDS and the ease of Object Oriented development. OpenSpliceDDS offers a generic solution realizing the relational data to Object transformations. It includes powerful navigation between objects, which is a natural but important mechanism of Object Oriented applications and offers a simple and easy to use interface.

With the DLRL the application developer can describe classes of objects with their methods, data fields and relations, manipulate these objects (i.e., create, modify, delete) using the native language constructs that will, under the scene, activate the attached DCPS entities accordingly. These objects can be refreshed transparently without risking potential conflicts between incoming updates and application-made modification, as the DLRL offers means to manage such potential conflicts.

The DLRL relies on a mapping between the DLRL objects and DCPS entities, this mapping is generated from a set of tags that annotate the DLRL model. The user can choose to use an explicit mapping, or decide to use DLRLs default mapping rules.
The DLRL also uses typed interfaces like DCPS.



OMG DDS profiles

3 The WhiteListed MessageBoard example

The DLRL tutorial example delivered with OpenSpliceDDS continues with the DCPS tutorial example. For DCPS a simple example is shown where a chatter application and a messageboard application implement a simple chatroom. Within this example any user is allowed to send messages to the chatroom. This example concept will be re-used within the DLRL example, but with a twist. Instead of any user being allowed to send messages to the chatroom, only users that are explicitly mentioned in a whitelist will have their messages displayed, messages from other users will be blocked.

This example is contains 3 small applications:

- WhiteListEditor
  - To create a whitelist and add users to the whitelist.
- WhiteListViewer
  - To view which whitelists exists and which users belong to each whitelist
- WhiteListedMessageBoard
  - Displays only messages by users that are whitelisted.

The DLRL example will not implement the chatter application itself, instead it will re-use the existing chatter application of the DCPS tutorial, to illustrate how DCPS applications and DLRL applications can work together without any problems.

3.1 The WhiteListEditor application

The WhiteListEditor allows you to create named whitelists, for example one whitelist containing all your family, and another one containing all your friends. These whitelists are in turn read by the WhiteListedMessageBoard application to determine if a message should be displayed or not. The WhiteListEditor application demonstrates how objects can be written into the system using DLRL, and uses a small and simple Object model to accomplish this task.

3.2 The WhiteListViewer application

The WhiteListViewer is a simple application which displays all whitelists known within the system and shows which users are contained within each whitelist. The WhiteListViewer application demonstrates how objects can be read from the system using a basic pull mechanism where the application decides the moment when updates are applied. It also shows how very specific modification information regarding the last update round can be easily retrieved, so that an application only needs to view the data that actually changed.

3.3 The WhiteListedMessageBoard application

The WhiteListedMessageBoard is an application that only displays messages from users contained within the specified whitelist, any other messages are blocked. The WhiteListedMessageBoard application demonstrates how relations are used within DLRL, how event driven applications can be written using DLRL and how selections work within DLRL.

3.4 How to build the example

First make sure you have successfuly built and run the DCPS tutorial examples. (See the Getting Started Guide). Get a good feeling of what the DCPS Chatter application does. Then start building the DLRL tutorial.

On Unix/Linux platforms make sure the release.com script of OpenSplice has been sourced, and then just run the `BUILD` script in a separate shell:

```
sh BUILD
```

On Windows platforms just run the `BUILD.bat` script for JAVA or for C++ build using the Microsoft Visual Studio projects on the C/C++ examples solution file.

3.5 How to run the example

Three simple scripts are provided to easily run each application:

- run_editor
  - Option: –name <whiteListName>
  - Option: -add <userID1,userName1>
    - May be used multiple times
  - Example: run_editor -name family -add 1,mam 2,dad 3,John
    - Adds mam, dad and John to the WhiteList named family. The userID's must comply to the id's you use in the DCPS Chatter application.
- run_viewer
  - Takes no options
  - Example: run_viewer
    - Displays all known whitelists and the users added to each whitelist
- run_messageboard
  - Option: -name <whiteListName>
  - Example: run_messageboard -name family
    - Displays only messages coming from senders mentioned in the WhiteList named 'family'.

The Chatter application is re-used from the DCPS tutorial example, see the DCPS tutorial for information on how to run that application.

3.6 Typical scenario

The typical scenarion for this tutorial is as follows:

- Start the OpenSplice deamons on the node you are running the example
  - see the DCPS examples on how to accomplish this.
- Create 1 or more named WhiteLists with some users added to the whitelists using the WhiteListEditor application.
- Run the WhiteListViewer application to see which whitelists exist and which users are added to each whitelist.
- Run the WhiteListedMessageBoard application in its own shell/command prompt, using one of the available named WhiteLists.
- Start up a number of Chatters (each in their own separate shell/command prompt, as not to mix up their screen output).

Watch what happens to messages from users that are mentioned in the WhiteList and what happens to messages of users that are not. Also try to add active users to the WhiteList while the WhiteListedMessageBoard is already running. See if it picks up the changes on the fly.

To terminate the messageboard, just make one Chatter transmit a terminate message (userID = -1):
Chatter -1

3.7 Different programming languages not a problem for OpenSpliceDDS!

As you can see you can use the DCPS chatter program and the DLRL messageboard will receive it's messages. This shows that building new programs with DLRL while you still have legacy DCPS applications is not a problem, they communicate just fine. Another interesting scenario is to use different programming languages for each application. For example try running the DCPS chatter in the 'C' language, the WhiteListEditor in 'Java' and the WhitelistedMessageboard in 'C++'. You'll notice they will all work together just fine and that you do not have to do anything special at all. It doesn't get any easier.

3.8 Example directory and file overview

The following section gives a short overview of which directories and files can be found in the DLRL example.

3.8.1 Standalone Java language binding

OSPL_HOME/etc/examples/dlrl/standalone/Java/Tutorial

- BUILD (BUILD.bat on windows machines)
  Contains the build script to build the example
- OpenSpliceDDS-dlrl-example-guide.pdf
  Contains this guide
- run_editor (run_editor.bat on windows machines)
  Script to run the WhiteListEditor application
- run_messageboard (run_messageboard .bat on windows machines)
  Script to run the WhiteListedMessageBoard application
- run_viewer (run_viewer.bat on windows machines)
  Script to run the WhiteListViewer application
- Common
  - DlrlUtility.java
    Contains various useful utility functions used throughout the example applications
  - PreconfigureTopics.java
    Contains functions to help simulate external topic creation
- Data_files
  - ExtChat.idl
    Contains the DCPS topic defintions
  - WhiteListObjects.idl
    Contains the Object Model as used by the WhiteListedMessageBoard application
  - WhiteListObjects_editor.idl
    Contains the Object Model as used by the WhiteListEditor and WhiteListViewer applications
  - mapping.xml
    Contains the mapping of the DCPS topic definitions onto the Object Model as used by the WhiteListedMessageBoard application
  - mapping_editor.xml
    Contains the mapping of the DCPS topic definitions onto the Object Model as used by the WhiteListEditor and WhiteListViewer application
- WhiteListEditor
  - WhiteListCustomImpl.java
    Implements the custom operation defined on the WhiteList object as used within the WhiteListEditor and WhiteListViewer applications
  - WhiteListEditor.java
    Contains the implementation for the WhiteListEditor application
- WhiteListedMessageBoard
  - ChatMessageCustomImpl.java
    Implements the custom operation defined on the WhiteList object as used within the WhiteListedMessageBoard application
  - MyChatMessageListener.java
    Implements the ChatMessage listener which is triggered each time an update regarding a ChatMessage object is received.
  - MyWhiteListFilter.java
    Contains an application specific filter algorithm to select a WhiteList object
  - MyWhiteListSelectionListener.java
    Implements the WhiteListSelection listener which is triggered each time something changes within the selection.
  - WhiteListedMessageBoard.java
    Contains the implementation for the WhiteListedMessageBoard application
- WhiteListViewer
  - WhiteListViewer.java
    Contains the implementation for the WhiteListViewer application

3.8.2 Standalone C++ language binding

OSPL_HOME/etc/examples/dlrl/standalone/C++/Tutorial

- BUILD (BUILD.bat on windows machines)
  Contains the build script to build the example
- OpenSpliceDDS-dlrl-example-guide.pdf
  Contains this guide
- run_editor (run_editor.bat on windows machines)
  Script to run the WhiteListEditor application
- run_messageboard (run_messageboard .bat on windows machines)
  Script to run the WhiteListedMessageBoard application
- run_viewer (run_viewer.bat on windows machines)
  Script to run the WhiteListViewer application
- Common
  - DlrlUtility.h & DlrlUtility.cpp
    Contains various useful utility functions used throughout the example applications
  - PreconfigureTopics.h & PreconfigureTopics.cpp
    Contains functions to help simulate external topic creation
  - dlrl_tutorial_if.h
    This header file is windows specific only, and is used for dll import/export definitions.

- Data_files
  - ExtChat.idl
    Contains the DCPS topic defintions
  - WhiteListObjects.idl
    Contains the Object Model as used by the WhiteListedMessageBoard application
  - WhiteListObjects_editor.idl
    Contains the Object Model as used by the WhiteListEditor and WhiteListViewer applications
  - mapping.xml
    Contains the mapping of the DCPS topic definitions onto the Object Model as used by the WhiteListedMessageBoard application
  - mapping_editor.xml
    Contains the mapping of the DCPS topic definitions onto the Object Model as used by the WhiteListEditor and WhiteListViewer application
- WhiteListEditor
  - WhiteListObjects_editorCustomImpl.cpp & WhiteListObjects_editorCustomImpl.h
    Implements the custom operation defined on the WhiteList object as used within the WhiteListEditor and WhiteListViewer applications
  - WhiteListEditor.h & WhiteListEditor.cpp
    Contains the implementation for the WhiteListEditor application
- WhiteListedMessageBoard
  - WhiteListObjectsCustomImpl.h & WhiteListObjectsCustomImpl.cpp
    Implements the custom operation defined on the WhiteList object as used within the WhiteListedMessageBoard application
  - MyChatMessageListener.h & MyChatMessageListener.cpp
    Implements the ChatMessage listener which is triggered each time an update regarding a ChatMessage object is received.
  - MyWhiteListFilter.h & MyWhiteListFilter.cpp
    Contains an application specific filter algorithm to select a WhiteList object
  - MyWhiteListSelectionListener.h & MyWhiteListSelectionListener.cpp
    Implements the WhiteListSelection listener which is triggered each time something changes within the selection.
  - WhiteListedMessageBoard.h & WhiteListedMessageBoard.cpp
    Contains the implementation for the WhiteListedMessageBoard application
- WhiteListViewer
  - WhiteListViewer.h & WhiteListViewer.cpp
    Contains the implementation for the WhiteListViewer application