Adapter Remoting Infrastructure

General Overview

Adapter Remoting Infrastructure: 1.9.6
Last updated: 15/5/2019

Table of contents

1 The Architecture

The Lightstreamer **Adapter Remoting Infrastructure** (**ARI**) provides an easy way to develop Remote Adapters for Lightstreamer Server. “Remote” means that an Adapter does not run within the same process as the Java Virtual Machine running the Lightstreamer Server, but within a different process. Such process can be either local (on the same box) or actually remote (on a different box).

So, the advantages of a Remote Adapter with respect to a standard in-process Java Adapter are the following:

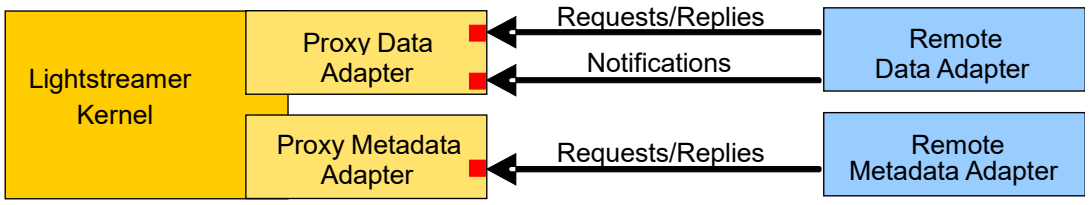
- Complete physical decoupling between the code of the Server and the code of the Adapter, avoiding possible interferences (memory usage, thread management, etc.).
- Possibility of deploying the Adapter on a different box and, if necessary, behind a second firewall. Thanks to the support for TLS communication and password-based authentication, the Adapter-server communication could even occur across the outer Internet.
- Possibility of implementing the Adapter in any language, rather than Java. The SDKs for .NET Adapter development and for Node.js adapter development, provided as part of Lightstreamer distribution, are actually based on the ARI.

But there is also a drawback. Every interaction between Lightstreamer Server and a Remote Adapter requires information to be exchanged over TCP packets, rather than communicating in-process, resulting in potentially poorer performance. For this reason, some highly intensive callbacks of the Metadata Adapter interface are not supported by the ARI (for example, isSelected()).

The communication between Lightstreamer Server and a Remote Adapter is based on unidirectional channels:

- **request channel** (data flow from Lightstreamer Server to the Remote Adapter)
- **reply channel** (data flow from the Remote Adapter to Lightstreamer Server)
- **notification channel** (data flow from the Remote Adapter to Lightstreamer Server)

Such channels are implemented over TCP sockets.
The figure below shows the general architecture of the ARI:



ARI Protocol

The Proxy Data Adapter and the Proxy Metadata Adapter are ready-made Adapters, written in Java and provided as part of the Lightstreamer Server. They expose the Data Provider and Metadata Provider interfaces over TCP/IP sockets.

The **Proxy Data Adapter** implements two server sockets. The first socket is used to transport the data flow of two unidirectional channels: **requests** (from the Proxy Data Adapter to the Remote Data Adapter) and **replies** (from the Remote Data Adapter to the Proxy Data Adapter). The second is used

to transport the data flow of one unidirectional channel: **notifications** (from the Remote Data Adapter to the Proxy Data Adapter).
There exist two flavors of Proxy Data Adapter: **NetworkedDataProvider** and **RobustNetworkedDataProvider** (see below).

The **Proxy Metadata Adapter** implements one server socket, used to transport the data flow of two unidirectional channels: **requests** (from the Proxy Metadata Adapter to the Remote Metadata Adapter) and **replies** (from the Remote Metadata Adapter to the Proxy Metadata Adapter).
There exist two flavors of Proxy Metadata Adapter: **NetworkedMetadataProvider** and **RobustNetworkedMetadataProvider** (see below).

There is always a one-to-one relationship between a Proxy Adapter instance and a Remote Adapter instance.

The protocol used for the communication between the Proxy Adapters and their Remote Adapters counterparts is fully documented in the Lightstreamer/DOCS-SDKs/sdk_adapter_generic/doc/ARI Protocol.pdf document.

NOTE: From a TCP/IP perspective, the Remote Adapters are always clients, that is, they open TCP connections to Lightstreamer Server.
Also with respect to encryption and authentication, when leveraged, the Remote Adapters act as clients: they receive and validate the TLS certificate from the Proxy Adapters and provide their credentials, to be checked by the Proxy Adapters.

When Lightstreamer Server is launched, with a configured Proxy Adapter, it listens on its server sockets(s) waiting for Remote Adapters to connect. After a Remote Adapter has connected and, if required, successfully authenticated, its connection will be kept open forever. Multiple concurrent connection attempts on the same server socket are supported, but, as soon as the first one succeeds, the other ones will be truncated.

NOTE: Lightstreamer Server performs Adapters initialization in parallel. Hence, if your custom executable manages several Remote Adapter Servers, the connections to the related Proxy Adapters can be performed in sequence and no specific order is required.
Just consider that, after the connection to a Proxy Adapter has succeeded, there is no formal guarantee that the connection to the next Proxy Adapter will succeed immediately; a short time gap may occur on Lightstreamer Server between the opening of the server sockets for different Proxy Adapters.

Closing the connection could provoke the automatic shutdown of Lightstreamer Server, depending on the Proxy Adapter involved. The **NetworkedDataProvider** and **NetworkedMetadataProvider** shut the Server down, if disconnected from the remote counterparts, in order to keep client and server state consistency (they cannot take any recovery action, so the best action is to have the client reconnect to another server instance in the cluster). On the other hand, the **RobustNetworkedDataProvider** and **RobustNetworkedMetadataProvider** contain some recovery capabilities and avoid to terminate the Lightstreamer Server process. Full details on the recovery behavior are available as inline comments within the example adapters.xml file available in the “Lightstreamer/DOCS-SDKs/adapter_remoting_infrastructure/doc/adapter_robust_conf_template” folder.

Connections unresponsive but not closed can be detected and treated as closed by the Proxy Adapter through a configurable timeout on the activity of the sockets. The mechanism is only meaningful if the Remote Adapter Servers are configured to send suitable keepalive packets at regular intervals.

2 Deployment of the Proxy Adapters

All the Proxy Adapters of the Adapter Remoting Infrastructure (i.e. Data and Metadata, in the normal and “Robust” version) are pre-deployed inside the Lightstreamer Server. The only necessary action to enable instances of the available adapters is to point them, using their symbolic names, in the adapters.xml file(s).

In order to deploy an Adapter Set which includes both a Proxy Metadata Adapter and a Proxy Data Adapter, the following steps should be followed:

1) Create a directory within “Lightstreamer/adapters”. It can be whatever name (for example, “proxy”).
2) Depending on the chosen Proxy Adapter type:
   - If deploying Proxy Adapters with “Robust” Proxy Data and Metadata Adapters, copy the “adapters.xml” file located under “Lightstreamer/DOCS-SDKs/adapter_remoting_infrastructure/doc/adapter_robust_conf_template” to the “proxy” directory.
   - If deploying normal Proxy Adapters, copy the “adapters.xml” file located under “Lightstreamer/DOCS-SDKs/adapter_remoting_infrastructure/doc/adapter_conf_template” to the “proxy” directory.
3) Edit the copied “adapters.xml” file to configure the Proxy Adapters. See the file for inline comments.

The provided “adapters.xml” files are templates showing a typical configuration, where both the Adapters (Data and Metadata) are either normal or “Robust” ones. Heterogeneous configurations are possible too. For example:

- A Proxy Data Adapter together with a provided Java Metadata Adapter (such as the LiteralBasedProvider).
- A Proxy Data Adapter together with a custom Java Metadata Adapter.
- A custom Java Data Adapter together with a Proxy Metadata Adapter.
- A Proxy Data Adapter based of “Robust” type together with a Proxy Metadata Adapter of normal type.
- Etc... Any other combination is possible.

If the defined Adapter Set includes multiple Data Adapters, each of them can be either a Proxy Data Adapter or a custom Java Data Adapter.