

Table of Contents

<i>Implementing Channel's Source and Sink</i>	2
--	---

[Programmers Reference, Adding Custom Channels, Source and sink](#)

Implementing Channel's Source and Sink

The first step that is required to add a custom channel to **AwareIM** is to write the component(s) that handle communication between **AwareIM** and the external software system or hardware device. In order to write the code for such components it is important to understand the architectural framework that **AwareIM** uses to communicate with an external software system or device ("external party").

AwareIM incorporates the **openadaptor** framework written by Dresdner Kleinwort Wasserstein to handle communication with external parties. This framework abstracts the process of sending messages between systems. The source of the message is abstracted in a component called *Source* and the destination of the message is abstracted in the component called *Sink*. Thus messages are generated by *Source* and travel to the corresponding *Sink*.

If communication with the external party requires that messages be sent from **AwareIM** to the external party the specific *Sink* needs to be implemented (**AwareIM** framework will automatically provide the corresponding *Source*). If communication requires that messages be sent from the external party to **AwareIM** the specific *Source* needs to be implemented (**AwareIM** framework will automatically provide the corresponding *Sink*).

Messages that float between **AwareIM** and external party represent arrays of **DataObject**'s. **DataObject** is the class that allows defining any number of attributes of different types and provides generic access to them - very much like the **IEntity** interface (in fact **IEntity** is a wrapper around **DataObject**). **DataObject** exposes the [getAttributeValue](#) and [setAttributeValue](#) methods that allow access to the values of its attributes.

AwareIM sends **DataObjects** of some specific structure to the external party and it expects **DataObjects** of this structure from the external party. As explained in the previous section this structure implies that there are only two forms of requests that can be sent between **AwareIM** and the external party – service requests and notifications.

AwareIM provides two utility classes [MessageBuilder](#) and [MessageInterpreter](#) that allow a programmer to easily create or read messages representing service requests and notifications without knowing too many details about the underlying structures of **DataObjects** and its attributes.

To summarize a programmer has to do the following:

1. When writing a *Sink* the service request or notification from **AwareIM** has to be translated into the protocol of the external party and sent to this party.
2. When writing a *Source* the message in the external party's format has to be translated into the standard service request or notification and sent to **AwareIM**.

The details of this process are described in the sections below.

- [Writing a Sink](#)
- [Writing a Source](#)
- [Handling replies to service requests](#)
- [MessageInterpreter class](#)
- [MessageBuilder class](#)

From:
<http://www.awareim.com/dokuwiki/> - **Documentation**



Permanent link:
<http://www.awareim.com/dokuwiki/docs/3500/0400/0410>

Last update: **2025/06/12 02:36**