

Table of Contents

Handling replies to service requests	2
--	---

Handling replies to service requests

When a service of the external party or **AwareIM** has been requested the service provider may send a reply. The reply is sent as a special notification (called the *service reply notification*). This notification implements the [INotification](#) interface and exposes methods that allow checking the status of the service request and retrieve error message if there was any.

If **AwareIM** requests services of external parties, replies from the external parties need to be delivered back to **AwareIM**. Rather than write the Source that would do this it is possible to use the special version of a Sink that would not only deliver service requests to the external party but also send the immediate replies back to **AwareIM**.

Rather than extending [AbstractSimpleSink](#) class such a Sink must extend the [AttachedReplySink](#) class. [AttachedReplySink](#) has the Source, which **AwareIM** automatically attaches to the Sink when Sources and Sinks are constructed. The only extra functionality that the Sink extending [AttachedReplySink](#) has to provide is to feed the reply to the attached Source when the reply becomes available – this will guarantee that the reply will be delivered to **AwareIM**. Feeding reply to the source is achieved by calling the [feedToSource](#) method of [AttachedReplySink](#) passing it the [DataObjects](#) as required by the Source.

The message fed to the source must contain the service reply notification. The service reply notification can be constructed calling the static method

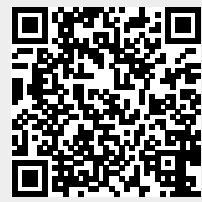
```
INotification DomainFactory.createServiceReplyNotification
(INotificationDefinition definition, int serviceStatus, String
errorMessage);
```

where `definition` must correspond to the structure of the reply declared for the requested service (or `null` if a “standard” reply has been declared), `serviceStatus` is the constant described in the [getServiceStatus](#) method of the [INotification](#) interface and `errorMessage` is the message describing the error if service failed.

Once the notification has been created it can be converted to [DataObjects](#) using the [MessageBuilder](#) class and fed to the source. Here is the snippet of the code illustrating this:

```
public void processMessage (Message msg)
{
    // call the service here
    // ...
    // construct service reply notification and feed it back to Aware IM
    INotification serviceReply =
        DomainFactory.createServiceReplyNotification (null,
        INotification.SERVICE_STATUS_SUCCESS, null);
    MessageBuilder mb = new MessageBuilder ();
    feedToSource (msg.peekDataObjects (), mb.addNotification (null,
    serviceReply));
}
```

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



Permanent link:
<http://www.awareim.com/dokuwiki/docs/3500/0400/0410/0413?rev=1749695763>

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