ASYNCHRONOUS MESSAGE CALL TO ANOTHER SERVICE

Making an asynchronous message call is exactly the same process as for a synchronous one, but with a `TypeConst.VOID` return type. The calling mechanism and server setup decide what type of call it actually is, but this example also shows how to pass a communication ID, so that the client service can be replied to later on, by the server service. If you have extended the base `Auto` class with your own class, the message request could look like:

```
//your service
YourServiceAsClient extends Auto
{
    //your method
    public void makeAsynchronousCall()
    {
        Element serviceUri = Handle.createNewUrlHandle(HttpServer.getIPAddress());
        serviceUri = Handle.addToHandle(serviceUri,
                                        Handle.asHandleElement("serviceToCall"));

        //parameters to some method
        parameters = new ArrayList();
        parameters.add(someData);

        //automatic method construction
        methodInfo = MethodFactory.createMethodCall("callAsynchronous",
                                TypeConst.VOID, serviceUri, parameters, passwordHandler);

        //additional parameters for agent-based communication
        //a communication ID, should be unique
        //also include the client URI to give the reply address
        commID = createUniqueID();
        methodInfo.setCommunicationID(commID);
        methodInfo.setClientURI(getFullPath());
        call.call(methodInfo);
    }
}
```

The service being invoked, could then have a method that looks like the following:

```
//your service on the server
YourServiceOnServer extends Auto
{
    public void callAsynchronous(Object someData) throws Exception
    {
        final String reply;                 //message reply
        final String thisCommID;            //this communication id
        final ArrayList parameters;             //method parameters
        final Element serviceUri;           //uri of reply service
```

```
        final MethodInfo methodInfo;            //method info
        final CallObject call;                  //object to make the call
        Thread executionThread;                 //execution thread

        //get the last communication id that was added
        //this is automatically stored by the system in the named variable
        thisCommID = lastAddedCommunicationID;

        //add any reply
        reply = (uuid + ": This is an asynchronous message reply");

        //add parameters, including the reply, for the reply method
        parameters = new ArrayList();
        parameters.addElement(reply);

        //use the last communication id to get the related client uri
        serviceUri = (Element)communicationIDs.get(thisCommID);
        communicationIDs.remove(thisCommID);

        call = new CallObject();
        methodInfo = MethodHandler.createMethodCall("messageReply",
                        TypeConst.BOOLEAN, serviceUri, parameters, passwordHandler);

        //for an agent-based communication, you can do this again
        methodInfo.setCommunicationID(thisCommID);
        methodInfo.setClientURI(getFullPath());

        //could start a new thread to process the request, but not required
        executionThread = new Thread(new Runnable() {
            public void run()
            {
                try
                {
                    Thread.sleep(5000);

                    //Add your own functionality here to process the request
                    //if you break this thread then other threads can run,
                    //when this thread can process the request and reply
                    //later. Thread.sleep can be used to test this.

                    call.call(methodInfo);
                }
                catch (Exception ex)
                {}
            }
        });

        executionThread.start();
    }
}
```

- If the client includes a communication ID and a client URI in the request method info, this is automatically saved and retrieved by the service, so long as it extends the `Auto` class.
- The last communication ID that was added is also stored in a separate variable called lastAddedCommunicationID. This variable can be accessed from the Auto-derived class and should represent the method that is currently being invoked.
- If your service is receiving calls from different clients, each will send their messages under a unique communication ID.
- When your Auto-derived service receives it, the first method to be invoked can use the lastAddedCommunicationID variable to tell it which client the call came from. It will then know what communication or message thread to process. This should then be noted and can be used later in the method to retrieve the client URI for the message reply.
- The default framework would suggest to always send a reply to the `messageReply` method on the client Auto-derived service. It can then be processed further from there.