Chapter 3. SAP BPM and PI Tuning

Business Process Management (Integration Processes) design includes many helpful functionalities which can rarely be replaced by anything else. The articles in this chapter will cover some of these. Additionally, every company will be required to perform a SAP PI tuning at some point, and the information covered in this chapter will make this easier to complete. As IDocs are still the most commonly used interfaces in this chapter we focus on IDoc tuning and present articles that show how to do IDoc bundling, IDoc tunneling and how to collect IDocs in a proper manner. You will also find information on how to schedule your communication channels so they are not running continuously, if not necessary. As SAP Process Integration sends messages via different technologies (RFC, HTTP, etc.) sometimes we might get a timeout exception and one of the articles in this chapter deals with many timeouts that can happen during a message flow inside SAP PI. Tuning also means having correct design of your interfaces, which is why we've included an article and tips on ABAP proxy implementations.

3.1 How to retrieve the MESSAGE_ID from a BPM

Introduction

This article shows a typical scenario where you need a Technical Context Object in a BPM and you are unable to retrieve it using the Transformation step in the BPM.

Article

This article is a summary of one of the recent discussions on the PI/XI forum on SDN on the possibilities of retrieving MESSAGE_ID from a BPM.

There are many times when we need Technical Context Objects such as MESSAGE_ID in our BPM:

- we want to create a file with a message_id as its name
- we need message_id to correlate request and response messages inside the BPM

Unfortunately some of these technical objects (message id for instance) are not available in the BPM; they can't be retrieved in a Transformation step.

The only way to retrieve them is to use an additional mapping step in the Interface determination.

1) Let's suppose we have a simple BPM. (Figure 84)

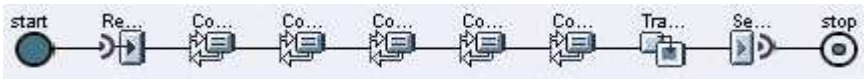

Figure 84

2) We've decided that we want to have the incoming message ID available to the fields of the abstract interface. We create a Simple Java function that will map the MESSAGE_ID to one of the fields of our abstract interface. (Figure 85)

Figure 85

3) The code of the simple java function:

```
String constant;
java.util.Map map;
map = container.getTransformationParameters();
```

```
constant = (String)
map.get(StreamTransformationConstants.MESSAGE_ID);
return constant;
```

4) We create mapping interface on the basis of the mapping program that we've just prepared.

5) Now we have to add interface mapping to our interface determination step in XI Directory. (Figure 86)

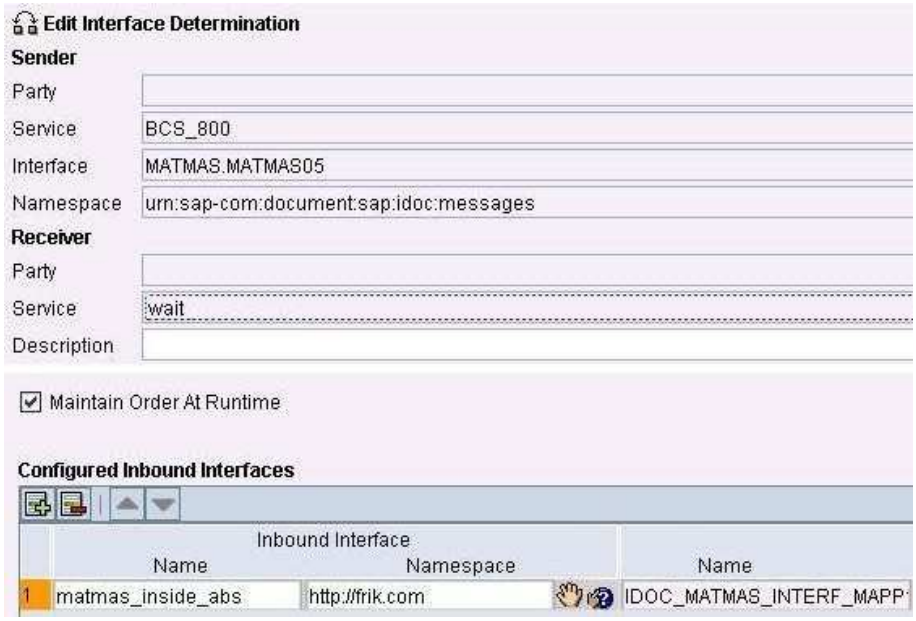

Figure 86

Then, after we send the message we can easily see that we received MESSAGE_ID in our abstract interface (so, in the BPM). The message id is ready to be used in our integration process. (Figure 87)

Figure 87