Process Integration (PI)
Performance Check -
Analyzing Performance
Problems and Possible
Solution Strategies

Document Version 1.0 – December 2009
## Typographic Conventions

<table>
<thead>
<tr>
<th>Type Style</th>
<th>Represents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example Text</td>
<td>Words or characters that appear on the screen. These include field names, screen titles, pushbuttons as well as menu names, paths and options. Cross-references to other documentation.</td>
</tr>
<tr>
<td>Example Text</td>
<td>Emphasized words or phrases in body text, titles of graphics and tables.</td>
</tr>
<tr>
<td>EXAMPLE TEXT</td>
<td>Names of elements in the system. These include report names, program names, transaction codes, table names, and individual key words of a programming language, when surrounded by body text, for example, SELECT and INCLUDE.</td>
</tr>
<tr>
<td>Example Text</td>
<td>Screen output. This includes file and directory names and their paths, messages, names of variables and parameters, source code as well as names of installation, upgrade and database tools.</td>
</tr>
<tr>
<td>Example Text</td>
<td>Exact user entry. These are words or characters that you enter in the system exactly as they appear in the documentation.</td>
</tr>
<tr>
<td>&lt;Example Text&gt;</td>
<td>Variable user entry. Pointed brackets indicate that you replace these words and characters with appropriate entries.</td>
</tr>
<tr>
<td>EXAMPLE TEXT</td>
<td>Keys on the keyboard, for example, function keys (such as F2) or the ENTER key.</td>
</tr>
</tbody>
</table>

## Icons

<table>
<thead>
<tr>
<th>Icon</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>⚠️</td>
<td>Caution</td>
</tr>
<tr>
<td>📜</td>
<td>Example</td>
</tr>
<tr>
<td>📝</td>
<td>Note</td>
</tr>
<tr>
<td>🌟</td>
<td>Recommendation</td>
</tr>
<tr>
<td>⌨️</td>
<td>Syntax</td>
</tr>
</tbody>
</table>

THE BEST-RUN E-BUSINESSES RUN SAP
1 Introduction

SAP NetWeaver Process Integration (PI) consists of the following functional components: Integration Builder (including Enterprise Service Repository (ESR), Service Registry (SR) and Integration Directory), Integration Server (including Integration Engine, Business Process Engine, Adapter Engine), Runtime Workbench (RWB), and System Landscape Directory (SLD). The SLD, in contrast to the other components, may not necessarily be part of the PI system in your system landscape because it is used by several SAP NetWeaver products (however it will be accessed by the PI system regularly). Additional components in your PI landscape might be the Partner Connectivity Kit (PCK), a J2SE Adapter Engine, or the decentral J2EE Adapter Engine.

These components can be used at design time and at runtime (see the following graphic). The communication and accessibility of these components can be checked using the PI Readiness Check (SAP Note 817920).

Processing at runtime is carried out by the Integration Server (IS) with the aid of the SLD. The IS in this graphic stands for the Web Application Server 7.1, installed as a double stack: an ABAP and a Java stack (J2EE Engine). In the classic PI environment (normal Adapter Engine processing), the ABAP stack is responsible for pipeline processing in the Integration Engine (Receiver Determination, Interface Determination, and so on) and the processing of integration processes in the Business Process Engine. The Java stack executes most of the mappings of messages (with the exception of ABAP mappings) and hosts the Adapter Framework (AFW). The Adapter Framework contains all XI adapters except the plain HTTP, WSRM, and IDoc adapters.

One of the main enhancements to SAP NetWeaver PI 7.1 is the Advanced Adapter Engine. The Adapter Engine was enhanced to provide additional routing and mapping call functionality so that it is possible to process messages locally. This means that a message that is handled by a sender and receiver adapter based on J2EE does not need to be forwarded to the ABAP Integration Engine. This saves a lot of internal communication, reduces the response time, and significantly increases the overall throughput.
The next graphic illustrates the components that are available in more detail. Note that the plain HTTP and WRMS adapters are not shown.

This graphic also gives a first impression of where a performance problem might be located: In the Integration Engine itself, in the Business Process Engine, or in one of the Adapter Engines (central, non-central, plain, PCK). Of course, a performance problem could also occur anywhere in between, for example, in the network or around a firewall. Note that there is a separation between a performance issue “in PI” and a performance issue in any attached systems: If viewed “through the eyes” of a message, (that is, from the point of view of the message flow), then the PI system technically starts as soon as the message enters an adapter or (if HTTP communication is used) as soon as the message enters the pipeline of the Integration Engine. Any delay prior to this must be handled by the sending system. The PI system technically ends as soon as the message reaches the target system, for example, a given receiver adapter (or in case of HTTP communication, the pipeline of the Integration Engine).
Server) has received the success message of the receiver. Any delay after this point in time must be analyzed in the receiving system.

There are generally two types of performance problems. The first is a more general statement that the PI system is slow and has a low performance level, or does not reach the expected throughput. The second is typically connected to a specific interface failing to meet the business expectation with regard to the processing time. The layout of this check is based on the latter: First you should try to determine the component that is responsible for the long processing time or the component that needs the highest absolute time for processing. Once this has been clarified, there is a set of transactions that will help you to analyze the origin of the performance problem.

If the recommendations given in this guide are not sufficient, SAP can help you to optimize your system if you ask for SAP Solution Management Optimization services of type “Interface Management Service (IFM)”, “Volume Test Optimization Service (VTO)”, or “Business Process Performance Optimization (BPPO)” (http://service.sap.com/smo). SAP might also handle smaller problems, restricted to a specific interface, if you describe your problem in a SAP customer message. The “Going Live Analysis (GA)” and “Going Live Verification (GV)” are free remote services for every customer with a PI license and they perform checks on the most important parameters. You can order any type of service using SAP Service Marketplace or your local SAP contacts.

If you have already worked with the PI Admin Check (SAP Note 884865) you will recognize some of the transactions. This is because SAP considers the regular checking of the performance to be an important administrative task. However, this check tries to get the approach to performance problems across to its reader. Also, it offers the most common reasons for performance problems and links to possible follow-up actions but does not refer to any regular administrative transactions.

Wily Introscope is a prerequisite for analyzing performance problems at the Java side. It allows you to monitor the resource usage of multiple J2EE server nodes and provides information about all the important components on the Java stack, like mapping runtime, messaging system, or module processor. Furthermore, the data collected by Wily is stored for several days so it is still possible to analyze a performance problem at a later date. Wily Introscope is provided free-of-charge for SAP customers. It is delivered as part of Solution Manager Diagnostics but can also be installed separately. For more information, see http://service.sap.com/diagnostics.
2 Working with This Document

If you are experiencing performance problems on your Process Integration system, start with Chapter 3: Determining the Bottleneck. It helps you to determine the processing time for the different runtimes within PI: Integration Engine, Business Process Engine and Adapter Engine, or connected Proxy System. Once you have identified the area in which the bottleneck is most probably located, continue with the relevant chapter. This is not always easy to do because a long processing time is not always an indication for a bottleneck: It can make sense to do this if a complicated step is involved (Business Process Engine), if an extensive mapping is to be executed (IS), or if the payload is quite large (all runtimes). For this reason you need to compare the value that you retrieve from chapter 3 with values that you have received previously, for example. The history data provided for the Java components by Wily Introscope is a big help. If this is not possible, you will have to work with a hypothesis.

Once the area of concern has been identified (or your first assumption leads you there), Chapters 4, 5, 6, and 7 will help you to analyze the Integration Engine, the Business Process Engine, the PI Adapter Framework, and ABAP Proxy runtime.

After (or preferably during) the analysis of the different process engines, it is important to keep in mind that the bottlenecks you have observed are possibly caused by other interfaces processing at the same time. It will lead you to slightly different conclusions and tuning measures. You therefore have to distinguish between the following if the problem occurs:

A. with regard to the interface itself, that is, it occurs even if a single message of this interface is processed
B. or with regard to the volume of this interface, that is, many messages of this interface are processed at the same time
C. or with regard to the overall message volume processed on PI.

This can be analyzed by repeating the procedure of Chapter 3 for A) a single message that is processed while no other messages of this interface are processed and while no other interfaces are running. Then compare this value with B) the processing time for a typical amount of messages of this interface, not simply one message as before. If the values of measurement A) and B) are similar, repeat the procedure with C) a typical volume of all interfaces, that is, during a representative timeframe on your productive PI system or with the help of a tailored volume test.

These three measurements – A) processing time of a single message of a single interface B) processing time of a typical amount of messages of a single interface C) processing time of a typical amount of messages of all interfaces – should enable you to distinguish between the three possible situations:

- A specific interface has long processing steps that need to be identified and improved. This situation typically requires a re-design of an interface.
- The mass processing of an interface leads to high processing times. This situation typically calls for tuning measures.
- The long processing time is a result of the overall load on the system. This situation can be solved by tuning measures and by taking advantage of PI features, for example, to establish separation of interfaces. If the bottleneck is hardware-related it could also require a re-sizing of the hardware that is used.

Chapters 10, 11, and 12 deal with more general reasons for bad performance, such as heavy tracing/logging, error situations, and general hardware problems. They should be taken into account if the reason for slow processing cannot be found easily or if situation C from above applies (long processing times due to a high overall load).
**Important**

Chapter 0 provides the basic checks for the hardware of the PI server. It should not only be used when analyzing a hardware bottleneck due to a high overall load and/or an insufficient sizing but should also be used after every change made to the configuration of PI to ensure that the hardware is able to handle the new situation. This is important because a PI system uses three runtime engines (IS, BPE, AFW). Tuning one engine for high throughput might have a direct impact on the others. With every tuning action applied, you have to be aware of the consequences on the other runtimes or the available hardware resources.
3 Determining the Bottleneck

The total processing time in PI consists of the processing time in the Integration Server, processing time in the Business Process Engine (if ccBPM is used), as well as the processing time in the Adapter Framework (if adapters except IDoc, plain HTTP, or ABAP proxies are used). In the subsequent chapters you will find a detailed description of how to obtain these processing times.

For reasons of completeness we will also have a look at the ABAP Proxy runtime and the available performance evaluations.

3.1 Integration Engine Processing Time

You can use an aggregated view of the performance data for the Integration Engine as shown below.

**Procedure**

Log on to your Integration Engine and call transaction SXMB_IFR. In the browser window, follow the link to the Runtime Workbench. In there, click “Performance Monitoring”. Change the display to “Detailed Data Aggregated”, select ‘IS’ as the Data Source (or ‘PMI’ if you have configured the Process Monitoring Infrastructure”) and “XIIntegrationServer/<your_host>” as the component. Then choose an appropriate time interval, for example, the last day. You have to enter the details of the specific interface you want to monitor here.

- The interesting value is the “Processing Time [s]” in the fifth column. It is the sum of the processing time of the single steps in the Integration Server. For now you are not interested in the single steps that are listed on the right side since you are still trying to find out the part of the PI system where the most time is lost. Chapter 4.4 will work with the individual steps.

If you do not know which interface is affected yet, you first have to get an overview. Instead of navigating to **Detailed Data Aggregated** in the Runtime Workbench, choose **Overview Aggregated**. Use the button **Display Options** and check the options for sender component, receiver component, sender interface, and receiver interface. Check the processing times as described above.
3.2 Adapter Engine Processing Time

Procedure

Log on to your Integration Server and call transaction SXMB_IFR. In the browser window follow the link to the Runtime Workbench. In there, click Message Monitoring, choose Adapter Engine <host> from Database from the drop down lists, and press Display. Enter your interface details and “Start” the selection. Select one message using the radio button and press Details.

- Calculate the difference between the start and end timestamp indicated on the screen.
- Do the above calculation for the outbound (AFW -> IS) as well as the inbound (IS -> AFW) messages.
- The audit log of successful message is no longer persisted in SAP NetWeaver PI 7.1 by default in order to minimize the load on the database. Instead, a cache is used that keeps the audit log information for a period of time until the cache entries are displaced. Thus, no detailed information is provided from a history point of view. You can use Wily Introscope for analyses of this type.

The audit log can be persisted temporarily on the database to allow historical analysis of performance problems on the Java stack. To do so, change the parameter “messaging.auditLog.memoryCache” from true to false for service “XPI Service: Messaging System” using NetWeaver Administrator (NWA). Note: Only do this temporarily if you have identified the bottleneck on the AFW. First try using Wily to determine the root cause of the problem.

- Since there is no aggregated overview as for the Integration Engine, try to get an overview of more than one message. It is usually enough to look at the list that the Message Monitoring provides and to assume that the processing time in the adapter itself (that is the data in the details) will remain more or less constant.
## 3.3 Processing Time in the Business Process Engine

**Procedure**

Log on to your integration server and call transaction SXMB_MONI. Adjust your selection in a way that you select one or more messages of the respective interface. Once the messages are listed, navigate to the **Outbound** column (or **Inbound** if your integration process is the sender) and click on **PE**. Alternatively, you can select the radio button for “**Process View**” on the entrance/selection screen of SXMB_MONI.

- To judge the performance of an integration process, it is essential to understand the steps that are executed. A long-running integration process in itself is not critical because it can wait for a trigger event. Therefore, it is important that you understand the steps that are included before analyzing an individual workflow log. For example, the screenshot below shows a long waiting time after the mapping. This is perhaps due to an integrated wait step.

- Calculate the time difference between the first and the last entry. Note that this time for the workflow does not include the duration of RFC calls, for example. To see this processing time, navigate to the “**List with Technical Details**” (second button from the left on the screenshot below or shift + F9).

- Since there is (currently) no aggregated overview, repeat the procedure for several processes of the specific workflow to get a better overview.
4 Analyzing the Integration Engine

If chapter Determining the Bottleneck has shown that the bottleneck is clearly in the Integration Engine, there are several transactions that can help you to analyze the reason for this. To understand why this selection of transactions helps to analyze the problem, it is important to know that the processing within the Integration Engine is done within the pipeline. The central pipeline of the Integration Server executes the following steps (some minor steps are not listed):

PLSRV_XML_VALIDATION_RQ_INB  (XML Validation Inbound Channel Request)
PLSRV_RECEIVER_DETERMINATION  (Receiver Determination)
PLSRV_INTERFACE_DETERMINATION (Interface Determination)
PLSRV_RECEIVER_MESSAGE_SPLIT (Branching of Messages)
PLSRV_MAPPING_REQUEST    (Mapping)
PLSRV_OUTBOUND_BINDING   (Technical Routing)
PLSRV_XML_VALIDATION_RQ_OUT (XML Validation Outbound Channel Request)
PLSRV_CALL_ADAPTER     (Transfer to Respective Adapter: IDoc, HTTP, Proxy, AFW)
PLSRV_XML_VALIDATION_RS_INBXML (Validation Inbound Channel Response)
PLSRV_MAPPING_RESPONSE   (Request Message Mapping)
PLSRV_XML_VALIDATION_RS_OUT (Validation Outbound Channel Response)

The steps indicated in red are new in SAP NetWeaver PI 7.1 and can be used to validate the payload of a PI message against an XML schema. These steps are optional and can be executed at different times in the PI pipeline processing, for example, before and after a mapping (as shown above).

The last three steps highlighted in bold are only available in case of synchronous messages and reflect the time spent in the mapping of the synchronous response message.

Equally important to know is that these pipeline steps are executed based on qRFC Logical Unit of Work (LUW) by using dialog work processes.

4.1 Work Process Overview (SM50 for each Application Server, SM66 Globally)

The work process overview is the central transaction to get an overview of the current processes running in the Integration Engine. The process overview is only a snapshot analysis. You therefore have to refresh the data multiple times to get a feeling for the dynamics of the processes. The most important questions to be asked are as follows:

✓ How many work processes are being used on average?
✓ Which user is using these processes? The Business Process Engine (ccBPM) will show up with user WF-BATCH, whereas the pipeline execution is typically executed by PIAFUSER (XIAFUSER, if upgraded). The latter user, however, is not a fixed user, but the last one that activated the inbound scheduler (see SAP Note 369007 - qRFC: Configuration for the QIN Scheduler, question 8).
✓ If you see high WF-BATCH activity, take a look at chapter 5.1
✓ Are there any long running processes and, if so, what are these processes doing with regard to the reports used and the tables accessed?
  ○ As a rule of thumb for the initial configuration, the number of DIA work processes should be around six times the value of CPU in your PI system (rdisp/wp_no_dia = 6*#CPUs + 10 based on SAP Note 1375656 - SAP Netweaver PI System Parameters).

If you would like to get an overview for an extended period of time without actually refreshing the transaction at regular intervals, use report /SDF/MON. It allows you to collect metrics such as CPU usage, amount of free Dialog work processes, and, most
importantly, a snapshot of the work process activity as provided in transactions SM50 and SM66. The frequency for the data collection can be as low as 10 seconds and the data can be viewed after the configurable timeframe of the analysis.

If you have Solution Manager Diagnostics installed and all the agents connected to PI, you can also monitor the ABAP resource usage using Wily Introscope. As you can see in the dashboard below, you can use it to monitor the number of free work processes (prerequisite is that the SMD agent is running on the PI system). The major advantage of Wily Introscope is that this information is also available from the past and allows analysis after the problem has occurred in the system.

4.2 qRFC Resources (SARFC)

Depending on the configuration of the RFC server group, not all dialog work processes can be used for qRFC processing in the Integration Server. The current resource situation can be monitored using transaction SARFC.

Procedure

Call transaction SARFC and refresh several times since the values are only snapshot values.

- Is the value for “Max. Resources” close to your overall number of dialog work processes? Max. resources is a fixed value and describes how many DIA work processes may be used for RFC communication. If the value is too low, your RFC parameters need tuning to provide the Integration Server with enough resource for the pipeline processing. The RFC parameters can be displayed by double-clicking on “Server Name”.

- A good starting point is to set the values as follows:
  - Max. no. of logons = 90
  - Max. disp. of own logons = 90
  - Max. no. of WPs used = 90
  - Min. no. of free WP = 3-10 according to SAP Note 1375656 – SAP NetWeaver PI System Parameters (this parameter must be increased if you plan to have synchronous interfaces or high traffic on the AAE (ABAP resources required for user management). To avoid any bottleneck and blocking situation, free DIA WPs should always be available. The value should be adjusted based on the overall number of DIA WPs available at the system and the above requirements.)
  - Max. wait time = 5

Note: You have to set the parameters using the SAP instance profile. Otherwise the changes are lost after the server is restarted.

- Is the value for “Resources” reasonably high, for example, above zero all the time? If the “Resources” value is zero, then the Integration Server cannot process XML messages because no dialog work process is free to execute the necessary qRFC. There are several conclusions that can be drawn from this observation as follows:
  1) Depending on the CPU usage, (see chapter 9.1) it might be necessary at this time to increase the number of dialog work processes in your system. If the CPU usage,
however, is already very high, increasing the number of DIA will not solve the bottleneck.

2) Depending on the number of concurrent queues (see chapter qRFC Queues (SMQ2)) it might be necessary to decrease the degree of parallelism in your Integration Server because the resources are obviously not sufficient to handle the load. The next section describes how to tune PI inbound and outbound queues.

The data provided in SARFC is also collected and shown in a graphical and historical way using Wily Introscope (as can be seen in the screenshot below). This allows easy monitoring of the RFC resources across all available PI application servers.

4.3 qRFC Queues (SMQ2)

The qRFC inbound queues on the ABAP stack are one of the most important areas in PI tuning. Therefore it is essential to understand the PI queuing concept.

PI generally uses two types of queues for message processing – PI inbound and PI outbound queues. Both types are technical qRFC inbound queues – and can therefore be monitored using SMQ2. PI inbound queues are named XBTI* or XBQI* and are shared between all interfaces running on PI by default. The PI outbound queues are named XBTO* and XBQO* and have a -specific suffix (in red: XBTO00004). PI outbound queues are separated for each receiver system which means that all interfaces of one receiver business system are contained in the same queue.

PI inbound and outbound queues process different steps in the PI pipeline as described below:

1) PI Inbound Queues:
- PLSRV_XML_VALIDATION_RO_INB (XML Validation Inbound Channel Request)
- PLSRV_RECEIVER_DETERMINATION (Receiver Determination)
- PLSRV_INTERFACE_DETERMINATION (Interface Determination)
- PLSRV_RECEIVER_MESSAGE_SPLIT (Branching of Messages)

2) PI Outbound Queues:
Looking at the steps above it is clear that tuning the queues will have a direct impact on the connected PI components. For example, by increasing the number of parallel outbound queues more mappings will be executed in parallel which will in turn put a greater load on the Java stack or more messages will be forwarded to the backend system. Thus, when tuning PI queues you must always keep the implications for the connected systems in mind.

Tuning the parallelism of PI queues can be done in transaction SXMB_ADM ➔ Integration Engine Configuration and by selecting the category TUNING. The relevant parameters are EO_INBOUND_PARALLEL and EO_OUTBOUND_PARALLEL.

Below you can see a screenshot of SMQ2. You can see the PI inbound queues and outbound queues. Also ccBPM queues are displayed and will be discussed later.

**Procedure**

Log on to your Integration Server, call transaction SMQ2, and execute. If you are running ABAP proxies on different clients, enter ‘*’ for the client. Transaction SMQ2 provides snapshots only, and must therefore be refreshed several times to get viable information.

- The first value to check is the number of queues concurrently active over a period of time. Since each queue needs a dialog work process to be worked on, the number of concurrent queues must not be too high compared to the number of dialog work processes usable for RFC communication (see transaction SARFC). The optimum throughput can be achieved if the number of active queues is equal to or slightly higher than the number of dialog work processes (provided that the number of DIA work processes can be handled by the CPU of the system, see transaction ST06).

- The second value of importance is the number of entries. Hit refresh a few times to check if the numbers increase/decrease/remain the same. An increase of entries for all queues, or a specific queue, points to a bottleneck. The conclusion that can be drawn from this is not simple, possible reasons have been found to include:
  1) A slow step in a specific interface, for example, the mapping step or receiver determination. This can be confirmed by looking at the processing time for each step (see chapter Pipeline Steps (SXMB_MONI or RWB)).
  2) Check if inbound or outbound queues face the backlog. The parallelism for the queues that are involved might be too low. This can be tuned by increasing the number of queues as described above. In such a case PI Message Packaging can also help.
  3) Queues stay in status READY in SMQ2:
     To see the status of the queues, use the filter button in SMQ2 as shown below.
If you observe a situation where many queues are in READY status, the following situations can apply:

- A resource bottleneck with regard to RFC resources. Confirm with transaction SARFC.
- Long DB loading times for RFC tables. If using Oracle, ensure that SAP Note 1020260 - Delivery of Oracle statistics is applied.
- To avoid overloading a system with RFC load, the QIN scheduler only reloads LUWs from the queues after a certain amount of DIA WPs have finished their work. This can lead to long waiting times in the queues as explained in SAP Note 1115861 - Behaviour of Inbound Scheduler after Resource Bottleneck. Since PI is a non-user system, we recommend setting rfc/inb_sched_resource_threshold to 3.
- Additional overhead during this step occurs due to every message being blocked during processing. This is no longer necessary and should be switched off by setting the parameter LOCK_MESSAGE of category RUNTIME to 0 as described in SAP Note 1058915.

4) An error situation is blocking one or more queues:

- Click the Alarm Bell (Change View) push button once to see only queues with error status. Errors delay the queue processing within the Integration Server and may decrease the throughput if, for example, multiple retries occur. Navigate to the first entry of the queue and analyze the problem by forward navigation to the relevant message in SXMB_MONI.

The above is also true for inbound queues with the SMD agent offering monitoring using Wily Introscope. For qRFC queues, the data is collected in five-minute interval. In the screenshot below you can see a blocked queue in a system. In this example, the blocked queue (with one entry) is displayed every 5 minutes but the age of the oldest qRFC inbound queue increases over time (lower-right corner). With the KPIs provided here, it is also possible to distinguish a blocking situation or a general resource problem after the problem is no longer present in the system.
4.4 Pipeline Steps (SXMB_MONI or RWB)

The duration of the pipeline steps is the ultimate answer to long processing times in the Integration Engine since it describes exactly how much time was spent at which point. The recommended way to retrieve the duration of the pipeline steps is the RWB and will be described below. Advanced users may use the Performance Header of the SOAP message using transaction SXMB_MONI, but the timestamps are not easy to read. If you still prefer the latter method, here is the explanation: 20091109092656.165 must be read as yyyymmddhh(min)(min)(sec)(sec).(microseconds), that is, the timestamp corresponds to November 9, 2009, at 09:25:56 and 165 microseconds.

Procedure

Log on to your Integration Server and call transaction SXMB_IFR. In the browser window follow the link to the Runtime Workbench. In there, click Performance Monitoring. Change the display to Detailed Data Aggregated and choose an appropriate time interval, for example, the last day. For this selection you have to enter the details of the specific interface you want to monitor.

✓ You will now see in the lower-part of the screen a split of the processing time into single steps. Check the time difference between the steps. Does any step take longer than expected? In the example in the screenshot below, the DB_ENTRY_QUEUEING starts after 0.032 seconds and ends after 0.243 seconds which means it took 0.211 seconds (211ms).
Compare the processing times for the single steps for different measurements as outlined in chapter "Pipeline Steps (SXMB_MONI or RWB)", for example, is a single step only long if many messages are processed or if a single message is processed? This helps you to decide if the problem is a general design problem (single message has long processing step) or if it is related to the message volume (only for a high number of messages this process step has large values).

Each step has different follow-up actions that are described next.

4.4.1 Long Processing Times for “PLSRV_RECEIVER_DETERMINATION” / PLSRV_INTERFACE_DETERMINATION

Steps that can take a long time in inbound processing are the Receiver and Interface Determination. In these steps, the receiver system and interface is calculated. Normally this is very fast but PI offers the possibility of enhanced receiver determinations. In these cases the calculation is based on the payload of a message. There are different implementation options:

- Content-Based Routing (CBR):
  CBR allows defining XPath expressions that are evaluated during runtime. These expressions can be combined by logical operators, for example, to check the value for multiple fields. The processing time of such a request directly correlates to the number and complexity of the conditions defined.

- No options exist for the system in regard to CBR. The performance of this step can only be changed by changing the design of the interfaces. If the number of conditions is very high and experience shows that a mapping program (as below) is faster then CBR.

- Mapping to determine Receivers:
  A standard PI mapping (ABAP, Java, XSLT) can also be used to determine the
receiver. If you observe high runtimes in such an interface determination, follow the steps outlined in the next section since the same runtime is used.

### 4.4.2 Long Processing Times for “PLSRV_MAPPING_REQUEST”

Before analyzing the mapping, you must understand which runtime is used. Mappings can be implemented in ABAP, as graphical mappings in the Enterprise Service Builder, as self-developed Java coding, or XSLT mappings. Normal debugging and tracing tools (transaction ST12) can be used for mappings executed in ABAP. Mappings running on the Java stack will be discussed next.

The mapping response time is heavily influenced by the complexity of the mapping and the message size. Therefore, to analyze a performance problem in the mapping environment you have to compare the mapping runtime during the time of the problem with values reported several days earlier to get a better understanding.

If no AAE is used, the starting point of a mapping is the Integration Engine that will send the request by RFC destination AI_RUNTIME_JCOSERVER to the gateway. There it will be picked up by a registered server program. The registered server program belongs to the J2EE Engine. The request will be forwarded to the J2EE Engine by a JCo call and then executed by the Java runtime. When the mapping has been executed, the result is sent back to the ABAP pipeline using JCo. There are therefore multiple places to check when trying to determine why the mapping step took so long.

**Affected interfaces**: Before analyzing the mapping runtime of the PI system, check if only one interface is affected or if you face a long mapping runtime for different interfaces. To do so, check the mapping runtime of messages being processed at the same time in the system.

The best tool for such an analysis is Wily Introscope which offers a dashboard for all mappings being executed at a given time. Each line in the dashboard represents one mapping and shows the average response time and the number of invocations.

In the screenshot below you can see that many different mapping steps have required around 500 seconds for processing. In this case, it is very likely that the underlying J2EE engine faces some major problems as described in section [J2EE Engine Bottleneck](#).
If there is only one mapping that faces performance problems, there would be just one line sticking out in all the graphs. If you face a general problem that affects different interfaces, you can choose a longer timeframe that allows you to compare the processing times in a different time period and verify if it is only a “temporary” issue – this would, for example, indicate an overload of the mapping runtime.

If you have found out that only one interface is affected, then it is very unlikely to be a system problem but rather a problem in the implementation of the mapping of that specific interface.

- Check the message size of the mapping in the runtime header using SXMB_MONI. Verify if the message size is larger than usual (which would explain the longer runtime).

- There can also be a lookup to a remote system included in a mapping (using RFC or JDBC). Together with the application, you then have to check if the connection to the backend is working properly. Such a lookup can also be analyzed using Wily Transaction Trace as explained in the appendix section Wily Transaction Trace.

If not one but several interfaces are affected, a potential system bottleneck occurs and this is described in the following.

- Not enough resources (registered server programs) available. That could either be the case if too many mapping requests are issued at the same time or if one J2EE server node is down and has not registered any server programs.

- To check if there were too many mapping requests for the available registered server programs, compare the number of outbound queues that are concurrently active with the number of registered server programs. The number of outbound queues that are concurrently active can be monitored with transaction SMQ2 and counting queues with the names XBTO* & XBQO*, XBTA*, and XBOA* (high priority) XBTZ* and XBQZ* (low priority), and XBTM* (large messages). In addition to this you have to take into account mapping steps that are executed in a ccBPM process. Mappings executed by a ccBPM process are not queued, but the ccBPM process calls the mapping program directly using tRFC. The number of registered server programs can be determined with transaction SMGW → Goto → Logged On Clients and filtering by the program ("TP Name") AL_RUNTIME_<SID>.

- If the problem occurred in the past, it is more difficult to determine the number of queues that are concurrently active. One way is to use transaction SXMB_MONI and specify every possible outbound queue (field “Queue ID”) in the advanced selection criteria (note: Wildcard search not available). The timeframe can be restricted in the upper-part of the advanced selection criteria. Advanced users could use transaction SE16 to search directly in table SXMSMAST, using the field “QUEUEINT”.

![Gateway Monitor for ncev00000012a.dhcp.ncel.sap.corp / Connections to](image-url)
The other option is to use the “qRFC Inbound Queue Detail” dashboard as described in qRFC Queue Monitoring (SMQ2).

\checkmark \ To check if the J2EE is not available, or if the server program is not registered for a different reason, use transaction SM59 to test the RFC destination AI_RUNTIME_JCOSERVER. If the problem occurred in the past, search the gateway trace (transaction SMGW \rightarrow GoTo \rightarrow Trace \rightarrow Gateway \rightarrow Display File) and the RFC developer traces dev_rfc* files available in the work directory for the string “AI_RUNTIME”. In addition, check the std_server<n>.out in the work directory for restarts of the J2EE Engine.

- Depending on the checks above, there are two options to resolve the bottleneck. Of course, if the J2EE was not available, the reason for this has to be found and prevented in the future. No further tuning needed.
  
  The two options for tuning are:
  1) the number of outbound queues that are concurrently active and
  2) the number of mapping connections from the J2EE server to the ABAP gateway.

\checkmark \ The increase of mapping connections is only recommended for strong J2EE servers since each mapping thread needs resources like CPU and Heap memory. Too many mapping connections mean too many mappings being executed on the J2EE Engine. In turn, this might reduce the performance of other parts of the PI system, for example, the pipeline processing in the Integration Engine or the processing within the adapters of the AFW, or can even cause out-of-memory errors on the J2EE engine.

\checkmark \ Add additional J2EE server nodes to balance the parallel requests. Each J2EE server node will register new destinations at the gateway and will therefore take a part of the mapping load.

\checkmark \ Another option is to reduce the number of outbound queues that are concurrently active to solve the bottleneck. This will certainly result in higher backlogs in the queue and is therefore only applicable for less runtime critical interfaces. This option can be used if the backlog is caused by a master data interface with high volume, but no critical processing time requirements. In such a case you can only lower the number of parallel queues for this interface by using a subparameter of EO_OUTBOUND_PARALLEL in transaction SXMB_ADM. By doing so, you slow down one interface to ensure other (more critical) interfaces have sufficient resources available.

- If multiple application servers configured on the PI system ensure that all the server nodes are connected to their local gateway, as described in the guide How To Scale PI.

### 4.4.3 Long Processing Times for “PLSRV_CALL_ADAPTER”

Call adapter is the last step executed in the PI pipeline and forwards the message to the next component along the message flow. In most cases (local Adapter Engine, IDoc, or BPE) this is a local call. The IDoc adapter only puts the message on the IRFC layer (SM58) so that the actual transfer of the IDoc is not included in the performance header. But network is also involved for ABAP proxies or forwarding the message to a decentral Adapter Engine.

Looking at the processing time, we have to distinguish asynchronous (EO and EOIO) and synchronous (BE) interfaces.

In asynchronous interfaces, the call adapter step includes the transfer of the message by the network and the initial steps on the receiver side (in most cases this just means persisting the message on the receiver’s database). A long duration can therefore have two reasons:

- Network latency: For large messages of several MB in particular, the network latency is an important factor (especially if the receiving ABAP proxy system is located on
another continent, for example). Network latency has to be checked in case of a long call adapter step.

- Insufficient resources on receiver side: Enough resources must be available at the receiver side to ensure quick processing of a message. Enough ICM threads and dialog work processes must be available in the case of an ABAP proxy system. Therefore, the analysis of long call adapter steps always has to include the relevant backend system.

For synchronous messages (request/response behavior) the call adapter step also includes the waiting time for the response message. Therefore the call adapter for synchronous messages includes the time of the transfer of the request message, the calculation of the corresponding response message at the receiver side, and the transfer back to PI. Therefore, the processing time of a request at the receiving target system for synchronous messages must always be analyzed to find the most costly processing steps.

### 4.4.4 Long Processing Times for “DB_ENTRY_QUEUEING”

The value for DB_ENTRY_QUEUEING describes the time that a message has spent waiting in a PI inbound queue before processing started. The inbound queues (XBTI*, XBT1*, XBT9*, XBTL* for EO messages and XBQI*, XBQ1*, XBQ9* for EOIO messages) process the pipeline steps for the Receiver Determination, the Interface Determination, and the Message Split (and optionally XML inbound validation).

Thus, if the step DB_ENTRY_QUEUEING has a high value, the inbound queues have to be monitored using transactions SMQ2 and SARFC. The reasons are similar as those outlined in chapter qRFC Resources (SARFC) and qRFC Queues (SMQ2).

- Not enough resources (DIA work processes for RFC communication) available.
- The number of parallel inbound queues is too low to handle the incoming messages in parallel. Note that a simple increase of the parameter EO_INBOUND_PARALLEL might not always be the solution (as enough work processes also have to be available to process the queues in parallel). The number of work processes is in turn restricted by the number of CPUs that are available for your system. If you would like to separate critical from non-critical sender systems, define a dedicated set of inbound queues by specifying a sender ID as a subparameter of EO_INBOUND_PARALLEL. For example, by doing so you can separate master data systems from business critical systems.
- The inbound queues were blocked by the first LUW. Use transaction SMQ2 to check if that is still the case. To identify if such a situation occurred in the past, use the Wily Introscope dashboard “ABAP System qRFC Inbound Detail”. It is generally not easy to find the one message that is blocking the queue. It might be achieved by checking RFC traces or by searching for a single message with a long pipeline processing step.
- Problems when loading the LUW by the QIN scheduler as described in qRFC Queues (SMQ2).

**Note:** If your system uses the parameter EO_INBOUND_TO_OUTBOUND = 0, you must also read chapter 4.4.5 for analyzing the reasons. EO_INBOUND_TO_OUTBOUND only determines whether inbound queues (value ‘0’) or inbound and outbound queues (value ‘1’) are used for the pipeline processing in the integration server. Check the value with transaction SXMB_ADM → Integration Engine Configuration → Specific Configuration. The default value is ‘1’, meaning the usage of inbound and outbound queues (the recommended behavior).

### 4.4.5 Long Processing Times for “DB_SPLITTER_QUEUEING”

The value for DB_SPLITTER_QUEUEING describes the time that a message has spent waiting in a PI outbound queue until a work process was assigned. The outbound queues
(XBTO*, XBTA*, XBZT*, XBTM* for EO messages and XBQO*, XBQA*, XBQZ* for EOIO messages) process the pipeline steps for the mapping, outbound binding, and call adapter. Thus, if the step DB_SPLITTER_QUEUEING has a high value, the outbound queues have to be monitored using transactions SMQ2 and SARFC. The reasons are similar as outlined in chapters qRFC_Resources (SARFC) and qRFC_ Queues (SMQ2) and listed above (use chapter 0).

o Not enough resources (DIA work processes for RFC communication) available.

o The number of parallel outbound queues is too low to handle the outgoing messages in parallel. Note that simply increasing the parameter EO_OUTBOUND_PARALLEL might not always be the solution as enough work processes have to be available to process the queues in parallel. The number of work processes in turn is limited by the number of CPUs available for the system. Also, the parameter EO_OUTBOUND_PARALLEL is used differently than the parameter EO_INBOUND_PARALLEL because it determines the degree of parallelism for each receiver system. Thus, it is possible to increase the number of parallel outbound queues for specific receivers whereas other receivers are handled with a lower degree of parallelism. Note that not every receiver backend is able to handle a high degree of parallelism so that it might be perfectly fine to observe a high value for DB_SPLITTER_QUEUEING.

o The outbound queues were blocked by the first LUW. To identify if such a situation occurred in the past, use the Wily Introscope dashboard “ABAP System qRFC Inbound Detail”. In general it is not easy to find the one message that is blocking the queue. It might be achieved by checking RFC traces or by searching for a single message with a long pipeline processing step.

o Problems when loading the LUW by the QIN scheduler as described in qRFC_ Queues (SMQ2).

o Since the outbound queues include a processing step that is bound to take longer than the other steps (that is, the mapping and the call adapter step) this might mean a long waiting time for all subsequent messages. As an example, assume that a mapping takes 1 second and that 100 messages are waiting in a queue. The first message gets executed at once, the second message has to wait about 1 second (a bit more since more steps are executed than just the mapping, but this should be ignored at the moment), the third takes 3 seconds, and so on. The 100th message has to wait 100 seconds until it is processed, that is, the value for DB_SPLITTER_QUEUEING is as high as 100. For a given outbound queue you would therefore see an increase for the DB_SPLITTER_QUEUEING value over time. If you experience this situation, proceed with chapter 4.4.2.

Note: The number of parallel outbound queues is also connected with the ability of the receiving system to process a specific amount of messages for each time unit.

- In section Adapter Parallelism, default restrictions of the specific adapters of the J2EE Engine are explained. Some adapters (JDBC, file, mail) can process only one message for each server node at a time (this can be changed). It does not make sense for such adapters to have too many parallel qRFC queues since this will only move the message backlog from ABAP to Java.

- For messages that are directed to the IDoc outbound adapter, the value for EO_OUTBOUND_PARALLEL is connected to the MAXCONN value of the corresponding outbound destination. You can define the maximum number of connections using transaction SMQS – the default value is 10 parallel connections. If applicable, SAP highly recommends that you use IDoc packaging in the outbound IDoc adapter to reduce the load during the transmission of tRFC calls from the PI system to the receiving system.

- For ABAP proxy systems, the number of outbound queues directly determines the number of messages sent in parallel to the receiving system.
Thus, you have to ensure that the resources available on PI are aligned with those on the sending/receiving ABAP proxy system.

### 4.5 PI Message Packaging

PI message packaging is a rather new feature that was introduced with SAP NetWeaver PI 7.0 SP13. Message packaging in BPE is independent of message packaging in PI (see chapter 5.6), but they can be used together.

To improve performance and message throughput, asynchronous messages can be assembled to packages and processed as one entity. For this the qRFC scheduler was enabled to process a set of messages (a package) in one LUW. Thus, instead of sending one message to the different pipeline steps (for example, mapping/routing), a package of messages will be sent that will reduce the number of context switches that is required. Furthermore, access to databases is more efficient since requests can be bundled in one database operation. Depending on the number and size of messages in the queue, this procedure improves performance considerably. In return, message packaging can increase the runtime for individual messages (latency) due to the delay in the packaging process.

Message packaging is only applicable for asynchronous scenarios (QoS EO and EIOI). Due to the potential latency of individual messages, packaging is not suitable for interfaces with very high runtime requirements. In general packaging has the highest benefit for interfaces with high volume and small message size.

The performance improvement achieved directly relates to the number of the messages bundled in each package. Message packaging must not solely be used in the PI system. Tests have shown that the performance improvement significantly increases if message packaging is configured end-to-end - that is, from the sending system using PI to the receiving system. Message packaging is mainly applicable for application systems connected to PI by ABAP proxy.

From the runtime perspective, no major changes are introduced when activating packaging. Messages remain individuals on the persistence layer and in the monitoring. Additional transactions are introduced allowing the monitoring of the packaging process.

Messages are also treated individually for error handling. If an error occurs in a message processed in a package, then the package will be disassembled and all messages will be processed as single messages. Of course, in a case where many errors occur (for example, due to interface design) this will reduce the benefits of message packaging. The sending and receiving applications in particular will not see any changes because they send and receive individual messages and receive individual commits.

The PI message packaging can be adapted to meet your specific needs. In general, the packaging is determined by three parameters:

1. **Message count**: Maximum number of messages in a package (default 100)
2. **Maximum package size**: Sum of all messages in kilobytes (default 1 MB)
3. **Delay time**: Time to wait before the queue is processed if the number of messages does not reach the message count (default 0 meaning no waiting time)

These values can be adjusted using transactions SXMS_BCONF and SXMS_BCM. To better use the performance improvements offered by packaging, you could, for example, define a specific packaging for interfaces with very small messages to allow up to 1,000 messages for each package. Another option could be to increase the waiting time (only if latency is not critical) to create bigger packages.

4.6 Tuning the IDoc Adapter

As stated above, the IDoc adapter uses the tRFC layer to transfer messages from PI to the receiver system. The IDoc adapter will only put the message in SM58 and from there the standard tRFC layer forwards the LUW. You therefore have to ensure that sufficient resources (mainly DIA WPs) are available for processing tRFC requests.

Wily Introscope also offers a dashboard (shown below) for the tRFC layer which shows the tRFC entries and their different statuses. With these dashboards you are also able to identify history backlogs on tRFC.

One option for improving the performance of messages processed with the IDoc adapter is to use IDoc packaging.

The receiver IDoc adapter in PI (for sending messages from PI to the receiving application system) already had the option for packaging IDocs in previous releases. For this, messages processed in the IDoc adapter are bundled together before being sent out to the receiving system. Thus, only one tRFC call is required to send multiple IDocs. The message packaging that is being discussed here uses a similar approach and therefore replaces the "old" packaging of IDocs in the receiver IDoc adapter.

For the sender IDoc adapter there was previously the option to activate packaging in the partner profile of the sending system. Multiple IDocs were also bundled together in one of these tRFC calls. At the PI side, these packages were disassembled by the IDoc adapter and PI processed the messages one by one.

This behaviour was changed with EHP 1 (SAP NetWeaver PI 7.11): If the sender backend sends an IDoc package to PI, a new option has been introduced which allows the processing of IDocs as a package on PI as well. You can specify an IDoc package size on the sender IDoc Communication Channel. The necessary configuration is described in detail in the following SDN blog - IDoc Package Processing Using Sender IDoc Adapter in PI EhP1. A significant increase in throughput was recorded for high volume interfaces with small IDocs.

In order to control the resources used when sending the IDocs from PI to the receiver backend, you can also think about registering the RFC destinations in SMQS in PI as known from standard ALE tuning. By changing the Max. Conn. or the Max. Runtime values in SMQS you can limit or increase the number of DIA WPs used by the tRFC layer to send the IDocs.
4.7 Message Prioritization on the ABAP Stack

A performance problem is often caused by an overload of the system, for example, due to a high volume master interface. If business critical interfaces with a maximum response time are running at the same time then they might face delays due to a lack of resources. Furthermore, as described in qRFC Queues (SMQ2), messages of different interfaces use the same inbound queue by default which means that critical and less-critical messages are mixed up in the same queue.

We highly recommend using message prioritization on the ABAP stack to prevent such delays for critical interfaces (for Java, see Prioritization in the Messaging System).

For more information about PI prioritization, see http://help.sap.com and navigate to SAP NetWeaver → SAP NetWeaver PI/Mobile/IdM 7.1 → SAP NetWeaver Process Integration 7.1 Including Enhancement Package 1 → SAP NetWeaver Process Integration Library → Function-Oriented View → Process Integration → Integration Engine → Prioritized Message Processing.

To balance the available resources further between the configured interfaces, you can also configure a different parallelization level for queues with different priorities. Details for this can be found in SAP Note 1333028 – Different Parallelization for HP Queues.
5 Analyzing the Business Process Engine (BPE)

This chapter helps to analyze performance problems if chapter Determining the Bottleneck has shown that the BPE is the slowest step for a message during its time in the PI system. Of course, this type of runtime engine is very susceptible to inefficient design. Information about best practices for designing BPE processes can be found in the documentation Making Correct Use of Integration Processes. In general, the memory and resource consumption is higher than for a simple pipeline processing in the Integration Engine. As outlined in the document linked above, every message that is sent to BPE and every message that is sent from BPE is duplicated. In addition, work items are created for the integration process itself as well as for every step. More database space is therefore required than previously expected and more CPU time is needed for the additional steps.

5.1 Work Process Overview

The Business Process Engine executes the steps (work items) of an integration process using tRFC calls to the RFC destination WORKFLOW_LOCAL_<client>. To check if there are enough resources available to process the work items, call transaction SM50 (on each application server) while one of the integration processes is running.

- Look for the user “WF-BATCH” and follow its activities by refreshing the transaction
- Are there always dialog work processes available?

The most important point to keep in mind here is the concurrent activity of the Business Process Engine (for ccBPM processes) and the Integration Engine (for pipeline processing). Both runtime environments need dialog work processes. Thus, performance problems in one of the engines cannot be solved by restricting the other. Rather, an appropriate balance between these two has to be found.

The activity of the WF-BATCH user is not restricted by default. It is highly recommended that you register the RFC destination WORKFLOW_LOCAL_<client> with transaction SMQS and assign a suitable amount of maximum connections. Otherwise, ccBPM activity may block the pipeline processing of the Integration Server and reduce the throughput of PI drastically. SAP Note 888279 - Regulating/distributing the Workflow Load gives more details.

In addition, it might help to use specific servers (dialog instances) to carry out a given workflow or to use load balancing. Of course, both options only apply for larger PI systems that consist of at least a central instance and one dialog instance. SAP Note 888279 explains how to set this up.

5.2 Duration of Integration Process Steps

Check in the workflow log for long-running steps as described in the previous chapter. Use the “List with Technical Details” to do so. Since the design of integration processes varies significantly, there is no general rule for the duration of a step or for specific sequences of steps. Instead, you have to get a feeling for your implemented integration processes.

- Did a specific process step decrease in performance over a period of time?
- Does one specific process step stick out with regard to the other steps of the same integration process?
- Do you observe long durations for a transformation step (“mapping”)?
- Do you observe long durations of synchronous send/receive steps in the integration process which correlates, for example, to a slow response from a connected remote system?

Use the SAP Notes database to learn about recommendations and hints. SAP Note 857530 - BPE: Composite Note Regarding Performance acts as a central note for all performance notes and might be used as the entry point.
Once you are able to answer the above questions, there are several paths to follow:

- Two common reasons should be taken into consideration for a performance decrease over time: Firstly, the load on PI might have increased over time as well. Check the hardware resources (chapter 0) and carefully monitor the work process availability (chapter 5.1). Alternatively, the underlying databases might slow down the integration processes. Use chapter 5.4 to check this option.

- If it is a specific process step that sticks out, the process design itself has to be reviewed. The SAP Notes mentioned above might help here.

- If the mapping step takes a long time, it is worth having a look at chapter 4.4.2 since the BPE and the IE use the same resources (mapping connections from the ABAP gateway to the J2EE server) to execute their mapping.

- If there is a long processing time for a synchronous send/receive step, check the connected backend system for any performance issues.

### 5.3 Advanced Analysis: Load Created by the Business Process Engine (ST03N)

Since the Business Process Engine runs on the ABAP part of the PI Integration Server, you can use the statistical data written by the dialog work processes to analyze a performance problem.

**Procedure**

Log in to your Integration Server and call transaction ST03N. Switch to “Expert Mode” and choose the appropriate timeframe (this could be the “Last Minute’s Load” from the Detailed Analysis menu or a specific day or week from the Workload menu). In the Analysis View, navigate to User and Settlement Statistics and then to User Profile. The user you are interested in is the WF-BATCH user who does all ccBPM-related work.

![Last Minutes Load on ncev00000012a_XIN_00](image)

- Compare the total CPU Time (in seconds) to the time frame (in seconds) of the analysis. In the above example, an analysis time frame of 1 hour has been chosen which corresponds to 3600 seconds. To be exact, these 3600 seconds have to be multiplied by the number of available CPUs. Out of these 3600*#CPU seconds, the WF-BATCH user used up 36 seconds.

- The above value gives you a good idea of how much load the Business Process Engine creates on your PI Integration Server. By comparison with the other users (as well as the CPU utilization from transaction ST06), you can determine if the Integration Server is able to handle this load with the available number of CPUs or not. This does not help you to optimize the load, but it does provide support when
deciding how to distribute the workload of the BPE and the workload of the Integration Server (see chapter 5.1).

✓ Experienced ABAP administrators should also analyze the database time, wait time, and compare these values.

5.4 Database Reorganization

The database is accessed many times during the processing of an Integration Process. This is not usually connected with performance problems, but if a specific database table is large then statements may take longer than for small database tables.

✓ Use transaction ST05 to collect a SQL trace while the integration process in question is being executed. Restrict the trace to user WF-BATCH. When the integration process is finished, stop the trace and view it. Look for high values in the duration column, especially for SELECT statements.

✓ Use transaction SE16 to determine the number of entries for the typical workflow tables: SWW_WI2OBJ and SWFRXI*. There are many more database tables used by BPE, but if the number of entries is high for the above tables, they are high for the rest as well. See SAP Note 872388 - Troubleshooting Archiving and Deletion in XI and the links given in this note if you find high amounts of entries.

5.5 Queuing and ccBPM (or: Increasing Parallelism for ccBPM Processes)

Until recently, ccBPM processes accepted messages from only one queue (one queue XBQO$PEWS* for each workflow). If there was a high message throughput for this workflow then a high backlog occurred for these queues. A couple of enhancements have therefore been implemented to allow an increase to the throughput for ccBPM processes:

- Inbound processing without buffering
- Use of one configurable queue that can be prioritized or assigned to a dedicated server
- Use of multiple inbound queues (queue name will be XBPE_WS*)
- Configure transactional behavior of ccBPM process steps to reduce number of persistency steps

Details about the configuration and tuning of the ccBPM runtime are described at: https://www.sdn.sap.com/irj/sdn/howtougides → SAP Netweaver 7.0 → End-to-End Process Integration:

- How to Configure Inbound Processing in ccBPM Part I: Delivery Mode
- How to Configure Inbound Processing in ccBPM Part II: Queue Assignment
- How to Configure ccBPM Runtime Part III: Transactional Behavior of an Integration Process

Procedure

✓ Use transaction SMQ2 to check if the queues of type XBQO$PE* show a high backlog.

- Based on this information, verify if the ccBPM process is suitable to run with multiple queues. This is especially the case for collect patterns realized in BPE. If you can use multiple queues, configure the workflow accordingly.

5.6 Message Packaging in BPE

Inbound processing takes up the most amount of processing time in many scenarios within BPE.
Message packaging in BPE helps improve performance by delivering multiple messages to BPE process instances in one transaction. This can lead to an increased throughput, which means that the number of messages that can be processed in a specific amount of time can increase significantly. Message packaging can also increase the runtime for individual messages (latency) due to the delay in the packaging process. The sending of packages can be triggered when the packages exceed a certain number of messages, a specific package size (in kB), or a maximum waiting time. The extent of the performance improvement that can be obtained depends on the type of scenario. Scenarios with the following prerequisites are generally most suitable:

- Many messages are received for each process instance
- Messages that are sent to a process instance arrive together in a short period of time
- Generally high load on the process type
- Messages do not have to be delivered immediately

For example, "collect scenarios" are particularly suitable for message packaging. The higher the number of messages is in a package, the higher the potential performance improvements will be.

Tests that have been performed have shown a high potential for throughput improvements up to factor 4.7 in tested Collect Scenarios depending on the packaging size that has been configured. For more details about the functionality of Message Packaging in BPE and its configuration, see SAP Note 1058623 - BPE: Message Packaging.

Message packaging in BPE is independent of message packaging in PI (see chapter 4.5), but they can be used together.
6 Analyzing the Adapter Framework (AFW)

This chapter describes further checks and possible reasons for bottlenecks in the Adapter Framework. Although this is not the place to describe the AFW architecture in detail, the following aspects are important to know for the analysis of a performance problem on the AFW:

- The AFW contains the Messaging System that is responsible for the persistence and queuing of messages. Within the flow of a message, it is placed in between the sender adapter (for example, a file adapter that polls from a folder) and the Integration Server as well as between the Integration Server and the receiver adapter (for example, a JMS adapter that sends a message out to an external system).

- In SAP NetWeaver PI 7.1, the Messaging System uses 4 queues for each adapter type and these are responsible for receiving and sending messages. To separate the message transfer, each adapter (for example, JMS, SOAP, JDBC, and so on) has its own set of queues. One queue for receiving synchronous messages (request queue), one queue for sending synchronous messages (call queue), one queue for receiving asynchronous messages (receive queue), and one queue for sending asynchronous messages (send queue). The name of the queue always states the adapter type and the queue name (for example, JMS_http://sap.com/xi/XI/SystemReceive for the JMS receiver asynchronous queues).

- A configurable number of consumer threads are assigned on each queue. These threads work on the queue in parallel, meaning that the Messaging Queues are not strictly FirstIn-FirstOut. Five threads are assigned by default to each queue and thus five messages can be processed in parallel. But, as will be discussed later, the parallel execution of requests to the backend depends heavily on the adapter being used since not every adapter can work in parallel by default.

- SAP NetWeaver PI 7.1 introduced a new queue: The dispatcher queue. The dispatcher queue is used to realize prioritization on the AAE (see chapter Prioritization in the Messaging System). Every message in the PI system is first assigned to the dispatcher queue before it is assigned to the adapter-specific queues.

Viewed from the perspective of a message that enters the PI system using a J2EE Engine adapter (for example, File) and leaves the PI system using a J2EE Engine adapter (for example, JDBC) asynchronously, the steps are as follows:

1) Enter the sender adapter
2) Put into the dispatcher queue of the messaging system
3) Forwarded to the File send queue of the messaging system
4) Retrieved from the send queue by a consumer thread
5) Sent from the messaging system to the pipeline of the ABAP Integration Engine (if no Integrated Configuration (AAE) is used
6) Processed by the pipeline steps of the Integration Engine
7) Sent to the messaging system by the Integration Engine
8) Put into the dispatcher queue of the messaging system
9) Put into the JDBC receive queue of the messaging system
10) Retrieved from the receive queue
11) Sent to the receiver adapter
12) Sent to the final receiver.
6.1 Adapter Performance Problem

Compared to a bottleneck in the Messaging System it is relatively easy to detect a performance problem/bottleneck in the adapter. Since the timestamps for inbound and outbound messages are written at different points in time, the procedure is described separately for sender and receiver adapters.

Note: It is not generally possible for all sender and receiver adapters to work in parallel (discussed in section Adapter Parallelism).

6.1.1 Adapter Parallelism

As mentioned before, not all adapters are able to handle requests in parallel. Thus, increasing the number of threads working on a queue in the messaging system will not solve a performance problem/bottleneck. There are 3 strategies to work around these restrictions:

1) Create additional communication channels with a different name and adjust the respective sender/receiver agreements to use them in parallel.

2) Add a second server node that will automatically have the same adapters and communication channel running as the first server node. This does not work for polling sender adapters (File, JDBC, or Mail) since the adapter framework scheduler assigns only one server node to a polling communication channel.

3) Install and use a non-central Adapter Framework for performance-critical interfaces to achieve better separation of interfaces.

Some of the most frequently-used adapters and the possible options are discussed below:

- Polling Adapters (JDBC, Mail, File):
  
  At the sender side these adapters use an Adapter Framework Scheduler which assigns a server node that does the polling at the specified interval. Thus, only one server node does the polling and no parallelization can be achieved. For example, since the channels are doing the same SELECT statement on the database or pick up files with the same file name, parallel processing will only result in locking problems. To increase the throughput for such interfaces, the polling interval has to be reduced to avoid a big backlog of data that is to be polled. If the volume is still too high, you should think about creating a second interface, for example, the new interface would poll the data from a different directory or database table to avoid locking.

  At the receiver side, the adapters work sequentially on each server node by default. For example, only one UPDATE statement can be executed for JDBC for each Communication Channel (independent of the number of consumer threads configured in the Messaging System) and all the other messages for the same Communication Channel will wait until the first one is finished. This is done to avoid blocking situations on the remote database. On the other hand, this can cause blocking situations for whole adapters as discussed in section Avoid Blocking Caused by Single Slow/Hanging Receiver Interfaces.

  To allow better throughput for these adapters, SAP NetWeaver PI 7.1 allows you to configure the degree of parallelism at the receiver side. In the Processing tab of the Communication Channel in the field “Maximum Concurrency”, enter the number of messages to be processed in parallel by the receiver channel. For example, if you enter the value 2, then two messages are processed in parallel on one J2EE server node. The parallel execution of these statements at database level of course depends on the nature of the statements and the isolation level defined on the database. If all statements update the same database record then no parallelization can be achieved.

- JMS Adapter:
The JMS adapter processes one request after the other. However, it is possible to use multiple JMS adapters at the same time, either on one J2EE server node by duplicating communication channels, or on several J2EE server nodes.

- **SOAP Adapter:**
  
  The SOAP adapter is able to process requests in parallel. The amount depends on the parallelism of the threads defined in the messaging system and (in case of the receiver adapter) the ability of the receiving system to cope with the load. Note that it is often also possible to send SOAP messages directly to the Integration Server pipeline instead of the SOAP sender adapter.

- **RFC Adapter:**
  
  The RFC adapter offers parameters to adjust the degree of parallelism by defining the number of initial and maximum connections to be used.

The table below gives a summary of the possible degree of parallelism for the different adapter types.

<table>
<thead>
<tr>
<th>Adapter Type</th>
<th>Sender</th>
<th>Receiver</th>
<th>Parallelism with 2 channels?</th>
<th>Parallelism with n&gt;1 server nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>Sequential</td>
<td>Configurable (per default sequential)</td>
<td>Yes (but not recommended for sender due to file locking issues). On Receiver side MaxConcurrency should be increased.</td>
<td>Sender: No Receiver: Yes</td>
</tr>
<tr>
<td>JDBC</td>
<td>Sequential</td>
<td>Configurable (per default sequential)</td>
<td>Yes (but not recommended for sender SELECT and not for UPDATE on receiver side only INSERT where no locking occurs). On Receiver side MaxConcurrency should be increased.</td>
<td>Sender: No Receiver: Yes</td>
</tr>
<tr>
<td>Mail</td>
<td>Sequential</td>
<td>Parallel</td>
<td>Not recommended / reasonable</td>
<td>Sender: No Receiver: Yes</td>
</tr>
<tr>
<td>JMS</td>
<td>Parallel</td>
<td>Parallel</td>
<td>Not reasonable since already parallel.</td>
<td>Yes</td>
</tr>
<tr>
<td>SOAP</td>
<td>Parallel</td>
<td>Parallel</td>
<td>Not reasonable since already parallel.</td>
<td>Yes</td>
</tr>
<tr>
<td>RFC</td>
<td>Parallel</td>
<td>Parallel</td>
<td>Not reasonable since already parallel.</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### 6.1.2 Sender Adapter

The first step(s) to be taken for a sender adapter depend on the type of adapter itself. Afterwards, however, the message flow is always the same, that is to put the message into the queue of the Messaging System. If the message is synchronous then it is the call queue, if the message is asynchronous that it is the send queue. The final action of the Adapter Framework is then to send the message from the messaging system to the Integration Server. Only the first entries of the audit log are of interest to establish if the sender adapter has a performance problem: From the first entry to the entry “Message successfully put into the queue”. These steps logically belong to the sender adapter while the subsequent steps belong to the Messaging System. The sender adapter uses one J2EE Engine application thread to execute its work. This is generally the same thread for all steps involved.

- Select the “Details” of a message in the AFW. The first entry of the audit log will be the beginning of the activity of the sender adapter. The end of the adapter’s activity is marked by the entry “The application sent the message asynchronously using connection *”. Returning to application”.

```
Note: With SAP NetWeaver PI 7.1 the audit log is not persisted for successful messages in the database by default to avoid performance overhead. Therefore, the audit log is only available in the cache for a limited period of time (based on the overall message volume).

Audit log persistence can be activated temporarily for performance troubleshooting where audit log information is required. To do so, use the NWA and change the parameter “messaging.auditLog.memoryCache” to false in service XPI Service: Messaging System by setting the parameter to false. Only do so temporarily during the time of troubleshooting to avoid any performance problems from the additional persistence.

- Depending on the type of the adapter, you now have to investigate why the sender adapter needs a high amount of processing time. This could be a complicated operating system command, a bad network connection, or a slow backend system among many other reasons.
- If you have defined your own non-standard module to be executed by the sender adapter, this needs to be analyzed as discussed in Performance of Module Processing.

### 6.1.3 Receiver Adapter

For a receiver adapter it is just the other way around when compared to the sender adapter. First the message is received from the Integration Server and then put into a queue of the Messaging System, this time either the request or the receive queue. The last step is then the activity of the receiver adapter itself.

- Select the “Details” of a message in the AFW. The adapter activity starts after the entry “The message status set to DLNG.” The message will then be forwarded to the entry point of the AFW “Delivering to channel: <channel_name>”, enters the Module Processor “MP: Entering Module Processor” and ends with “The message was successfully delivered to the application using connection **”. 

[Image: Message Display Tool (Detail Display) - Mozilla Firefox]
Depending on the type of the adapter, you now have to investigate why the receiver adapter needs a high amount of processing time. This could be a complicated operating system command, a bad network connection, or a slow backend system among many other reasons.

Here, too, custom modules can be the reason for a prolonged processing time. Check **Performance of Module Processing** for details.

### 6.2 Messaging System Bottleneck

As described earlier, the Messaging System is responsible for storing and queuing messages that come from any sender adapter and go to the Integration Server pipeline, or that come from the Integration Server pipeline and go to any receiver adapter. The time difference between receiving the message from the one side and delivering it to the other side is the value that can be used to analyse bottlenecks in the Messaging System. The following chapters describe how to determine the time difference for both directions.

A bottleneck (or rather a “fake” bottleneck) in the Messaging System can be closely connected with a performance problem in the receiver adapter. You must therefore make sure you execute the check described in chapter 6.1.3. If the receiver adapter needs a lot of time to process messages, then of course the messages get queued in the messaging system and remain there for a long time since the adapter is not ready to process the next one yet. It looks like the messaging system is not fast enough, but actually the receiver adapter is the limiting factor.

- Execute the checks described in chapter 6.2.1 and **6.2.2**. To get an overview of the situation, you should do this for multiple messages.
Decide if a specific receiver adapter is causing long waiting times in the messaging system by executing the check described in chapter 6.1.3. An additional hint at a receiver adapter problem is if only messages to specific receivers show a long wait time in the queue.

Use the queue monitoring in the Runtime Workbench (RWB) to analyze how many threads are active for each of the queue types and how large the queue size currently is. To do so, choose Component Monitoring $\rightarrow$ Adapter Engine. Press the Engine Status button and choose tab “Additional Data”. This will open a page similar to the one below showing the number of messages in a queue, the threads currently working, and the maximum available. Note: This view only shows the Messaging System of one J2EE server node. To get the whole picture you have to check all the server nodes that are available in your system.

Since checking all the server nodes with this browser frontend can be very tedious, it is better to use Wily Introscope since it allows easy graphical analysis of the queues and thread usage of the messaging system.

The starting point in Wily Introscope is usually the PI triage showing all the backlogs in the queue. In the screenshot below we can see that we faced a backlog in the dispatcher queue and in the PI inbound queues.
The dispatcher queue is not the problem here since it will forward messages to the adapter queue if any free threads are available. The analysis must start in the PI inbound queues. By using the navigation button in the upper right corner of the inbound queue size we can directly jump to a more detailed view where we can see that the file adapter was causing the backlog:

To see the consumer thread usage, we can then follow the link to the file adapter. In the screenshot below we can see that all consumer threads were exceeded during the time of the backlog. Thus, increasing the number of consumer threads could be a solution if the resulting delay is not acceptable for business:
If it is obvious that the messaging system does not have enough resources, increase the number of consumers for the queue in question. Use the results of the previous check to decide which queues need more consumers.

The adapter specific queues in the messaging system have to be configured in the NWA using service “XPI Service: AF Core” and property “messaging.connectionDefinition”. The default values for the sending and receiving consumer threads are set as follows:

```
{name=global, messageListener=localejbs/AFWListener, exceptionListener=localejbs/AFWListener,, pollInterval=60000, pollAttempts=60, Send.maxConsumers=5, Recv.maxConsumers=5, Call.maxConsumers=5, Rqst.maxConsumers=5)
```

To set individual values for a specific adapter type, you have to add a new property set to the default set with the name of the respective adapter type, for example:

```
{name=JMS_http://sap.com/xi/XI/System, messageListener=localejbs/AFWListener, exceptionListener=localejbs/AFWListener, pollInterval=60000, pollAttempts=60, Send.maxConsumers=7, Recv.maxConsumers=7, Call.maxConsumers=7, Rqst.maxConsumers=7)
```

The name of the adapter specific queue is based on the pattern `<adapter_name>_<namespace>`, for example RFC_http://sap.com/xi/XI/System.

Note that you must not change parameters such as pollInterval and pollAttempts. For more details, see SAP Note 791655 - Documentation of the XI Messaging System Service Properties.

Not all adapters use the parameter above. Some special adapters like CIDX, RNIF, or Java Proxy can be changed by using the service “XPI Service: Messaging System” and property messaging.connectionParams by adding the relevant lines for the adapter in question as described above.

**Important**: Make sure that your system is able to handle the additional threads, that is, monitor the CPU usage and thread availability (system threads as well as application threads) as well as the memory of the J2EE Engine after you have applied the change.

**Important**: Keep in mind that not all adapters are able to handle requests in parallel, even if you provide more queues to send messages to them. See chapter Adapter Parallelism for details.
A backlog of messages in the messaging system is highly critical for synchronous scenarios due to timeouts that might occur. Therefore, the number of synchronous threads always have to be tuned so that no bottleneck occurs.

6.2.1 Messaging System in Between AFW Sender Adapter and Integration Server

Open the “Details” of a message and look for the following entry: “The message was successfully retrieved from the send queue”. Note that the queue name would be “Call Queue” if the message was synchronous. To determine how long the message waited to be picked up from the queue, compare the timestamp of the above entry with the timestamp for the step “Message successfully put into the queue”. A large time difference between those two timestamps would therefore indicate a bottleneck for consumer threads on the sender queues in the Messaging System.

Instead of checking the latency of single messages, we recommend using Wily Introscope to monitor the queue behaviour as shown above.

Asynchronous messages only use one thread for processing a message in the messaging system. Multiple threads are used for synchronous messages. The adapter thread puts the message into the messaging system queue and will wait till the messaging system delivers the response. The thread is therefore not available for other tasks until the response is returned. A consumer thread on the call queue sends the message to the Integration Engine. The response will be received by a third thread (consumer thread of send queue) which correlates the response with the original request. After this, the initiating adapter thread will be notified to send the response to the original sender system. This correlation can be seen in the audit log of the synchronous message below.
6.2.2 Messaging System in Between Integration Server and AFW Receiver Adapter

Open the “Details” of a message and look for the following entry: “Message successfully put into the queue”. Compare this timestamp with the timestamp for the step “The message was successfully retrieved from the receive queue”. The time difference between these two timestamps is the time that the message waited in the messaging system. A large time difference between those two timestamps herefore indicates a bottleneck in the Messaging System.

6.2.3 Prioritization in the Messaging System

SAP NetWeaver PI 7.1 offers the possibility for prioritization of messages similar to the ABAP stack. This new feature is especially helpful if high volume interfaces run at the same time with business critical interfaces. To minimize the delay to critical messages, the Messaging System now makes it possible for you to define high, medium, and low priority processing at interface level. Based on the priority, the Dispatcher Queue of the messaging system (which will be the first entry point for all messages) will forward the messages to the standard adapter-specific queues. The priority assigned to an interface determines the number of messages that are forwarded once the adapter-specific queues have free consumer threads available. This is done based on a weighting of the messages to be reloaded. The weights for the different priorities are as follows: High: 75, Medium: 20, Low: 5.

Based on this approach you can ensure that more resources are available for high priority interfaces.

You can find more details on configuring usage prioritization within the AAE at: http://help.sap.com, and navigate to SAP NetWeaver → SAP NetWeaver PI/Mobile/IdM 7.1 → SAP NetWeaver Process Integration 7.1 Including Enhancement Package 1 → SAP NetWeaver Process Integration Library → Function-Oriented View → Process Integration → Process Integration Monitoring → Component Monitoring → Prioritizing the Processing of Messages.

6.2.4 Avoid Blocking Caused by Single Slow/Hanging Receiver Interface

As already discussed above, some receiver adapters process messages sequentially on one server node by default. This is independent of the number of consumer threads defined for
the corresponding receiver queue in the messaging system. For example, even though you have configured 20 threads for the JDBC Receiver, one Communication Channel will only be able to send one request to the remote database at a given time. If there are many messages for the same interface, all of them will get a thread from the messaging system but will be blocked when performing the adapter call.

In the case of a slow message transmission, for example, for EDI interfaces using ISDN technology or remote system with small network bandwidth, this behavior can cause a "hanging" situation for a specific adapter since one interface can block all messaging threads. As a result, all the other interfaces will not get any resources and will be blocked.

Based on this, a new parameter `messaging.system.queueParallelism.maxReceivers` was introduced which can be set in NWA in service XPI Service: Messaging System. With this parameter you can restrict the maximum number of receiver consumer threads that can be used by one interface. Note: This is a global parameter at the moment that affects all adapters. It should therefore not be set too restrictively. We recommend that you set it to 5 (so that each interface can use 5 consumer threads for each server node) and increase the overall number of threads on the receive queue for the adapter in question (for example, JDBC) to 20. With this configuration it will be possible for four interfaces to get resources in parallel.

For more information, see SAP Note 1136790 - Blocking Receiver Channel May Affect the Whole Adapter Type.

6.3 Performance of Module Processing

If your analysis in chapter Adapter Performance Problem pointed to a problem in the Module Processing, then you must analyze the runtime of the modules used in the Communication Channel. Adapter modules can be combined so that one CC calls multiple modules in a defined sequence. Adapter modules can be custom-developed or SAP-standard and can be used for many different purposes – like to transform the EDI message before sending it to the partner or to change header attributes of a PI message before forwarding it to a legacy application. Due to the flexible usage of adapter modules there is also no standard way to approach performance problems.

In the audit log shown below you can see two adapter modules. One module is a customer-developed module called `SimpleWaitModule`. The next module called `CallSAPAdapter`, which is a standard module that inserts the data into the messaging system queues.

In the audit log you will get a first impression of the duration of the module. In the example above you can see that the `SimpleWaitModule` requires 4 seconds of processing time.

Once again, Wily Introscope offers a better overview of the processing time of modules. In the screenshot below you can see a dashboard showing the cumulative/average response times and number of invocations of different modules. If there is one that has been running for a very long time, then it would be very easy to identify since there will be a line indicating a much higher average response time. The tooltip displays the name of the module.
Once you have identified the module with the long running time you have to talk to the responsible developer to understand why it is taking that long. Eventually the module executes a look-up to a remote system using JCo or JDBC which could be responsible for the delay. In the best case the module will print out information in addition to the audit log to detect such steps. If not, use the Wily Introscope transaction trace as explained in appendix Wily Transaction Trace.

6.4 Advanced Adapter Engine/Integrated Configuration

As stated earlier, SAP NetWeaver PI 7.1 allows you to configure scenarios that only run on the Java stack using the Advance Adapter Engine (AAE) when only Java-based adapters are used. A new configuration object is used for this that is called Integrated Configuration. When using this, the steps executed so far in the ABAP pipeline (Receiver Determination, Interface Determination and Mapping) are also executed by the services in the Adapter Engine.

The major advantage of AAE processing is a reduced overhead due to the context switches between ABAP and Java. Thus, the overall throughput can be increased significantly and the overall latency of a PI message can be reduced greatly. If possible, interfaces should always use the Integrated Configuration to achieve best performance. Therefore, the best tuning option is to change an ABAP-based scenario to an AAE-based one.

The following describes the steps used in Integrated Configuration to help you better understand the message flow of an interface. The example is JMS to Mail:

1) Enter the JMS sender adapter
2) Put into the dispatcher queue of the messaging system
3) Forwarded to the JMS send queue of the messaging system
4) Message is taken by JMS send consumer thread:
   a. No message split used:
      In this case, the JMS consumer thread will do all the necessary steps previously done in the ABAP pipeline (like receiver determination, interface determination, and mapping) and will then also transfer the message to the backend system (remote database, in our example). Thus, all the steps are executed by one thread only. Retrieved from the send queue by a consumer thread
   b. Message split used (1:n message relation):

In this case there is a context switch. The thread taking the message out of the queue will process the message up to the message split step. It will create the new message and put it in the Send queue again from where it will be taken by different threads (which will then map the child message and finally send it to the receiving system).

As we can see in this example for Integrated Configuration, only one thread does all the different steps of a message. The consumer thread will not be available for other messages during the execution of these steps. The tuning of the Send queue (call for synchronous messages) is therefore much more important for scenarios using Integrated Configuration than for ABAP-based scenarios.

The different steps of the message processing can be seen in the audit log of a message. If, for example, a long-running mapping or adapter module is indicated, then you can use the relevant chapter of this guide. There is no difference in the analysis except that for mappings no JCo connection is required since the mapping call is done directly from the Java stack.

### 6.5 J2EE Engine Bottleneck

All tuning that can be done in the AFW is limited by the ability of the J2EE Engine to handle a given amount of threads and to provide enough memory that is needed to process all requests. Of course, the CPU is also a limiting factor, but this will be discussed in Chapter 0.

From SAP NetWeaver PI 7.1 onwards, the SAP VM is used on all platforms. Therefore the analysis of all platforms can use the same tools.

#### 6.5.1 Java Memory

The first analysis of the J2EE memory is usually done using the garbage collection (GC) behaviour of the Java Virtual Machine (JVM). To get the information of the GC written to the standard log file, the JVM has to be started with the parameter –verbose:gc.

1. Analyze the file std_server<n>.out for the frequency of GCs, their duration, and the allocated memory.
2. Look for unusual patterns indicating an Out-of-Memory error or an unintentional restart of your J2EE engine. This would be visible if the allocated memory drops to 0 at a given point in time. A healthy Garbage Collection output would display a saw tooth pattern – that is, the memory usage would increase over time but then go down to its original value as soon as a full Garbage Collection is triggered. You should see the memory usage go down to initial values during low volume times (night-time).
3. Pay attention to GCs with a high duration. This is important because no PI message can be mapped or processed by the J2EE Adapter Framework during a GC in a JAVA VM. One of the many reasons for long GCs that should be mentioned here is paging (see chapter 9.2). If the heap space of the J2EE Engine is exhausted, all objects in the heap are evaluated for their references. If there is still a reference, the object is kept. If there is no reference, the object is deleted. If some or all of the objects have been swapped out due to insufficient RAM, then they have to be swapped in to be evaluated. This leads to very long runtimes for GCs. GCs of more than 15 minutes have been observed on swapping systems.
4. Different tools exist for the Garbage Collection Analysis:
   1. **Solution Manager Diagnostics:**

      Solution Manager Diagnostics (SMD) offers a memory analysis application that is able to read the GC output in the std_server<n>.out files. To start, call the Root Cause Analysis section. In the Common Task list you find the entry Java Memory Analysis which will give you the chance to upload your std_server<n>.out file. It shows the memory allocated after the GC and also prints the duration of each GC (black dots with duration scale on right axis).
A normal GC output should show a saw tooth pattern where the memory always goes down to an initial value (especially during low volume times). This is shown in the example screenshot below.

2) Wily Introscope:

Again, Wily Introscope offers different dashboards that can be used to check the Garbage Collection behavior on the J2EE engine. The dashboard can be found using the J2EE Overview → J2EE GC Overview and shows important KPIs for the GC, such as the count of GC or the duration for each interval. The %GC Time dashboards shows the ratio of time spent in GC. One example is shown in the screenshot below.
3) *Netweaver Administrator:*

The NWA also offers a view to monitor the Java heap usage. However, the important information about the duration of the GC is not available. This monitor can be found by navigating to *Availability and Performance Management ➔ Java System Reports ➔ Choose Report System Health.*

- Search for restarts of the J2EE Engine by searching for the string “is starting” in the file std_server<n>.out. Apart from the first of these entries (which simply marks the start of the J2EE Engine), the second and any later entries mark a fatal situation after which the J2EE server node had to be restarted. This could, for example, be an out-of-memory situation. Search the log above those entries for a possible reason.

- Search for exceptions and fatal entries. Was an out-of-memory error reported (search for pattern *OutOfMemory*)? Was a thread shortage reported?

- This document does not deal with tuning the J2EE Engine. If at this point it becomes clear that the J2EE Engine is the limiting factor, proceed with SAP Notes or open a customer message. Only some very basic recommendations are given here.

- The parameters for the SAP VM are defined by Zero Admin templates. From time to time new Zero Admin templates might be released which change important parameters for the SAP VM. Thus, you should check [SAP Note 1248926 - AS Java](#).
VM Parameters for NetWeaver 7.1-based Products regularly and apply the changes accordingly.

- If it is already obvious that the memory of your J2EE Engine has been exceeded then you have several options for scaling your PI landscape:
  1. Increase the Java Heap of your server node: If sufficient physical memory is available, increase the default heap size of 2 GB to a higher value such as 3 or 4 GB. Experience shows that the SAP VM can handle these heap sizes without major impact on performance. Larger heap sizes can cause longer processing time of GCs, which in turn can affect the PI application negatively. Increasing the maximum heap size on the J2EE engine will automatically adapt the new area of the heap due to a dynamic configuration (by default 1/6 of the overall heap). After increasing the heap size, monitor the duration of the GC execution.
  2. Add an additional server node to distribute the load over more processes. SAP recommends that productive systems have at least 2 Java server nodes. Prerequisite is that your physical host has enough RAM and CPU.
  3. Add an additional application server consisting of a double stack Web Application Server (ABAP and J2EE Engine) to distribute the load to an additional hardware. Find details on the scaling of PI with multiple instances in the Guide How To Scale SAP PI 7.1.
  4. Use a decentral Adapter Engine to separate large messages that cause the memory problem from business critical scenarios.

6.5.2 Java System and Application Threads

A system or application thread is required for every task that the J2EE Engine executes. Each of these thread types are maintained using a thread pool. If the pool of available threads is expired, no more requests can be processed. It is therefore important that you have enough threads available at all times.

- In general, SAP recommends that you increase the system and application thread count for a PI system. As described in SAP Note 937159 - XI Adapter Engine is Stuck, the application threads should be increased to 350 and the system threads to 120 for all J2EE server nodes.
- There are two options for checking the thread usage:
  1) Wily Introscope:

Wily Introscope provides a dashboard in the J2EE Overview section called J2EE CPU and Memory Detail that can also be used to monitor the thread usage on a PI engine. Different views exist for Application and System Threads as shown in the screenshot below.
2) NWA:

The NWA also offers a monitor similar to the one available for the memory mentioned above. This monitor can be found by navigating to Availability and Performance Management → Java System Reports → Choose Report System Health and looking at the windows shown in the screenshot below.

- Check if the ThreadPoolUsageRate is below 80% for both the application threads and the system threads. Also check the maximum value by choosing a longer history in Wily Introscope.
- Has the thread usage increased over the last days/weeks?
- Is there one server node that shows a higher thread usage than others?
- In case of high load and performance problems, generate a Thread Dump and analyze the threads for possible reasons for the performance problems. Thread Dumps can be created using the SAP management console (http://<server>:5<Instanc_number>13 → AS Java → Process Table)
An additional type of thread was introduced with SAP NetWeaver PI 7.1 that is responsible for HTTP traffic: FCA Server Threads. The FCA Server Threads are responsible for receiving HTTP calls at the Java side (after the Java dispatcher is no longer available). Also, FCA Threads use a Thread Pool. Fifteen FCA Threads are configured by default but based on SAP Note 1375656 - SAP Netweaver PI System Parameters we recommend that you increase it to 50. This can be done in the NWA by changing the parameter FCAServerThreadCount of service HTTP Provider.

FCA Server Threads are particularly crucial for synchronous message transfer and for HTTP-based scenarios like WebServices or calls from the ABAP engine to the Adapter Engine. If there is a long duration of the HTTP call due to a slow backend system then the thread is blocked for the whole time and not available to send other HTTP requests (other PI messages).

There is currently no standard monitor available for FCA Threads except thread dumps. A new dashboard will be integrated into Wily Introscope soon and will also show the FCA Server Threads that are in use.

### 6.5.4 Switch Off VMC

The Virtual Machine Container (VMC) is enabled by default for an ABAP WebAS 7.1. Since PI does not use VMC at runtime, the VMC can be switched off on a PI system to avoid resource overhead. This recommendation is based on SAP Note 1375656 - SAP Netweaver PI System Parameters.

You can activate the VM Container setting the profile parameters Vmcl/enable = off which can be changed using profile maintenance (transaction RZ10). Once you have made the change, you need to restart the SAP Web Application Server instance.
ABAP Proxy System Tuning

Every ABAP WebAS > 6.20 includes an ABAP Proxy runtime. This enables a SAP system to talk native PI protocol (XI-SOAP) so that no costly transformation is necessary.

In general, ABAP proxy is just a framework that is triggered by (sender) or triggers (receiver) application-specific coding. It is not possible to give a general tuning recommendation because the applications and use cases of ABAP proxy can differ so greatly.

In this section we would like to highlight the system tuning options that can be applied to improve the throughput at the ABAP Proxy backend side.

In general you can increase the throughput of EO interfaces by changing the queue parallelization. Two different parameters exist to change the number of qRFC queues on the ABAP Proxy backend. As in PI, ABAP Proxy processing is based on qRFC inbound queues (SMQ2) with specific names. A sender proxy uses XBTS* queues (10 parallel by default) and a receiver proxy XBTR* (20 parallel by default).

The sender proxy only forwards the message to PI by executing an HTTP call that must not take too much time and which is not resource-critical. On the other hand, the receiver proxy executes the inbound processing of the messages based on the application context (and which can be very time-consuming). It is therefore more likely to face a backlog situation in the Receiver Proxy queues. In the screenshot below you can see the performance header of a receiver proxy message that required around 20 minutes in the PLSRV_CALL_INBOUND_PROXY (which corresponds to the posting of the application data).

Of course, such a long-running message will block the queue and all messages behind it will face a higher latency. Since this step is purely application-related, it is only possible to perform tuning at the application side.

The number of sender and receiver ABAP Proxy queues can be changed in transaction SXMB_ADM on the proxy backend with parameter EO_INBOUND_PARALLEL and subparameter SENDER (XBTS* queues) and RECEIVER (XBTR* queues). The prerequisite for increasing the number of queues is that there are enough qRFC resources are available at the proxy system (as discussed in section qRFC Resources (SARFC)).

Also, as described in chapter Message Prioritization on the ABAP Stack, prioritization can be configured in the ABAP Proxy system. This can be used to separate runtime-critical interfaces from interfaces with a long application processing (as shown above).
8 Message Size as Source of Performance Problems

The message size directly influences the performance of an interface. The size of a PI message depends on two elements: The PI header elements with a rather static size and the payload which can vary greatly between interfaces or over time for one interface (for example, larger messages during year-end closing).

The size of the PI message header can cause a major overhead for small messages of only a few kB and can cause a decrease in the overall throughput of the interface. Furthermore, many system operations (like context switches or database operations) are necessary for only a small payload. The larger the message payload, the smaller the overhead due to the PI message header. On the other hand, large messages require a lot of memory on the Java stack which can cause heavy memory usage on ABAP or excessive garbage collection activity (see section Java Memory) that will also reduce the overall system performance. Very large messages can even crash the PI system by causing an Out-of-Memory exception, for example. You therefore have to find a compromise for the PI message size.

Below you see throughput measurements performed by SAP. In general, the best throughput was identified for messages sizes of 1 to 5 MB.

The message size in these measurements corresponds to the XML message size processed in PI and not the size of the file or IDoc being sent to PI. You can use the Runtime Header of the ABAP stack to check the message size. Below you can see an example of a very small message. While the MessageSizePayload field describes the size of the payload in bytes (here 433 bytes), the MessageSizeTotal describes the total message size (header + payload). In the example this is around 14 KB demonstrating the overhead by the PI header for small messages. The next two lines describe the payload size before and after the mapping. In the example below, the mapping reduces the payload size. The last two lines determine the size of the response message that is sent back to PI before and after the response mapping for synchronous messages.

Based on the above observations, we highly recommend that you use a reasonable message size for your interfaces. During the design and implementation of the interface we therefore...
recommend using a message size of 1 to 5 MB, if possible. Messages can be collected at the sender side or by using IDoc packaging as described in Tuning the IDoc Adapter to achieve this. In case of large messages, a split has to be performed by changing the sender processing or by using the split functions available in the structure conversion of the File adapter.
9 General Hardware Bottleneck

During all tuning actions discussed in the previous chapters you must keep in mind that the limitation of all activities is set by the underlying CPU and memory capacity. The physical server and its hardware have to provide resources for three PI runtimes: the Adapter Engine, the Integration Engine, and the Business Process Engine. Tuning one of the engines for high throughput leaves fewer resources for the remaining engines. Thus, the hardware capacity has to be monitored closely.

9.1 Monitoring CPU Capacity

When monitoring the CPU capacity it is not enough to use the average that is displayed in the “Previous Hours” section of transaction ST06. Instead, you should start your interface and monitor the CPU while it is running. The best procedure is to use operating system tools to monitor the CPU usage (particularly if using hardware virtualization).

The SMD Host Agent also reports CPU data to Wily Introscope which can be viewed using dashboards, as shown below. This example shows two systems where one is facing a temporary CPU overload and the other a permanent one.

You can also monitor the CPU usage in ST06 as described below

**Procedure**

Log on to your Integration Server and call transaction ST06. Take a look at the snapshot that is provided on the entry screen, than navigate to “Detailed Analysis Menu”.

- Snapshot view: Is the idle time (upper left corner) at a reasonable value at all times, for example, around 10% or higher?
- Snapshot view: The load average is a good indicator of the dimension of the bottleneck. It describes the number of processes for each CPU that are in a wait queue before they are assigned to a free CPU. As long as the average remains at one process for each available CPU, the CPU resources are sufficient. Once there is an average of around three processes for each available CPU, there is a bottleneck at the CPU resources. In connection with a high CPU usage, a high value here can indicate that too many processes are active on the server. In connection with a low CPU usage, a high value here can indicate that the main memory is too small. The processes are then waiting due to excessive paging.

- Detailed Analysis View → TOP CPU. Which thread uses up the highest amount of CPU? Is it the J2EE Engine, the work processes, or the database?

  - There are different follow-up actions to be taken depending on the findings of the second check: The first option is of course to simply increase the hardware. This should be accompanied by a thorough sizing (for which SAP provides the Quicksizer in the Service Marketplace [http://service.sap.com/quicksizer](http://service.sap.com/quicksizer)).

  - The second option is to identify the CPU-consuming threads and to think about a reduction of the load in this component. For example, if the J2EE Engine is the...
largest consumer then it is worth re-evaluating the number of concurrent mapping threads and/or consumer queues as described in the previous chapters.

- SAP Note 742395 - Analyzing High CPU Usage by the J2EE Engine provides a good entry point if the J2EE Engine consumes too much CPU.
- If it is the work processes that consume a lot of CPU, use transaction ST03N to figure out if it is the Integration Server or the Business Process Engine. You can do that by looking at the CPU usage per user: BPE uses the user WF-BATCH while the Integration Server uses the last user who activated a qRFC queue (typically PIAFUSER or PIAPPLUSER). Use the previous chapter to restrict these activities.

9.2 Monitoring Memory and Paging Activity

As stated above, paging is very critical for the Java stack since it influence the Java GC behavior directly. Therefore, paging should be avoided in every case for a Java-based system.

The OS tools are the most reliable for monitoring the paging activity on your system. Transaction ST06 can also be used to perform a snapshot analysis. Take a look at the snapshot that is provided on the entry screen:

- Snapshot view: Is there enough physical memory available?
- Snapshot view: Is there considerable paging? Paging can have a negative influence on your J2EE Engine. If it does, this can be seen in long GC times as described in chapter 6.4.1.

9.3 Monitoring the Database

Monitoring database performance is a rather complex task and requires information to be gathered from a variety of sources. Only a basic set of indicators is given for the purpose of this guide. A more detailed analysis is required if these indicators point to a major performance problem in the database. If assistance is needed, open a support message under the component BC-DB-ORA (or MSS, SDB, DB6 respectively).

9.3.1 Monitoring Database (Oracle)

Procedure

Log on to your Integration Server and call transaction ST04.
Many of the numbers in the screenshot above depend on each other. The following checklist names a few key performance figures of your Oracle database.

- The data buffer quality, for instance, is based on the ratio of physical reads versus the total number of reads. The lower the ratio, the better the buffer quality. The data buffer quality should be better than 94%; the statistics should be based on 15 million total reads. This number of reads ensures that the database is in an equilibrated state.

- Ratio of user and recursive calls. A good performance is indicated by ratio values greater than 2. Otherwise the number of recursive calls, compared to the number of user calls, is too high. Over time this ratio always declines because more and more SQL statements get parsed in the meantime.

- Number of reads for each user call. If this value exceeds 30 blocks for each user call, this indicates an expensive SQL statement.

- Check the value of Time/User call. Values larger than 15 ms often indicate an optimization issue.

- Compare busy wait time versus CPU time. A ratio of 60:40 generally indicates a well-tuned system. Significantly higher values (for example, 80:20) indicate room for improvement.

- The DD-cache quality should be better than 80%.
9.3.2 Monitoring Database (MS SQL)

Procedure
To display the most important performance parameters of the database, call Transaction ST04, or choose Tools → Administration → Monitor → Performance → Database → Activity. An analysis is only meaningful if the database has been running for several hours with a typical workload. To ensure a significant database workload, we recommend a minimum of 500 CPU busy seconds. Note: The default values displayed in section Server Engine are relative values. To display the absolute values, press button Absolute values. Check the values in (1).

- The cache hit ratio (2), which is the main performance indicator for the data cache, shows the average percentage of requested data pages found in the cache. This is the average value since startup. The value should always be above 98 (even during heavy workload). If it is significantly below 98, the data cache could be too small. To check the history of these values, use Transaction ST04 and choose Detail Analysis Menu → Performance Database. A snapshot is collected every 2 hours.
- Memory setting (5) shows the memory allocation strategy used, and shows the following:
  - FIXED: SQL Server has a constant amount of memory allocated, which is set by SQL Server configuration parameters min server memory (MB) = max server memory (MB).
  - RANGE: SQL Server dynamically allocates memory between min server memory (MB) < > max server memory (MB).
  - AUTO: SQL Server dynamically allocates memory between 4 MB and 2 PB, which is set by min server memory (MB) = 0 and max server memory (MB) = 2147483647.
  - FIXED-AWE: SQL Server has a constant amount of memory allocated, which is set by min server memory (MB) = max server memory (MB). In addition the address windowing extension functionality of Windows 2000 is enabled.
9.3.3 Monitoring Database (DB2)

**Procedure**

To get an overview of the overall buffer pool usage, catalog, and package cache information, go to transaction ST04, choose Performance → Database → section Buffer Pool (or section Cache respectively).

- **Buffer Pools Number:** Number of buffer pools configured in this system.
- **Buffer Pools Total Size:** The total size of all configured buffer pools in KB. If more than one buffer pool is used, choose Performance → Buffer Pools to get the size and buffer quality for every single buffer pool.
- **Overall buffer quality:** This represents the ratio of physical reads to logical reads of all buffer pools.
- **Data hit ratio:** In addition to overall buffer quality, you can use the *data hit ratio* to monitor the database: \( \frac{(\text{data logical reads} - \text{data physical reads})}{\text{data logical reads}} \times 100\% \)
- **Index hit ratio:** In addition to overall buffer quality, you can use the *index hit ratio* to monitor the database: \( \frac{(\text{index logical reads} - \text{index physical reads})}{\text{index logical reads}} \times 100\% \).
- **Data or Index logical reads:** The total number of read requests for data or index pages that went through the buffer pool.
- **Data or Index physical reads:** The total number of read requests that required I/O to place data or index pages in the buffer pool.
- **Data synchronous reads or writes:** Read or write requests performed by db2agents.
Catalog cache size: Maximum size of the catalog cache that is used to maintain the most frequently accessed sections of the catalog.

Catalog cache quality: Ratio of catalog entries (inserts) to reused catalog entries (lookups).

Catalog cache overflows: Number of times that an insert in the catalog cache failed because the catalog cache was full (increase catalog cache size).

Package cache size: Maximum size of the package cache that is used to maintain the most frequently accessed sections of the package.

Package cache quality: Ratio of package entries (inserts) to reused package entries (lookups).

Package cache overflows: Number of times that an insert in the package cache failed because the package cache was full (increase package cache size).

9.3.4 Monitoring Database (MaxDB / SAP DB)

Procedure

As with the other database types, call transaction ST04 to display the most important performance parameters.
With the Database Performance Monitor (transaction ST04) and its submonitors, the CCMS in the SAP R/3 System allows you to view, in the SAP R/3 System, all of the information that can be used to identify bottleneck situations.

- The SQL Statements section provides information about the number of SQL statements executed and related sizes.
- The I/O Activity section lists physical and logical read and write accesses to the database.
- The Lock Activity section (in transaction ST04) allows you to identify possible bottlenecks in the assignment of locks by the database.
- The Logging Activity section combines information about the log area.
- The Scan and sort activity section can be helpful in identifying that suitable indexes are missing.

- The Cache Activity section provides information about the usage of the caches and the associated hit rates.
  - The data cache Hit rate should be 99% in a balanced system. If this is not the case, you need to check if the low hit rate is due to the size of the data cache being too small, or due to an unsuitable SQL statement. If necessary, you can increase the data cache by increasing the MaxDB parameter Data_Cache_Size (lower than 7.4) or Cache_Size (7.4 or higher).
  - The catalog cache hit rate should be 85% in a balanced system. It is not a cause for concern if the catalog hit rate is temporarily lower, since accesses to the system tables and an active command monitor can temporarily impair the hit rate. If the catalog cache is busy, the pages are paged out in the data cache rather than on the hard disk.
  - Log entries are not written directly to the log volume, but first into a buffer (LOG_IO_QUEUE) so as to be able to write several log entries together asynchronously with one I/O on the hard disk. Only once a LOG_IO_QUEUE page is full is it written to the hard disk. However, there are situations in which you need to write LOG entries from the LOG_QUEUE onto the hard disk as quickly as possible, for example if transactions are completed (COMMIT, ROLLBACK). The transactions wait until the log writer reports the OK informing them that the log entry is on the hard disk. Firstly, this means that it is important to use the quickest possible hard disks for the log volume(s) and secondly, you must ensure that no LOG_QUEUE_OVERFLOWS occur in production operation. If the LOG_IO_QUEUE is full, then all transactions that want to write LOG entries must wait until free memory becomes available again in the LOG_IO_QUEUE. At LOG_QUEUE_OVERFLOWS (Transaction DB50 -> Current Status -> Activities Overview -> LOG_IO_QUEUE Overflow) you need to increase the LOG_IO_QUEUE parameter.
In accordance with Note 819324 - FAQ: MaxDB SQL optimization, check which SQL statements are responsible for the most disk accesses, and whether they can be optimized. You should also optimize SQL statements that have a poor runtime and poor selectivity.

**Bottleneck Analysis**

The Bottleneck Analysis function is available in transaction ST04 under the Problem Analyzes ➔ Performance ➔ Database Analyser ➔ Bottlenecks. You can use this function to activate the corresponding analysis tool dbanalyzer.

The dbanalyzer then collects important performance data every 15 minutes and evaluates this for possible bottlenecks.

The result of the bottleneck analysis is output in text format to provide a quick overview of possible causes of performance problems. For a more detailed description of the bottleneck messages, see the online documentation (http://help.sap.com) in the SAP Web Application Server area and search for “SAP DB bottleneck analysis messages”.

The dbanalyzer tool can also be started at operating system level. It logs its analysis results in files with date stamps in the subdirectory `<rundirectory>\analyzer` (you can find the actual directory by double-clicking on ‘Properties’ in DB50) of the relevant database instance.
9.4 Monitoring Database Tables

Some of the tables of SAP PI grow very quickly and can cause severe performance problems if archiving or deletion does not take place frequently. For troubleshooting, see SAP Note 872388 – Troubleshooting Archiving and Deletion in PI.

Procedure

Log on to your Integration Server and call transaction SE16. Enter the table names as listed below and execute. In the following screen simply press the button “Number of Entries”. The most important tables are:

- SXMSPMAST (cleaned up by XML message archiving/deletion). If you use the switch procedure, you have to check SXMSPMAST2 as well.
- SXMSCLUD / SXMSCLUD2 (cleaned up by XML message archiving/deletion). If you use the switch procedure, you have to check SXMSCLUDUR2 and SXMSCLUD2 as well.
- SXMSPHIST (cleaned up by the deletion of history entries). If you use the switch procedure, you have to check SXMSPHIST2 as well.
- SXMSPFWRAWH and SXMSPFWRAWD (cleaned up by the performance jobs, see SAP Note 820622 – Standard Jobs for XI Performance Monitoring).
- SWFRXI* (cleaned up by specific jobs, see SAP Note 874708 - BPE HT: Deleting Message Persistence Data in SWFRXI*)
- SWWWIHEAD (cleaned up by work item archiving/deletion).
- Check for all tables if the number of entries is reasonably small or remains roughly constant over a period of time. If that is not the case, check your archiving/deletion setup.
- Use an appropriate database tool, for example, SQLPLUS for Oracle for the database tables of the J2EE schema. There are currently only two central table known to fill up quickly: BC_MSG and BC_MSG_AUDIT (when audit log persistence is enabled).
10 Traces, Logs, and Monitoring Decreasing Performance

10.1 Integration Engine

There are three important locations for the configuration of tracing logging in the Integration Engine: The pipeline settings of PI itself, the Internet Communication Manager (ICM), and the gateway. All three are heavily involved in message processing and therefore their tracing or logging settings can have an impact on performance. On top of this, you might have changed the SM59 RFC destinations and have switched on the trace in order to analyze a problem: If so, then this needs to be reset as well.

Procedure

✓ Start transaction SXMB_ADM and navigate to Integration Engine Configuration -> Specific Configuration and search for the following entries:

<table>
<thead>
<tr>
<th>Category</th>
<th>Parameter</th>
<th>Subparameter</th>
<th>Current Value</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>RUNTIME</td>
<td>TRACE_LEVEL</td>
<td>&lt;none&gt;</td>
<td>&lt;your value&gt;</td>
<td>1</td>
</tr>
<tr>
<td>RUNTIME</td>
<td>LOGGING</td>
<td>&lt;none&gt;</td>
<td>&lt;your value&gt;</td>
<td>0</td>
</tr>
<tr>
<td>RUNTIME</td>
<td>LOGGING_SYNC</td>
<td>&lt;none&gt;</td>
<td>&lt;your value&gt;</td>
<td>0</td>
</tr>
</tbody>
</table>

Set the above parameters back to the default value which is the recommended value by SAP.

✓ Start transaction SMICM and check the value of the Trace Level that is displayed in the overview (third line). Set the trace level to ‘1’ if not already the case. You can do so by navigating to Goto -> Trace Level -> Set

✓ Start transaction SMGW, navigate to Goto -> Parameters -> Display and check the parameter Trace Level. Set the trace level to ‘1’ if not already the case. You can do so by navigating to Goto -> Trace -> Gateway -> Reduce Level (or Increase Level, respectively).

✓ Start transaction SM59 and search the following RFC destination for the flag “Trace” on the tab Special Options and remove the Flag “Trace” if it has been set:

- AI_RUNTIME_JCOSERVER
- AI_DIRECTORY_JCOSERVER
- LCRSAPRFC
- SAPSLDAPI

It is possible that other RFC destinations are used for sending out IDocs and similar.

10.2 Business Process Engine

You only have to check the Event Trace in the Business Process Engine.

Procedure

✓ Call transaction SWELS and check if the Event Trace is switched on. The recommended setting is ‘off’ (as shown in the screenshot below).
10.3 Adapter Framework

Since the Adapter Framework runs on the J2EE Engine, the tracing and logging can be conveniently checked and controlled using NetWeaver Administrator. To improve the performance SAP recommends that you set all trace levels to default. Higher trace levels are only acceptable in productive usage during problem analysis. Below you will find a description on how to set the default trace level for all locations at once. It is of possible to do it for every location separately, but this way you might forget to reset one of the locations.

Procedure

✓ Start the NetWeaver Administrator and navigate to Problem Management → Logs and Traces and choose Log Configuration. In the Show drop down listbox choose Tracing Locations. Select the Root Location in the tree and press the button Default Configuration. This will set the default configuration for all locations.
11 Errors as Source of Performance Problems

If errors occur within messages or within technical components of the SAP Process Integration, then this can have a severe impact on the overall performance. Thus, to solve performance problems, it is sometimes necessary to analyze the log files of technical components and to search for messages with errors. To search for messages with errors, use the PI Admin Check which is available using SAP Note 884865.

Procedure

- **ICM (Internet Communication Manager):**
  Start transaction SMICM, open the log file (Shift + F5 or GoTo → Trace File → Show All), and check for errors.

- **Gateway:**
  Start transaction SMGW, open the Log File (CTRL + Shift + F10 or Goto → Trace → Gateway → Display File), and check for errors.

- **System Log:**
  Start transaction SM21, choose an appropriate time interval, and search the log entries for errors. Repeat the procedure for remote systems if you are using dialog instances.

- **ABAP Runtime Errors ("Core dumps"):**
  Start transaction ST22, choose an appropriate time interval, and search for PI-related dumps.

- **Alerts / CCMS:**
  Start transaction RZ20 and search for recent alerts.

- **Work Process and RFC trace:**
  In the PI work directory check all files which begin with dev_rfc* or dev_w* for errors.

- **J2EE Engine:**
  In the PI work directory search for errors in file dev_server0.out. If you are using more than one J2EE server node, check dev_server<n>.out as well.

- **Applications running on the J2EE Engine:**
  The trace for application-related errors are the default.trc files located in /j2ee/cluster/server<n>/log directory. To monitor exceptions across multiple server nodes, the best tool to use is the LogViewer service of the NWA. This is available using Problem Management → Logs and Traces and choose Log Viewer. Search for exceptions in the area of the performance problem, for example, the PI Messaging system.
A  Appendix

A.1  Wily Transaction Trace

As mentioned above, the Wily Transaction trace can be used to trace expensive steps that were noticed during the mapping or module processing. The transaction trace will allow you to drill down further into Java performance problems and to distinguish if it is a pure coding problem or caused by a look-up to a remote system or a slow connection to the local database.

The Wily Transaction trace can be used to trace all steps that exceed a specific duration on the Java stack. If a user is known (for example, for SAP Enterprise Portal) the tracing can be restricted to a specific user. In PI, if you encounter a module that lasts several seconds then you can restrict the tracing as shown below.

Note: Starting the Transaction Trace increases the data collection on the satellite system (PI) and is therefore only recommended in productive environment for troubleshooting purposes.

With the selection above the trace will run for 10 minutes (this is the maximum and it can be canceled earlier) and will trace all steps exceeding 3 seconds for the agents of the PI system. If such long-running steps are found then a new window will be displayed listing these steps. In the screenshot below you can see the result in the Trace View. The Trace View shows the elapsed time from left to right—in the example below around 4.8 seconds. From top to bottom we can see the call stack of the thread. In general we are interested in long-running threads on the bottom of the trace view. A long-running block at the bottom means that this is the lowest level coding that was instrumented and which is consuming all the time.

In the example below we see a mapping call that is performing many individual database statements – this will become visible by highlighting the lowest level. In such a case you have to review the coding of the mapping to see if the high amount of database calls can be summarized in one call.
Another case that is often seen is that a lookup using JDBC to a remote database or RFC to an ABAP system takes a long time in the mapping or adapter module. In such a case there will be one long block at the bottom of the transaction trace that also gives you some details about the statement that was executed.