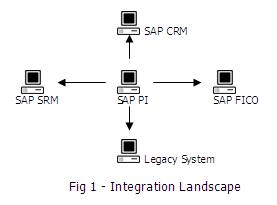
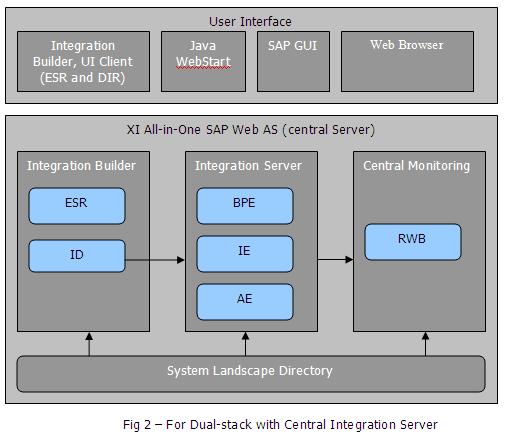
## **为什么我们需要SAP PI**



在大型的机构中，除了遗留系统之外，SAP ERP也不是由一个单一系统组成的，而是集成了多个系统，如CRM，SRM和FICO等。为了处理这种复杂性，SAP引入了PI：一个可以为所有系统提供单一集成点的平台。它不需要接触已有的遗留系统的复杂网络。这是一个可以为SAP和非SAP应用之间、企业内部和内部或者内部和外部之间提供平滑的端对端集成的强大的中间件。SAP PI支持B2B和A2A交换，支持同步和异步消息交换，并且包含了用于设计和执行PI的内建引擎。

[回到顶部](https://www.cnblogs.com/hhelibeb/p/7105070.html" \l "_labelTop)

## **SAP PI架构**



 SAP PI有着轮辐式结构，由中心和辐条组成；辐条连接外部系统，中心会在它们之间交换消息。源系统成为发送者系统，目标系统成为接收者系统。PI不是一个单独的组件，而是很多个可以根据集成场景灵活地一起工作的组件的集合。该架构包含了在设计期间使用的组件、在配置期间使用的组件和在运行期间使用的组件。

我们可以把PI划分为多个领域：

1. 集成服务器（Integration Server）
2. 集成构建器（Integration Builder）
3. 系统规划（System Landscape）
4. 配置和监控（Configuration and Monitoring）

集成服务器是SAP PI的中心处理引擎。所有消息都在这里以一致的方式处理。它包含三个独立引擎：

1. 集成引擎（Integration Engine）
2. 适配器引擎（Adapter Engine）
3. 业务处理引擎（Business Process Engine）

集成引擎可以被看做是中心，而适配器引擎则是轮辐。

关于业务处理引擎，本文会晚些解释。

集成构建器是一个用于访问和编辑集成对象的C/S框架，它包含两个相关的工具：

1. 企业服务库（Enterprise Service Repository ，ESR）——用于设计和开发在不同场景下使用的对象。
2. 集成目录（Integration Directory，ID）——用于配置开发场景的ESR组件。

二者放在一起，就是通常被成为场景的集成过程。

系统规划是数据中心的一个有关软件和系统的信息的中心库，简化了系统规划的管理。

在配置和监控中，可以监控消息和适配器。

[回到顶部](https://www.cnblogs.com/hhelibeb/p/7105070.html" \l "_labelTop)

## **单栈与双栈**

在PI初次发布的时候，不是所有的组件都是在同一个平台上构建的。集成引擎和业务处理引擎由ABAP构建，然而适配器引擎、集成构建器、SL、CM和Mapping Runtime由Java构建。因此PI需要Java和ABAP环境来运行，这被称为双栈。

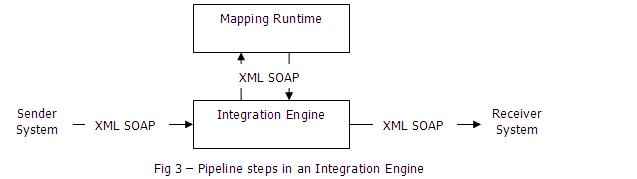
|  |  |
| --- | --- |
| **ABAP Stack** | **Java Stack** |
| 1. Integration Engine 2. Business Process Engine 3. Integration Builder | * Enterprise Service Repository * Integration Directory  1. Runtime Workbench 2. System Landscape Directory 3. Adapter Engine 4. Mapping Runtime |

但是在晚些的版本中，所有组件都是由Java构建的。某些双栈组件已经废除，或者在被修改后运行在Java栈。因此PI只需要Java环境来运行。这就是单栈。

（单双栈各有利弊，但是本文不会涉及到相关内容）

[回到顶部](https://www.cnblogs.com/hhelibeb/p/7105070.html" \l "_labelTop)

## **集成引擎**



集成引擎负责中央集成服务器服务，例如管线步骤：路由和映射。如果源消息结构和目标的消息结构不同，集成引擎调用Mapping Runtime，源结构会被转换成目标结构。Mapping Runtime基于Java栈。集成引擎也可以利用ABAP程序来转换，这个基于ABAP栈。

消息可以是两种类型：

1. 同步的——有请求和响应两部分。
2. 异步的——只有请求或者响应二者之一。

在PI中，消息由接口表示。

接口：XML格式的消息结构和说明。

基于上面的限制，会有三种接口类型：

1. 外向接口——连接发送系统。
2. 内向接口——连接接收系统
3. 抽象接口——连接BPE。

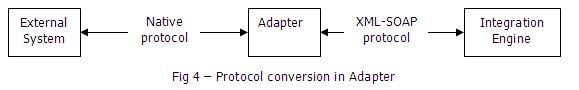
在PI中为每一个业务需求配置集成逻辑（场景）的时候，集成引擎会以循序渐进的方式执行配置。术语“管线”指的是在处理XML消息的时候执行的所有步骤。管线步骤包含：

1. 接收者识别——决定参加消息交换的系统。
2. 接口识别——判断应该使用何种接口接受消息。
3. 消息分割——如果找到了不止一个接收者，PI会为每一个接收者实例化新的消息。
4. 消息映射——把源消息映射为目标消息的格式。
5. 技术路由——为消息绑定特定的目标和协议。
6. 调用适配器——发送转换过的消息给适配器或者代理。

[回到顶部](https://www.cnblogs.com/hhelibeb/p/7105070.html" \l "_labelTop)

## **适配器引擎**

你一定已经发现，集成引擎只使用XML-SOAP协议处理消息。但是如果我们有一对发送和接收系统，它们的数据格式是不同的呢？这时我们使用适配器引擎中的不同的适配器来将XML和基于HTTP的消息转换为这些系统需要的指定的协议和格式，或者相反。



如本文早先讨论的那样，SAP PI是轮辐式结构的，其中适配器引擎可以被看作轮辐。我们使用适配器引擎来连接集成引擎（中心）和外部系统。适配器框架基于适配器引擎，适配器框架是基于SAP J2EE Connector Archtiecture（JCA）的。适配器框架提供了用于配置、管理和监控适配器的接口。

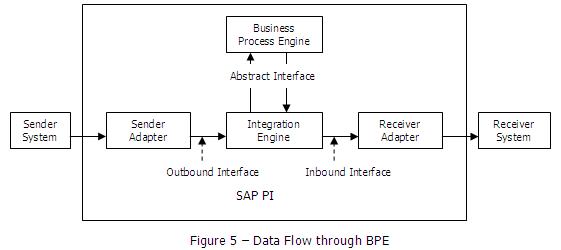
在双栈系统中，大多数适配器基于Java栈，只有两个基于ABAP栈：

|  |  |
| --- | --- |
| Java Stack | RFC adapter, SAP Business Connector adapter, file/FTP adapter, JDBC adapter, JMS adapter, SOAP adapter, Marketplace Adapter, Mail adapter, RNIF adapter, CIDX adapter |
| ABAP stack | IDOC adapter and HTTP adapter |

在SAP PI从双栈变为单栈的时候，这两个适配器成为了Java栈的一部分。修改后的适配器引擎成为高级适配器引擎，两个适配器分别叫做IDOC\_AAE和HTTP\_AAE。

[回到顶部](https://www.cnblogs.com/hhelibeb/p/7105070.html" \l "_labelTop)

## **业务处理引擎**



业务处理引擎（Business Process Engine）的职责是执行和持久化集成过程。

BPM代表跨组件业务处理管理（Business Process Management ）或者ccBPM，也叫做集成过程。集成过程是指可运行的、跨系统的消息处理。在集成过程中，你可以定义所有需要运行的的处理步骤和相关的过程控制参数。业务处理管理提供了SAP Exchange Infrastructure，包含以下功能：

1. 全状态消息处理：集成过程的状态可以在集成服务器上持久化。
2. 可以使用相关性建立消息间的语义关系。
3. 当你想要定义、控制、监控复杂的集成过程的时候，比如扩展到企业和应用程序边界，即收集/合并、拆分、多播的时候，需要实现集成过程。

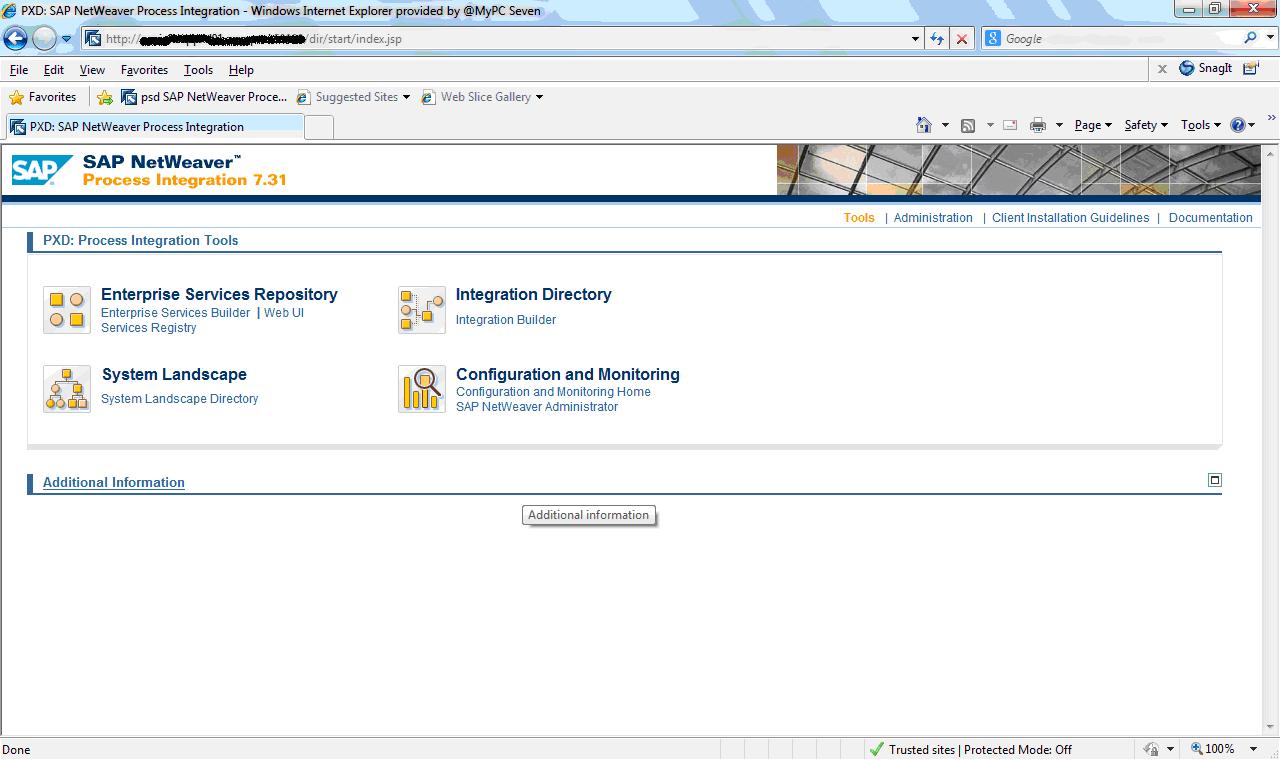
在运行期间，BPE执行集成过程。集成过程可以只通过抽象接口发送和接收消息。

[回到顶部](https://www.cnblogs.com/hhelibeb/p/7105070.html" \l "_labelTop)

## **在SAP PI中建立场景**

如果需要在PI中建立场景（scenario），要从主页开始。

主页界面如下：



主页有以下四个工作区的超链接：

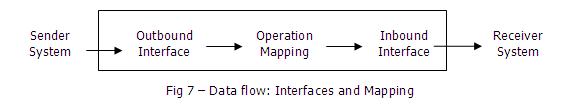
1. 企业服务库（ESR）
2. 集成目录（ID）
3. 系统规划（SL）
4. 配置和监控（CM）

每个超链接都可以打开对应的应用。这四个都是Java应用。ESR和ID是swing应用。它们基于JNLP，需要从浏览器启动，所以第一次会花较多的时间来下载整个库文件。但是从第二次开始，加载时间就会变短了。SL和CM是纯web应用，运行在浏览器上。

[回到顶部](https://www.cnblogs.com/hhelibeb/p/7105070.html" \l "_labelTop)

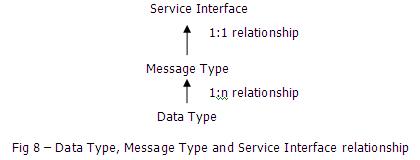
## **企业服务库**

 使用企业服务库设计和创建用于制作场景的对象。PI中的数据流是这样的：

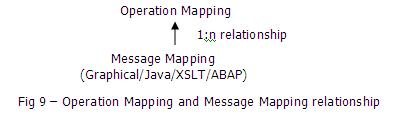


找到以下设计的选项：

1. 接口对象——服务接口，消息类型，数据类型。
2. 映射对象——操作映射和消息映射。
3. 集成过程。



PI使用集成库来为发送者和接收者设计消息结构，并且通过相应的消息结构开发接口消息，接口消息是与外部世界互动的一个点。数据类型和消息类型可以用来对复杂接口进行简化和模块化设计。



操作映射允许源结构和目标结构之间的转换。但是如果源结构和目标结构是相同的，那该过程可能会免于执行。和服务接口类似，消息映射用于简化和木块话复杂的操作映射。消息映射可以通过四种方式进行：

1. 图形化映射。
2. Java映射
3. XSLT映射
4. ABAP映射

图形化映射是最常用的手段，因为它允许开发者图形化地映射结构的属性，以通过服务接口传递数据。对于其它三个，需要通过写代码来开发映射。如果是如果是单栈服务器，ABAP映射是不可用的。

（还有些其它方面，本文没有涉及）

[回到顶部](https://www.cnblogs.com/hhelibeb/p/7105070.html" \l "_labelTop)

## **集成目录**

这里我们通过早先配置的ESR对象来制作管线步骤。这些步骤在运行期间通过集成引擎执行。

在我们开始配置之前，我们需要在DIR创建/导入以下的对象：

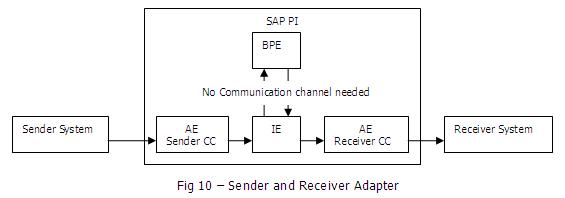
1. 服务——业务系统/业务服务/集成过程
2. 通信通道

服务允许你处理消息的发送者或者接收者。根据你使用这些服务的目的，你可以选择以下的服务类型：

1. 业务系统——如果你想要将指定的业务系统作为消息的发送者或者接收者处理，选择该消息类型。在系统规划中，业务系统是真实的应用系统。
2. 业务服务——如果你想要将抽象业务实体作为消息的发送者或者接收者处理，选择这个服务类型。业务服务不会再系统规划中定义。
3. 集成过程服务——如果你想要将集成过程作为消息的发送者或者接收者处理，选择这个服务类型。在运行期间，这些集成过程由消息控制，他们自己也可以发送消息。

通信通道决定了消息的内向和外向处理。消息会通过适配器从原生格式被转换为soap-xml指定的消息格式，或者相反。通常一个场景中会有两个通信通道：

1. 发送者信道。
2. 接收者信道。



必须为服务分配一个信道。根据服务被视为消息的发送者或接收者，信道也会有一个发送者/接收者角色，二者必须匹配。不可以把信道分配给集成过程服务。

管线步骤DIR中的通过以下四步配置：

1. 发送者协议
2. 接收者判定
3. 接口判定
4. 接收者协议

发送者协议定义了发送者的消息如何转换，因此它可以由集成系统处理。它包含：

1. 发送者组件
2. 发送者接口
3. 发送者信道

发送者协议类似于表中的主键。同一个规划中不可以有两个相同的发送者协议。

接收者协议则定义了消息如何被转换为接收者可以处理的形式。它包含：

1. 发送者组件
2. 接收者组件
3. 接收者接口
4. 接收者信道

使用接收者判定来指定消息发送的对象。可以通过定义条件以转发消息，它包括：

1. 发送者组件
2. 发送者接口
3. 接收者组件

接收者判定包含2个类型——标准的和扩展的。使用哪个取决于你想要手工指定接收者、还是在在运行期间通过映射动态地指定。

接收者判定和接口判定——加在一起通常称为逻辑路由。发送者协议和接收者协议——这两个加在一起通常成为合作协议。

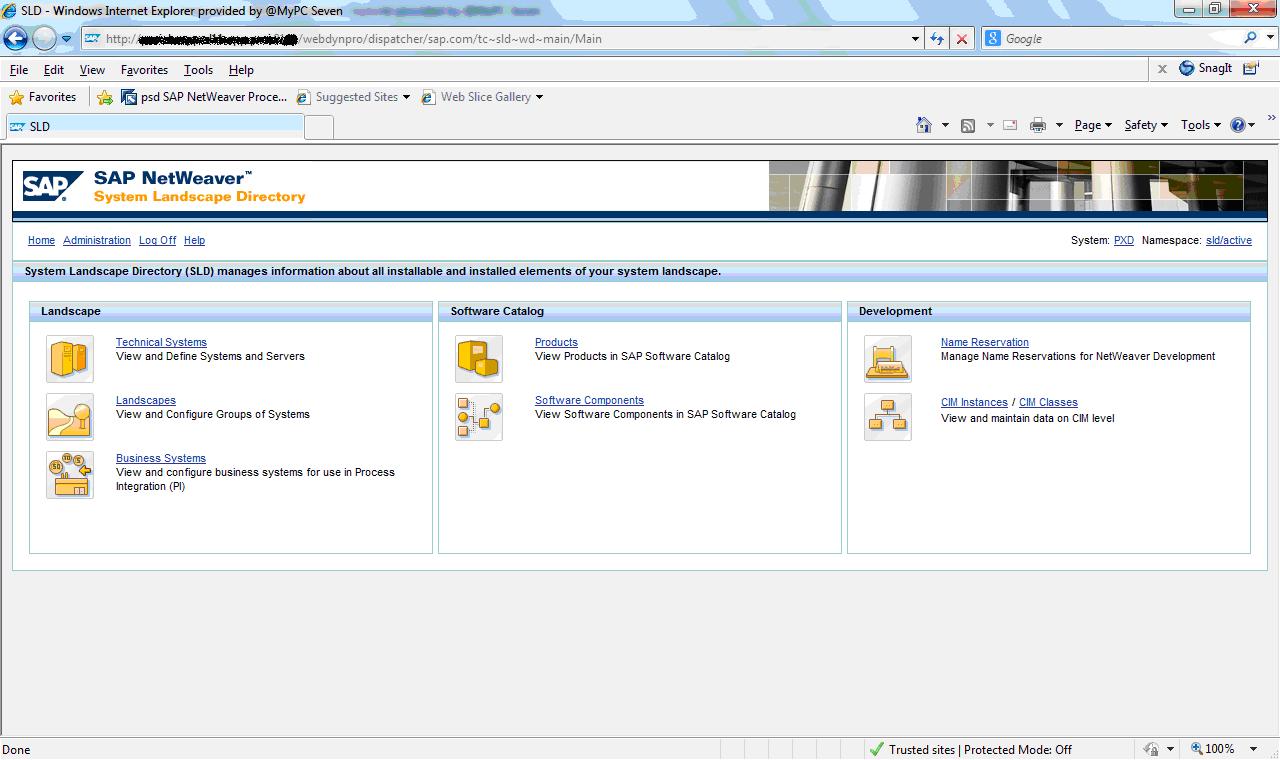
[回到顶部](https://www.cnblogs.com/hhelibeb/p/7105070.html" \l "_labelTop)

## **系统规划**

SAP System Landscape Directory（SLD）是系统规划中的核心信息的提供者。在web页面上你可以发现以下连接：

1. 技术系统——技术系统是在你的系统规划中安装的应用系统。
2. 业务系统——业务系统是逻辑系统，在PI内作为发送者/接收者存在。业务系统与相关的技术性同有着一对一的依赖关系。
3. 产品和组件——这是有关所有SAP产品和组件的信息，包含他们的版本。如果系统规划内有任何第三方产品，它们也会注册在这里。

SLD的界面如下图所示：



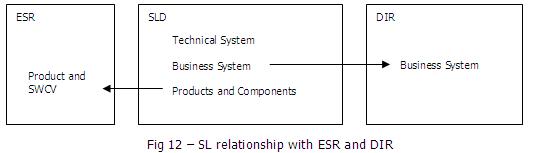
产品和组件都可以叫做组件信息。

技术系统和业务系统都叫做规划描述（Landscape Description）。

一个业务系统可以配置为集成服务器或者应用系统。

1. 集成服务器（Integration server）——集成服务器只运行在集成构建器中配置的集成逻辑。它们也可以被识别为管线步骤。它接受XML信息、判断接收者、运行映射、路由XML信息到相应的接收者系统。因此配置过的集成引擎被识别为中央配置集成引擎。
2. 应用系统（Application system）——应用系统不会执行集成逻辑。它一次调用集成服务器以运行集成逻辑。它会扮演XML消息的发送者或接收者的角色。因此，带有本地集成引擎的应用系统需要集成服务器来执行集成逻辑。

只有一个SAP系统中的客户端可以配置为集成服务器。



以下信息从SLD提取到ESR和DIR中：

1. ESR中用到的用于定义产品的组件信息和SWCV。

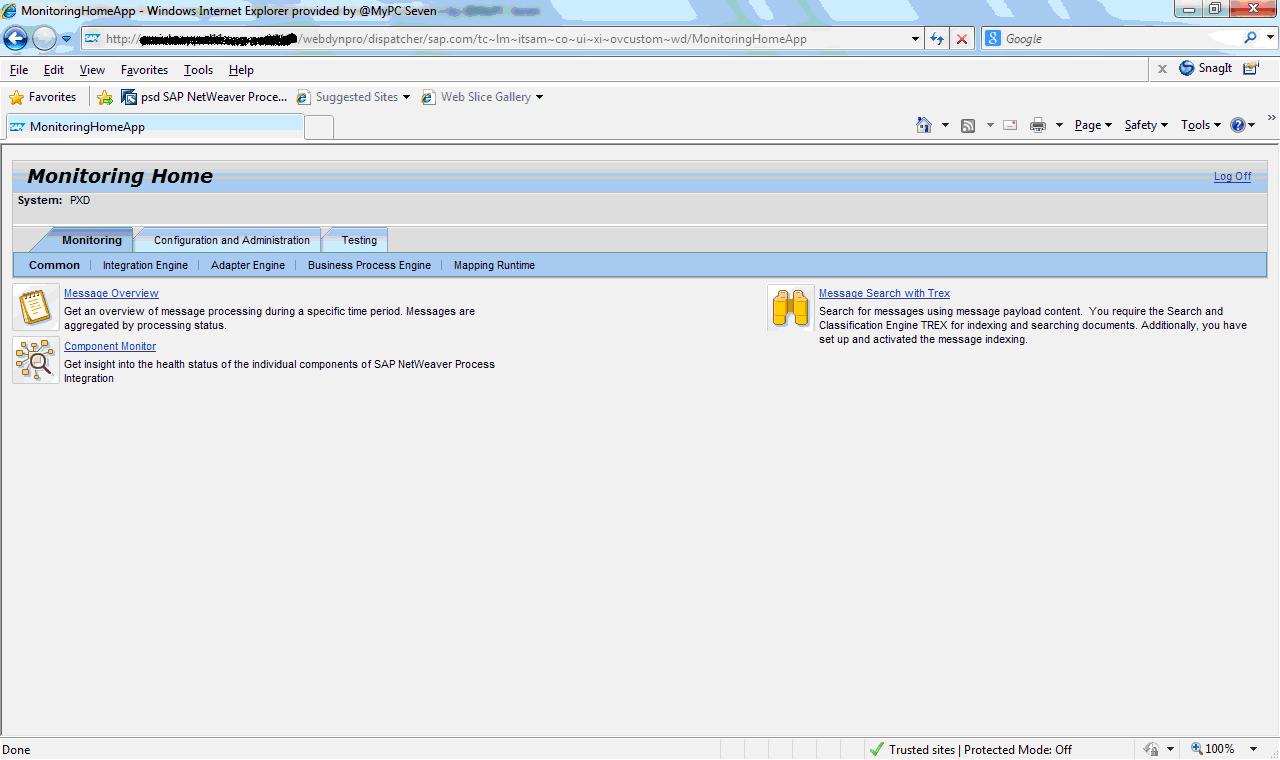
在目录中用于定义消息发送者和消息接收者的业务系统。

[回到顶部](https://www.cnblogs.com/hhelibeb/p/7105070.html" \l "_labelTop)

## **配置和监控**

配置和监控是监测的中心入口。它给予了你导航到集成引擎的功能，也可以与计算中心管理系统（Computing Center Management System，CCMS）、SAP的进程监控设施（Process Monitoring Infrastructure，PMI ）集成。

配置和监控的界面如下图：



配置和监控支持以下监控功能：

1. 组件监控——监控不同的SAP PI组件，包括Java和ABAP部分。
2. 消息监控——跟踪SAP PI组件中的消息处理状态，以及错误侦测和分析。
3. 端对端监控——从PI的视角监控消息的生命周期。
4. 性能监控——可以通过RWW统计SAP PI的不同方面的性能。这里，你可以选择并聚合性能数据，比如，根据组件、时间序列、消息属性等。
5. 索引管理——通过管理和监控每个PI组件的消息的索引，可以在消息监视中启用基于索引的消息搜索。这种消息搜索提供了增强的选择标准，包含指定适配器的消息属性和消息载荷中的术语或短语。
6. 警报配置——通过使用警报框架，PI中的中心监控可以在消息处理期间获得所有的错误报告。它可以帮助改进ABAP运行期间和基于Java的适配器引擎来改进对错误的处理。为此，警报框架包含了基于确定时间的规则，相关内容处于PI消息协议的头部。这些规则决定了警报是否发送。如果发送了警报，警报可以用于错误分析。
7. 警报信箱——警报信箱是用户特定的、显示各个警报服务器中根据警报配置而产生的所有警报。
8. 缓存监控器——缓存监控器显示当前运行时缓存中的缓存对象。不同的缓存对象的监控是依据缓存实例进行的。

[回到顶部](https://www.cnblogs.com/hhelibeb/p/7105070.html" \l "_labelTop)

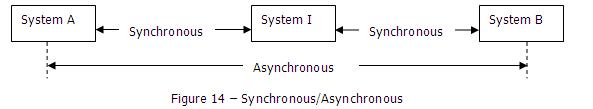
## **同步 vs. 异步**

处理可以定义为同步或者异步。

* 同步处理通过请求/响应操作调用，处理的结果立刻通过操作返回给调用者。
* 异步处理通过单方向的操作调用，结果和错误会通过另一个单向的操作调用。结果通过回调操作返回。

计算机的世界里没有异步通信，所有的两个系统之间的通信总是通过方法调用进行（请求/响应操作）。所以如何使其异步呢？答案是，在调用者和被调用者之间引入一个第三方的系统。

 假设存在两个系统——A和B。A与B之间所有的通信通过一个方法调用来进行，因此他们是同步的。我们在AB间引入一个第三方系统，称其为中间系统I。A和I之间的通信通过方法调用，I和B之间的通信也是通过方法调用进行。但是A和B之间的调用可以是异步的，因为A不需要等待来自B的响应。



这是异步通信的基本原理，那么什么是中间系统呢？答案是队列。A被称为调用者，B被称为接收者。来自于A的消息首先添加到队列中，接着它再次被从队列中拉出，并且发送给B。B的响应通过相同的方式返回给A。在某些情况下，业务需求要求消息按照以A触发的时顺序发送给B，这种情况下可以依据先进先出策略。如果没有这样的需求，则消息会以随机顺序从队列发送至B。

因此可以把消息通信分为三类：

1. 同步的
2. 异步且无序的
3. 异步且有序的

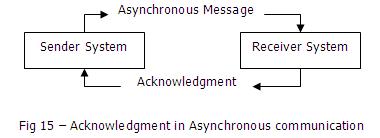
在PI中，我们定义它们为：同步——BE（Best Effort），异步且无序的——EO（Exactly Once），异步且有序的（Exactly Once in Order）。

[回到顶部](https://www.cnblogs.com/hhelibeb/p/7105070.html" \l "_labelTop)

## **确认**

确认是异步通信的基础，为什么？

对于同步通信，系统A调用系统B时，如果B发送响应失败，处理会失败。但是在异步通信中，系统A调用系统I并且系统I会调用系统B。所以假设A与I之间的通信成功，然而I和B之间的通信失败。A该怎样得知发送到B的过程失败了呢？它通过确认来实现，该确认通过消息从A到B相同的路由方式，反向发送给A。如果从B到A的确认没有成功抵达A，那么A会认为处理失败，并且再次发送消息。



当我们讨论PI中的异步的时候，我们会使用术语 ‘Exactly Once’ 来表示EO和EOIO。Exactly Onc的意思是一旦发送的消息不能再次发送。为了实现这一特性，每一个从A发往B的消息都会有一个确认。通信的终端是适配器，因此适配器必须支持确认。

所有适配器都提供系统确认（system-acknowledgment），比如发送确认。支持同步通信的适配器除了支持系统确认以外还支持应用确认。

所以在PI中存在着以下类型的确认：

1. 系统确认——系统确认在运行期间使用，以确认异步消息已抵达接收者。
2. 应用确认——应用确认用以确保异步消息成功地被接收者处理。

[回到顶部](https://www.cnblogs.com/hhelibeb/p/7105070.html" \l "_labelTop)

## **Remote Function Call**

在进行PI工作时，你会接触到名词——RFC。这是什么？为了建立两个SAP系统之间的连接，比如R/3和PI，我们创建了RFC目标。RFC目标需要配置以下内容：

1. 连接类型
2. 接收者的IP地址和端口

连接类型描述了系统连接的类型，比如R/3，TCP/IP，内部连接等等..

创建的RFC目标可以根据通信类型分类。按照异步或者同步通信可以分为：

1. 同步通信——同步RFC
2. 异步通信且无顺序——Transactional RFC（tRFC）
3. 异步通信且有顺序——Queued RFC（qRFC）