Tivoli. software

IBM

IBM White Paper: IBM Maximo 7.1 Integration Framework Architecture Basics

White Paper

Barbara Vander Weele (bcvander@us.ibm.com)

July 2008

Table of Contents

Introduction

About this Paper

The Integration Framework provides Web services and service-oriented architecture (SOA) technologies to support application services and coordination between enterprise systems and external applications. Integration can be quickly configured and customized to meet specific business requirements with predefined components and object structures.

The Integration Framework is designed with highly flexible business components to ensure compatibility with Web-based infrastructures. The technology is a proven solution that has been deployed in large enterprises and small organizations. Using the J2EE services and underlying components, the architecture uses the latest Web technologies to integrate with external applications.

This paper explores how the Integration Framework provides a solution for the synchronization and integration of data between applications.

Audience

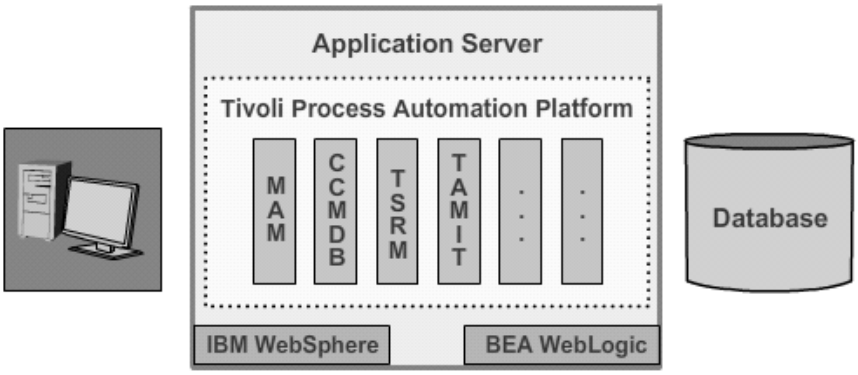
This paper is intended for technical professionals who need an introduction to the Integration Framework.

White Paper

1 Integration Framework Architecture

The Integration Framework is part of Tivoli Process Automation Platform (TPAP).[1] The environment in which the applications run is a J2EE-complaint application server. The two application servers currently used are IBM WebSphere and BEA WebLogic.

Four IBM applications currently use the TPAP common architecture and run under the application server. They are Maximo Asset Management (MAM), Tivoli Change and Configuration Management Database (CCMDB), Tivoli Service Request Management (TSRM), and Tivoli Asset Mgmt for IT (TAMIT). These four products include the Integration Framework functionality. These applications use a common set of facilities called Base Services (synonymous with TPAP), which offer a platform to develop applications.



The Tivoli Process Automation Platform includes extensive base capabilities. There is a common base storage for different databases (Oracle, SqlServer, and DB2). The database data can be accessed with a common interface. Examples of application data are assets, CIs (Configuration Items), and metadata that defines the objects.

The Tivoli Process Automation Platform is 100% J2EE compliant. All the services are Web-based, and there is a common user interface that the user can configure. There are also reports provided with the product, user-defined reports that can be created, a workflow engine, and the integration to external systems can be configured by the user.

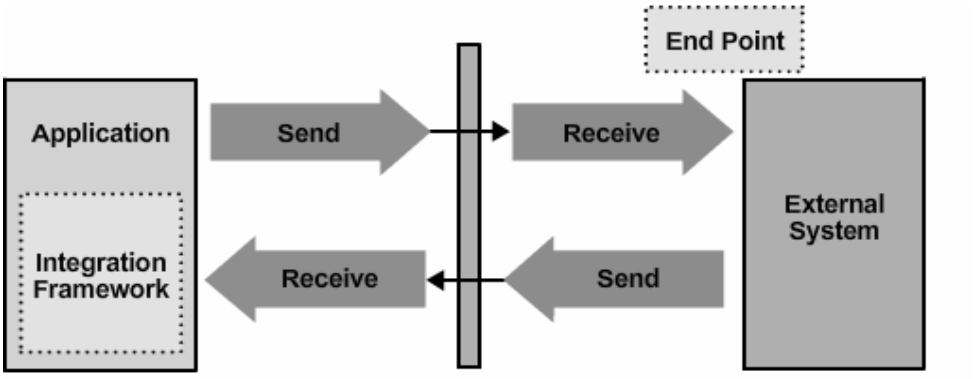
1 Base Services is synonymous with Tivoli Process Automation Platform (TPAP)

1.1 Integration Framework Basics

The Integration Framework is an integral part of TPAP and allows the synchronization and integration of data between an external system and applications that use these Base Services.

There are 57 integration objects (object structures) provided with the product that can be used by configuring the Integration Framework. The Integration Framework follows the Service Oriented Architecture (SOA) directive and implements Web Services. It is completely customizable and the behavior can be changed with rules instead of programming. There are also facilities to transform XML with XSL and Java customization.

In this paper, the word *application* as a generic term for any application that has been implemented using the Tivoli Process Automation Platform (MAM, CCMDB, and so on). The Integration Framework is available to all applications that implement this platform.



The basic concept of integration starts with an application that has been implemented using the TPAP and an external system with which to integrate.

When an event-generating action occurs on the application side, such as the approval of a purchase requisition or the addition of a new company, the Integration Framework can send the transaction information to an external system for processing.

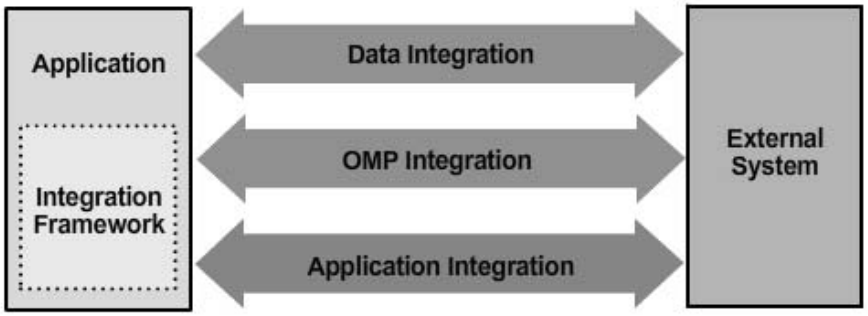
When an event occurs on the external system side, such as the approval of a purchase order, the external system can send the information to the application.

The external system can be another system (such as Oracle Financials), or it can be the same application, such as Maximo Asset Management or CCMDB. It is any system that can accept transactions. The external system (the location to which the transaction is sent) is known as an ***end point***.

1.1.1 Integration Types

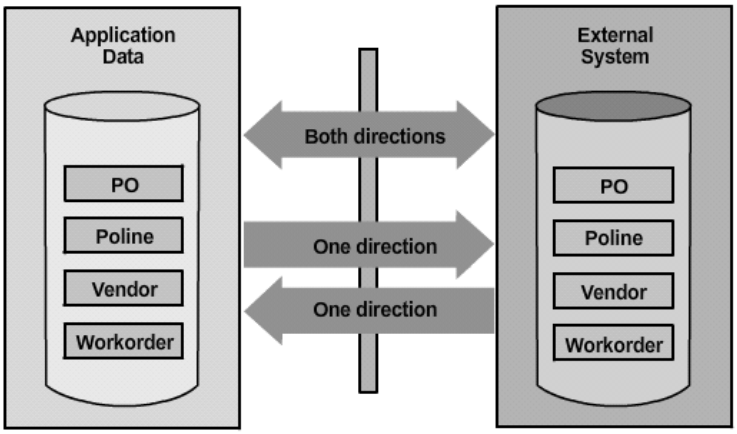
Integration types define how the integration is performed. When integrating an application with an external system, the integration can be classified as:

- **Data Integration**: In this case, the data for two applications is synchronized. Whenever something happens in the application, the external system will have the same data. The reverse is also true. This type of integration is transactional and usually asynchronous.
- **OMP Integration**: Operational Management Product (OMP) integration allows communication with an OMP, such as Tivoli Provisioning Manager (TPM). Operations specific to the OMP can be performed, such as software deployment.
- **Application Integration**: This represents real-time, synchronous communication between the application and the external system to retrieve information from each other or perform actions



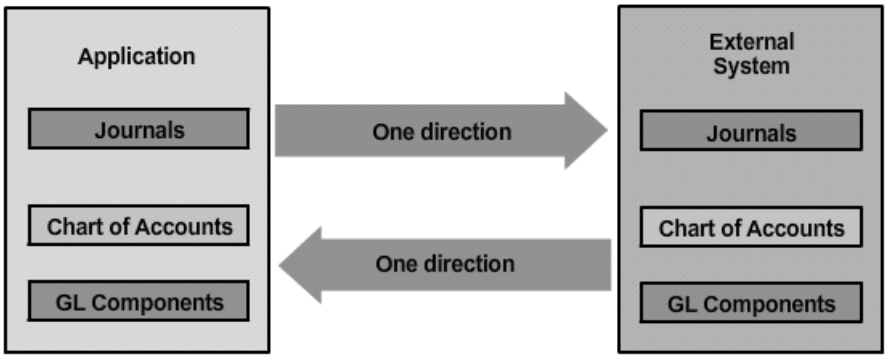
1.1.2 Integration Direction

The application data (PO, Poline, and so on) is stored in a database. The external system has similar data, but it is usually in a different format. Because the goal is typically to synchronize the data in both systems, the Integration Framework has the ability to integrate in both directions. In some cases, the data only goes in one direction.



TPAP comes with an extensive set of pre-defined integration objects. There are 57 integration objects provided with the product that do not require any customizing. For example, it is possible to send and receive PRs, POs, contracts, receipts, invoices, and so on.

An example of a situation whereby integration is only required in one direction is sending the journals of monetary transactions to a financial system such as Oracle Financials or SAP. Maximo Asset Management is not a financial system and has no reason to receive journals. In most cases, chart of accounts and general account components are created in a financial system, and Maximo Asset Management can be configured to receive them.



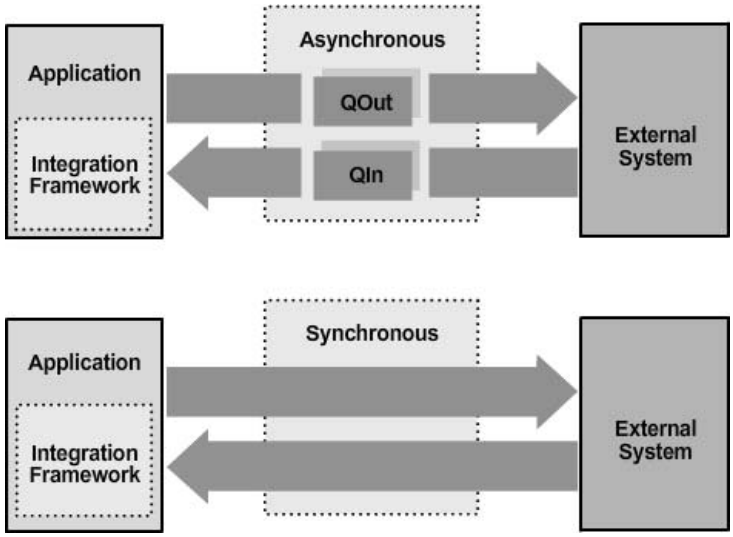
The **Database Configuration** application (one of many Maximo Asset Management applications) can be used to configure the application data in the same way as the external system data.

1.1.3 Transaction Processing

To process the integration types, the Integration Framework can integrate an application with an external system using ***asynchronous*** processing. This is not a real-time connection and the two systems communicate through a queue.

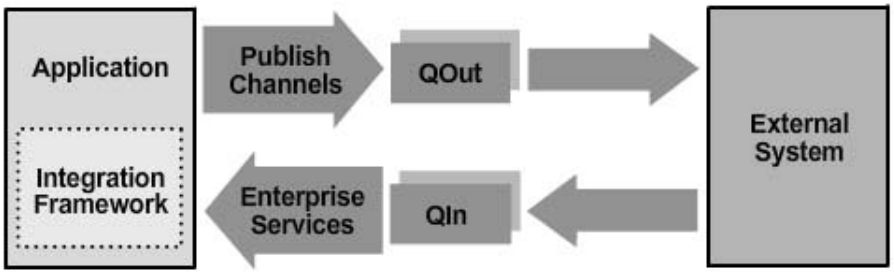
The transaction is sent from the application to a queue. From the queue, the transaction is sent to the external system. For transactions coming from the external system, the external system sends a transaction to a queue. From the queue, it is sent to the application.

Synchronous processing allows a real-time connection between the two systems. Whenever a transaction is sent from the application, it is immediately received by the external system. If a transaction is sent from the external system, it will be immediately received by the application. Synchronous processing requires that both systems be running.



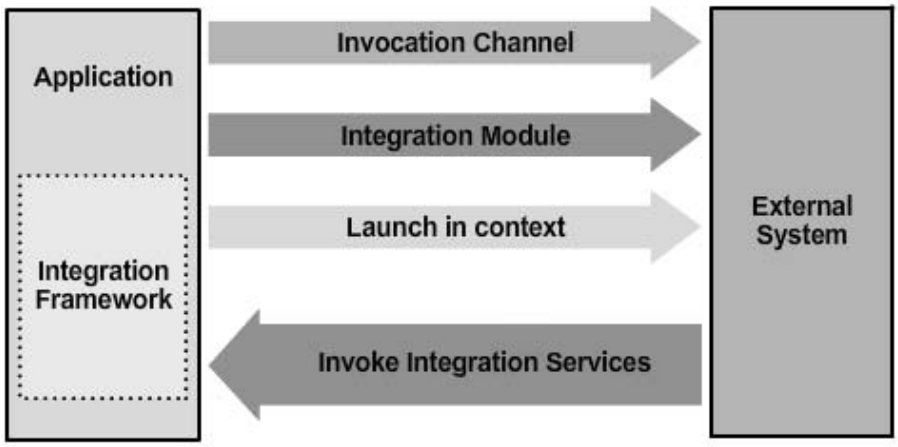
1.1.3.1 Asynchronous Processing

To process transactions asynchronously, the application is effectively publishing the transaction data to an external system. The data is configured to go out asynchronously on publish channels. From the publish channel, the transactions go to the queue.



For incoming transactions, Enterprise Services allow transactions to be received asynchronously.

1.1.3.2 Synchronous Processing



The Integration Framework offers several mechanisms for synchronous processing that involves real-time integration without queues.

One option is an ***invocation channel***, whereby something is invoked in another system. A Service Oriented Architecture environment enables external services for processing data from multiple sources. Invocation channels support SOA features and allow the system to call an external service synchronously. The system also returns the service response to the caller for subsequent processing.

The ***Integration Module*** and ***Launch in Context*** capabilities allow integration with OMPs such as Tivoli Provisioning Manager. The Integration Framework provides a mechanism to launch an external system, such as TPM.

An external system can access the application by invoking the Integration Services.

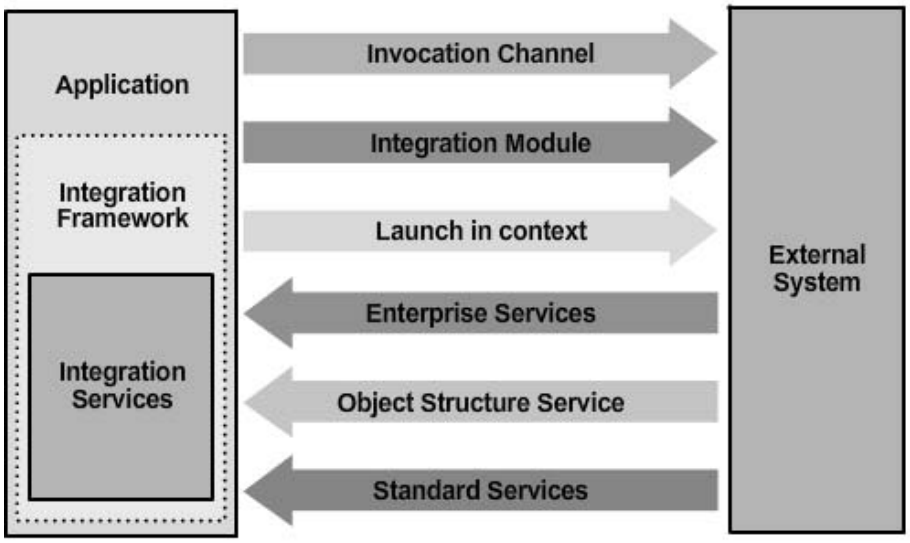
1.1.3.3 Incoming Transactions

The Integration Framework has services for incoming transactions. It uses a servlet that accepts an HTTP request, or an EJB (Enterprise Java Bean), or Web Services.

Users can program on the external system side with HTTP, or use EJBs or a Web service to send transactions to the Integration Framework.

Three services for incoming transactions are:

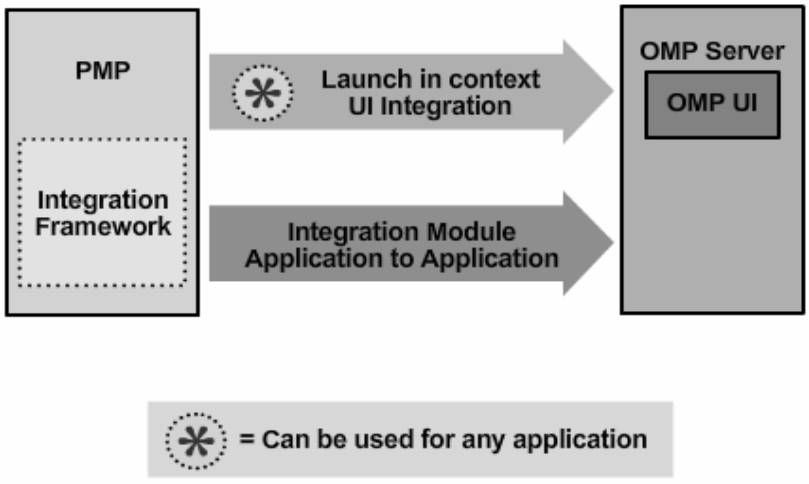
- Enterprise Services
- Object Structure Services
- Standard Services



The ***Enterprise Services*** can be customized for communication and indirectly uses the object structure. The Enterprise Services is a layer above the object structures. The object structures allow more customization. The ***Object Structure Services*** use the object structures directly. The ***Standard Services*** allow calls to service methods that are part of the Base Services. When Maximo Asset Management starts, a number of services are logged to the console during startup. Methods in those services are exposed and can be called.

1.1.4 OMP Integration

Operational Management Product (OMP) integration takes place through a series of system calls and invocations. The context in which to launch can be defined. For example, the system can be configured to launch and display a specific part number in an external application.

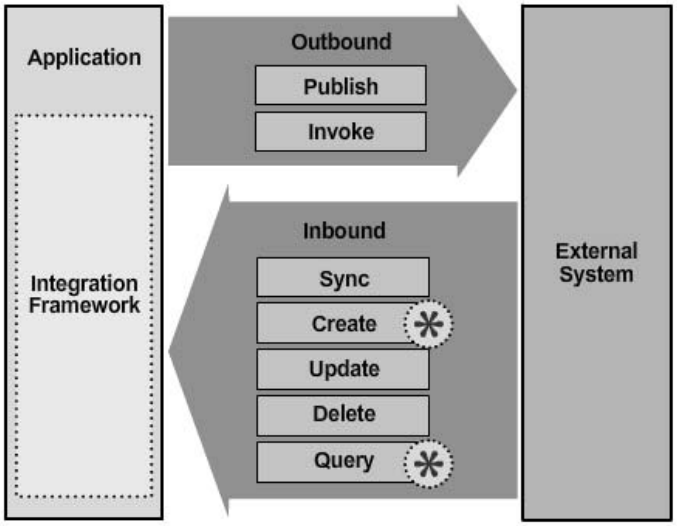


A Process Manager Product, which is part of CCMDB, calls an Integration Module (IM). which in turn communicates with the OMP. With this framework, actions such as software deployment can be automated whereby the PMP uses Integration Modules to invoke the OMP to perform the work.

The Launch in Context capability (although developed for OMP Integration) can be used for UI integration with any external application. The Integration Module can be used only with OMPs.

1.1.5 Operations

The operations identify the purpose of an integration transaction.



Two types of operations identify outbound transactions:

- ***Publish*** is for asynchronous transactions generated by a publish channel.
- ***Invoke*** is for synchronous transactions that use invocation capabilities.

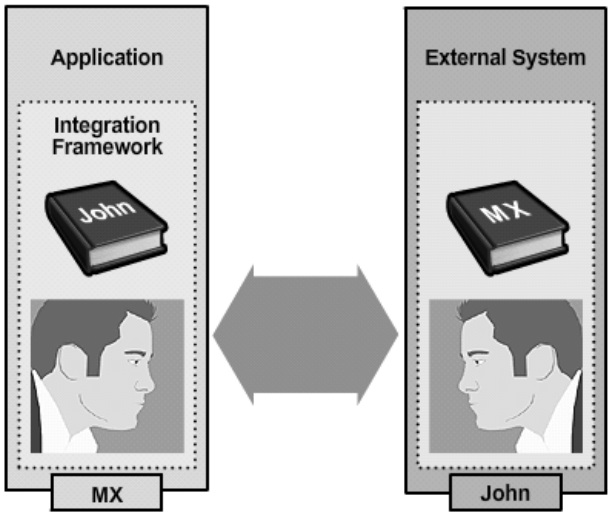
Five types of operations identify inbound transactions:

- The ***Sync*** operation allows data synchronization. It updates the record if it exists. Otherwise, a new record is created.
- Use ***Create*** to create a new record.
- ***Update*** indicates that the record is going to be updated. An error is produced if record does not exist.
- ***Delete*** indicates that an existing record needs to be deleted. An error is generated if the record does not exist.

- ***Query*** is used for query and response. The query returns a response with all the data defined in the response object.

1.1.6 Communication

The process of integration is based on two systems exchanging messages. If the application communicates with an external system, it sends a message. The external system can reply with another message. To communicate, the systems must agree on the language to be used and rules to follow. The external system is made aware of a book of messages (the language) and a set of rules that it must follow.

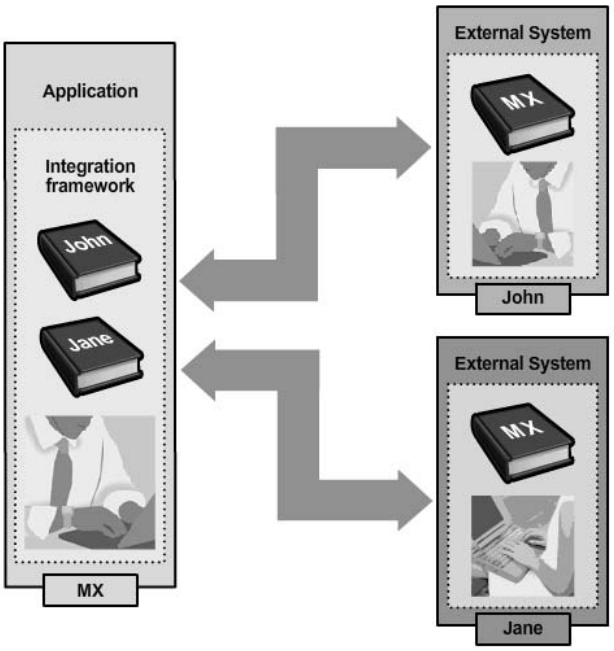


An introduction is necessary. The application introduces itself as MX. The external system introduces itself as John.

The application is using the Integration Framework and keeps track that it is communicating with John and by John's messages and rules.

Rather than communicate with only one external system, it is possible to communicate with other external systems using the same book of messages and rules.

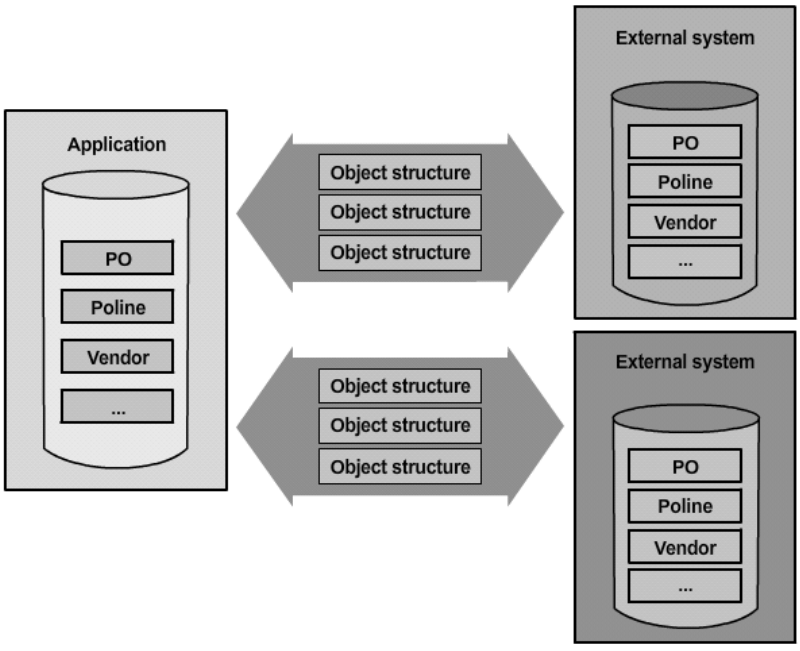
To communicate with an external system named Jane, record in the Integration Framework the application is communicating with Jane, and using Jane's messages and rules.



To do this, the application learns the new language and rules by a customization to the Integration Framework.

1.1.7 Object Structure

When integrating data, it is necessary to send or receive a transaction with data. The ***object structure*** represents the transaction data content.



When communicating with different external systems, it is possible to exchange different objects with different external systems. The messages contain the object structures.

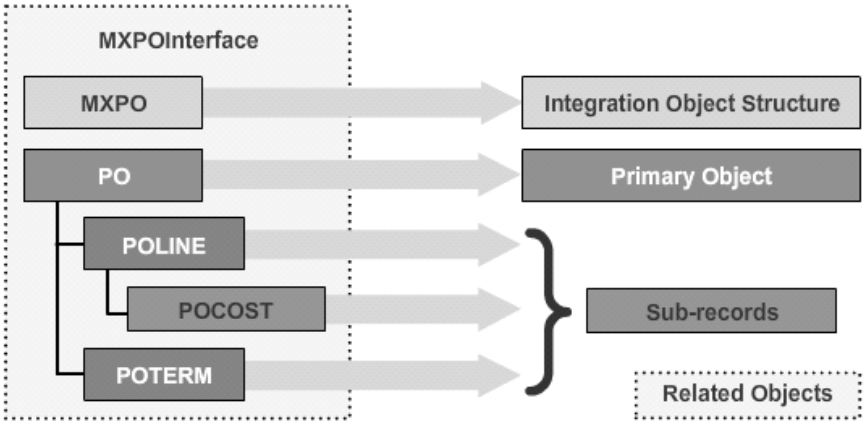
An object structure defines the content of a message being sent to or received from the external system.

The application has data in the database, such as PO and poline. The object structure content is defined based on the application data. This content is what is exposed to the external system.

1.1.7.1 Object Structure Content

This example uses a PO (purchase order) to study the content of an object structure. The name of the integration object structure for the PO is MXPO.

PO is the primary or main object. The subrecords represent the data in a PO.

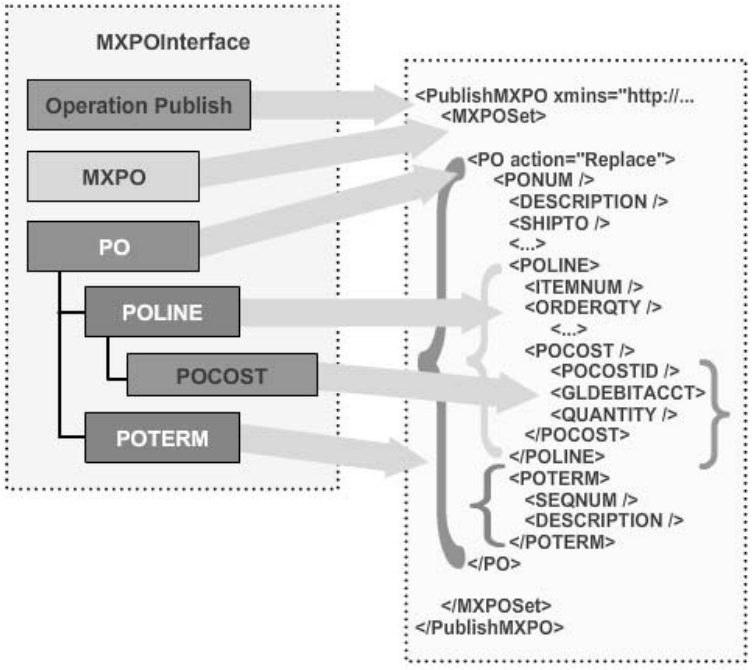


The data associated with a PO starts with the PO object. All the related objects, such as POLINE, POCOST, and POTERM are necessary. These objects constitute a PO.

The intent is to publish this structure. The publish channel will be MXPOInterface.

The XML represents the hierarchy in the PO object structure. Examining the XML illustrates how a transaction uses it.

The root element illustrated below indicates the operation, which in this case is Publish. This transaction is outbound. Following the tag with the name of the object structure (MXPO) is the data in the PO object and its related objects (POLINE, POCOST, POTERM, and so on).



A transaction can have several POs, each with several POLINES, and each POLINE with several POCOSTs.

When sending messages, the contents are known.

The message format for the integration transaction could be the default format. The default format is the format in which the application has the data. For example, for Maximo it would be the Maximo data format; for CCMDB it would be the CCMDB format.

The transaction can be in the external system format. For example, if integrating with SAP, then the transaction would be in SAP format.

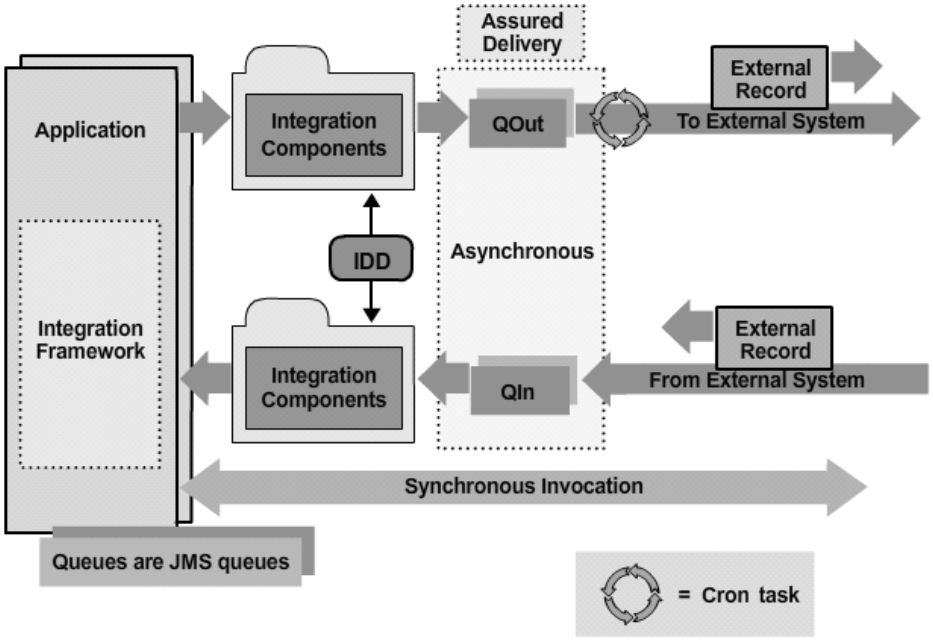
Internal application format is transformed to the external system format for transactions going out and vice versa through customization.

1.2 Inbound and Outbound Integration Process Flow

The asynchronous process goes through queues and the synchronous process does not use queues.

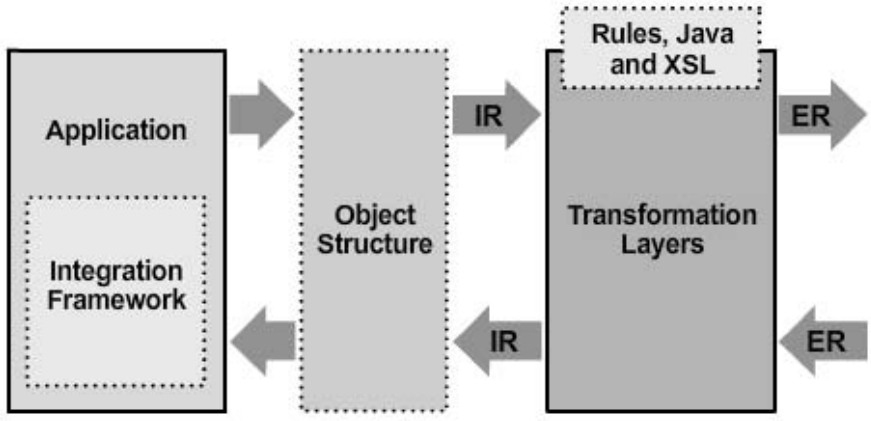
When it has a transaction, the application calls an integration component to build the transaction. The integration component is a component in the Integration Framework. To build the transaction, the integration component uses the object structure content. The object structure definition is kept in the Integration Data Dictionary (IDD).

After the XML for the transaction is created, it is put in a queue to be sent to the external system. A cron task (a Base Services feature to schedule a job) polls the queue. If records exist, it will read them and send them to the external system.



To illustrate the process in reverse, start from an external system. The external system has a record with data collected from the database. The record is placed in the queue. That record goes to the integration component which checks the data dictionary to determine what to do with the transaction.

The queues are all JMS queues. The delivery is assured.



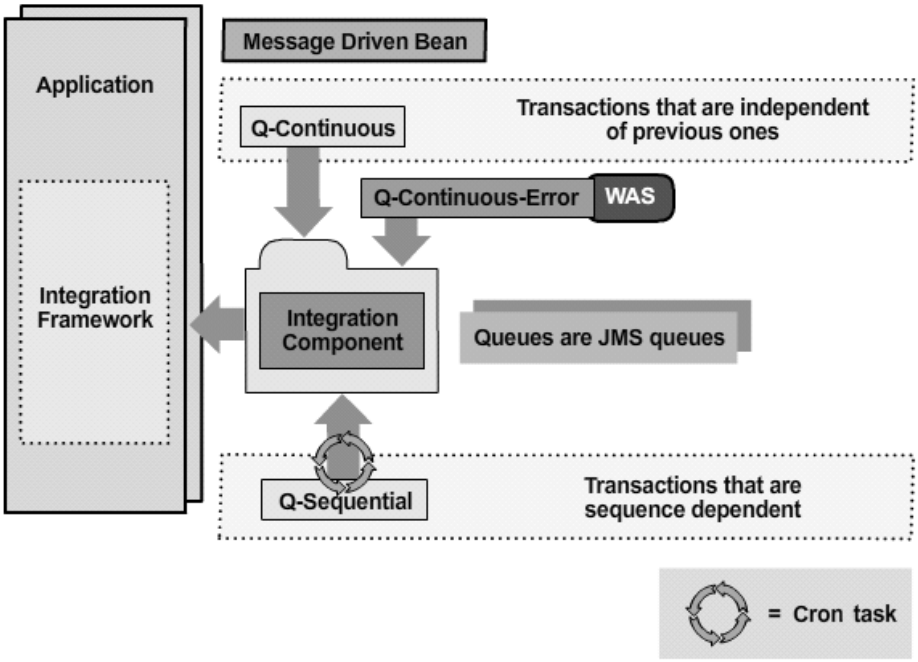
The object structure is the base object for the transaction content. The XML content represents the internal record (IR).

For outbound transactions, this record can move through a series of transformation layers. Using rules (no programming), Java, or XSL, the record can be customized for the external system. The external system receives the external record (ER) in a format understood by the external system.

The process for inbound is the reverse. The external record goes through a series of transformations to convert it to a format that is understood by the application.

1.2.1 Inbound Queues

For inbound transactions, the Integration Framework uses two types of inbound queues: continuous and sequential.



The sequential queue is for transactions that are sequence dependent, such as purchase orders, receipts, and invoices. For example, a receipt cannot be processed before the purchase order is processed. Another example is exchange rates. This type of transaction should go through a sequential queue; otherwise the data would be inconsistent.

Sequential transactions are processed one by one. If the first one fails, the queue comes to a halt.

Other transactions do not depend upon each other. If one fails, the next one can still be processed. These transactions can go to the continuous queue. The Integration Framework can simultaneously process several transactions from the continuous queue.

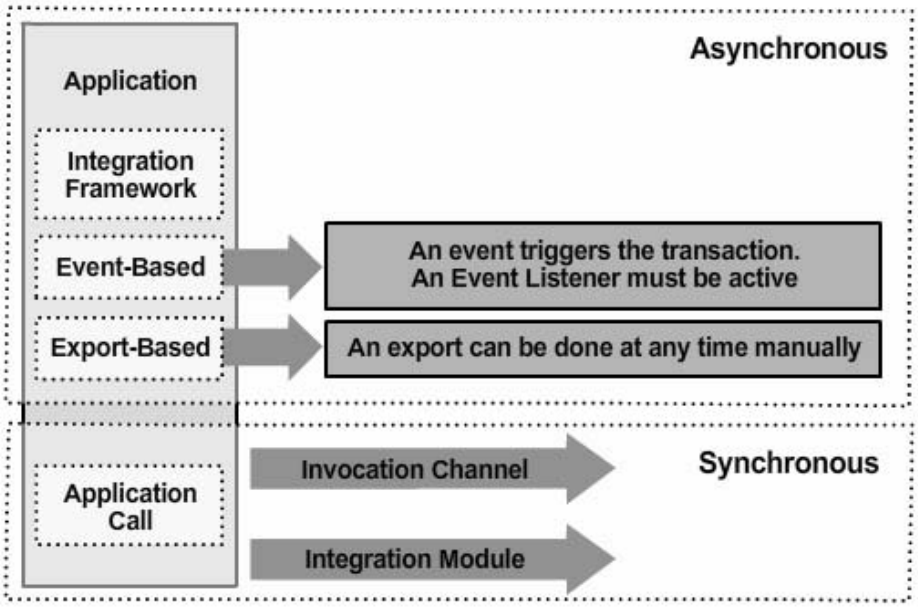
The WebSphere Application Server (WAS) has a second queue that can be configured to process the transactions that fail in the continuous queue. When a transaction in the continuous queue fails, it is placed in the error queue to allow the continuous queue to continue processing other records.

The continuous queues are JMS queues and processed with a message-driven bean. The sequential queue is driven by a cron task.

The continuous queue processes multiple records simultaneously. That number can be adjusted. IBM WebSphere defaults to 5 and BEA WebLogic to 8.

1.2.2 Outbound Transactions

There are two types of outbound transactions: asynchronous and synchronous. The two ways of generating asynchronous transactions are event-based or export-based.



Event-based means that when something happens on the application side, such as the approval of a work order, a transaction is sent out.

Export-based allows any publish channel to be exported manually at any time.

There are two ways of sending a transaction synchronously:

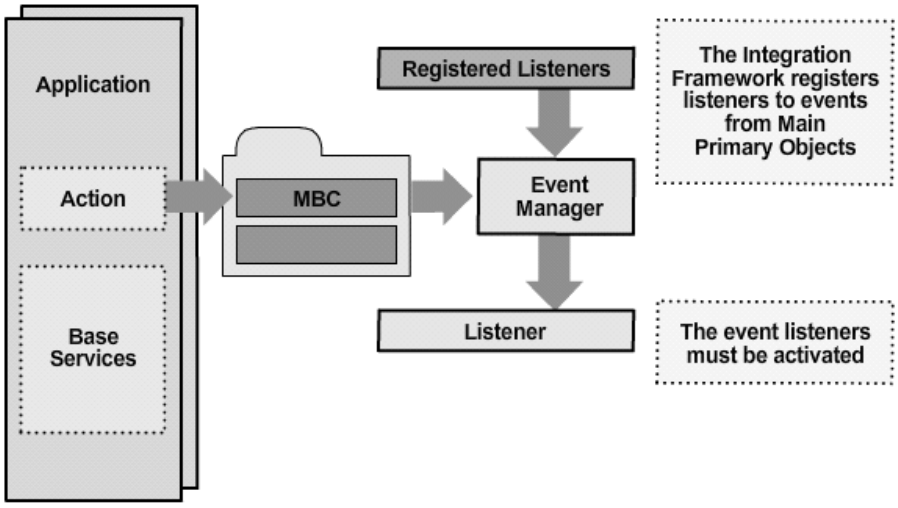
- Invocation channel
- Integration module

The invocation channel lets the application call an inbound facility on any external system and can use any of the end points supported by the Integration Framework.

The integration module is used by the OMP integrations. Usually, there is no need to use an integration module for cases other than OMP because the invocation channel would serve the same purpose.

1.2.3 Events

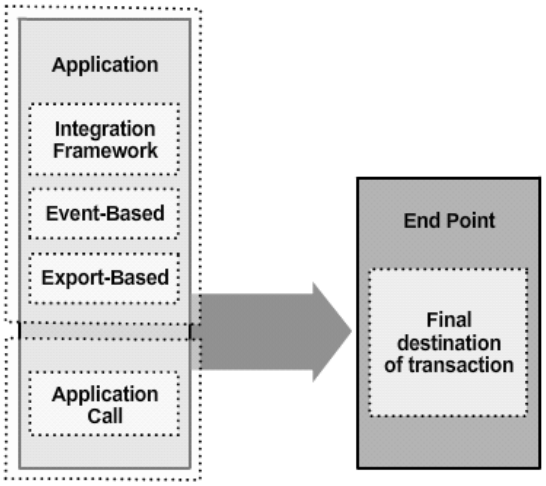
An ***event*** is an action within the Base Services framework. When an action occurs on the application side, its MBC (Maximo Business Component) is called. When the MBC adds to, updates, or deletes from the database, it generates an event. Examples would be the creation of a work order or an update of a company.



The Base Services has an event manager to register listeners for events that occur in the application. Listeners are Java classes that are executed when an event occurs. And they must be activated. When an event occurs, the Event Manager checks the list of the registered listeners for that event. If the listener exists, it instantiates that class and executes it.

The Integration Framework registers listeners for all the main primary objects. For the Integration Framework to process the call, the listener must be activated in the Integration Framework's **Publish Channels** application.

1.2.4 Outbound Transaction End Point



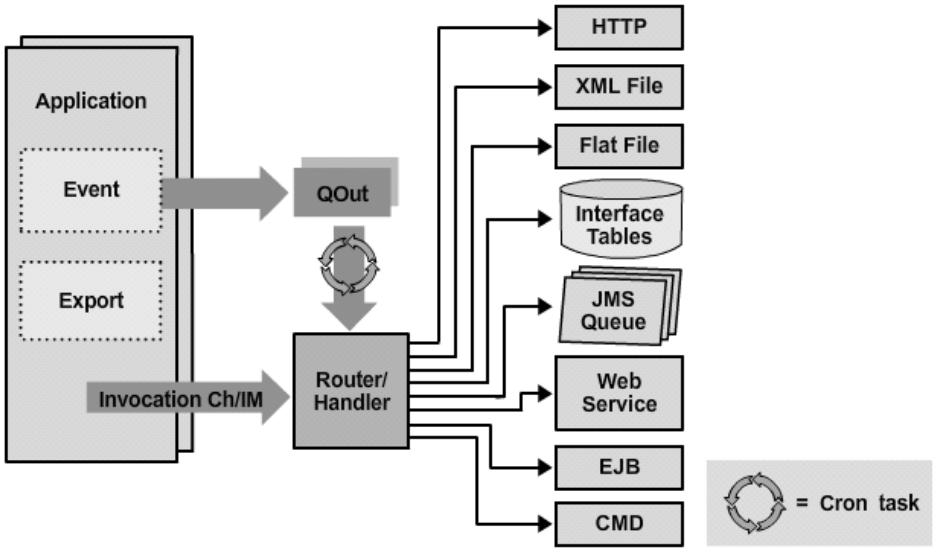
When an outbound transaction is processed, the end point is the transaction's final destination in any of the following situations:

- An event occurs and the listener is active
- An export operation is executed
- The application calls an external system

The event-based and export-based operations are configurable, and the application call is custom-coded.

1.2.5 Outbound Flow

For outbound transactions, the Integration Framework has multiple options for sending the transaction to an end point.



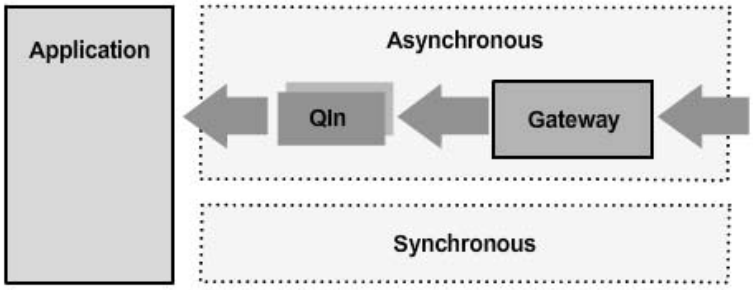
The transaction can asynchronously go to a queue or synchronously via an invocation channel or integration module. In either case, the Integration Framework has a router that handles the transaction and sents it to the end point.

The end points supported by the Integration Framework are:

- HTTP
- XML file
- Flat file
- Interface tables
- JMS queue
- A call to a Web Service
- A call to an EJB (Enterprise Java Bean) in another application server
- A call to a command in another system (new for the OMPs)

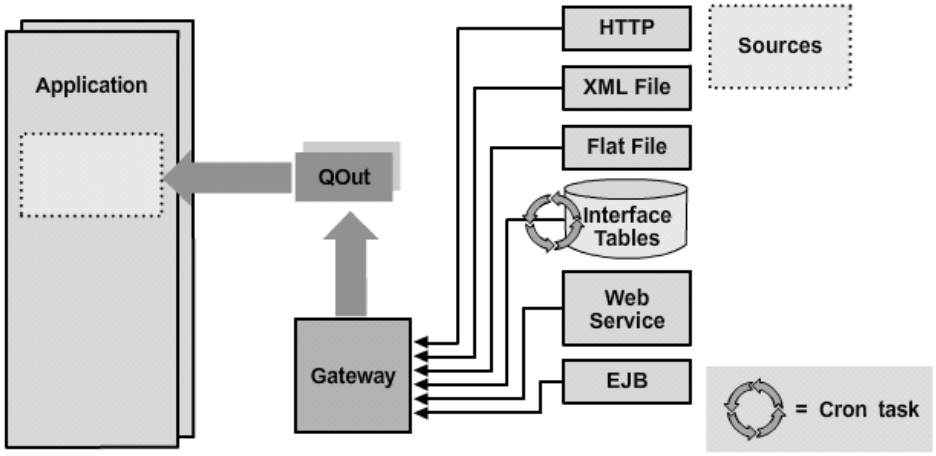
1.2.6 Inbound Flow

The inbound transactions can come asynchronously through a gateway and to a queue, or arrive synchronously with a real-time call.



The asynchronous transaction goes through a gateway facility that determines if the transaction should be placed in the queue**.**

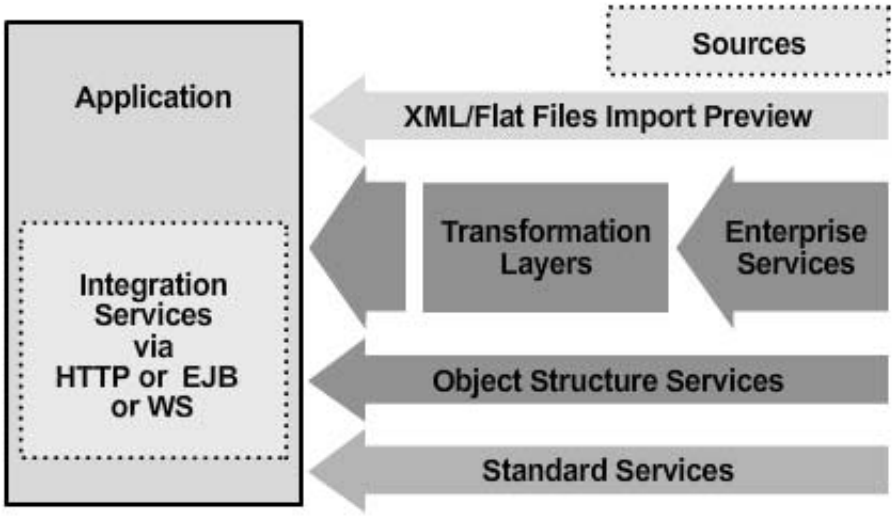
1.2.6.1 Asynchronous Inbound Flow



The asynchronous inbound transactions go through a gateway and to a queue. Those transactions can use HTTP because the Integration Framework has a servlet that accepts HTTP requests. It also has an EJB (Enterprise Java Bean) that can be called. Another possibility is a Web Service that has been deployed to be processed.

Inbound transactions can also come from XML, flat files, or interface tables that have been populated from the external system,

1.2.6.2 Synchronous Inbound Flow



The synchronous inbound transactions can be sent to the application in multiple ways. An XML or flat file can be imported using the manual import process with the Preview option. The transaction is processed but not committed. The user receives a message indicating whether the transaction would be successful or not, and what error it would produce.

Integration services can be accessed using HTTP, calling an EJB, or a Web Service. The Integration Framework provides three types of services to an external system:

- The Enterprise Services, which can be configured for synchronous processing and they can go through the different transformation layers.
- Object Structure Services is a direct call to the object structure defined in the Integration Framework. They do not go through transformation layers.
- Standard Services are direct calls and invoke a method of a service defined in the Base Services.

In conclusion, an understanding of the Integration Framework architecture provides the foundation for learning to configure it for enterprise applications and their complementary external applications.

Conclusion

Summary

You should now have a basic understanding of the underlying architecture of the Integration Framework.

Acknowledgements

Special thanks to Guido Viarisio, IBM Software Developer, for developing graphics to accompany this paper, in addition to an ongoing flow of information regarding the Integration Framework technology and its components.

