



webMethods Integration Cookbook

Document Control Information**Document Edit History**

Version	Date	Additions/Modifications	Author
1.0.0	26-Aug-2011	Initial Draft	Santosh Arora K
1.0.1	23 –Sep-2011	<ul style="list-style-type: none">• Best Practices• Performance Tuning	Ramesh Babu Dondapati Santosh Arora K
1.0.2	26-Sep-2011	<ul style="list-style-type: none">• TIPS	Santosh Arora K

Document Review/Approval History

Date	Name	Title	Comments

Distribution of Final Document

The following people are designated recipients of the final version of this document:

Name	Title

Table of Contents

1	Introduction	5
2	WebMethods Architecture	6
2.1	webMethods Platform Usage	6
2.2	Points To Ponder	6
2.3	Illustrative webMethods product stack mapping for SOA blueprint	7
3	webMethods Components	8
3.1	webMethods Designer	8
3.2	webMethods BAM	9
3.3	WebMethods BPM	9
3.4	webMethods Integration Server	10
3.5	webMethods Centrasite	11
3.6	webMethods Broker Server	12
4	Integration Design Patterns	13
4.1	Database-centric Integration	13
4.2	Pattern Variant – Staging Tables	13
4.3	Technical Implementation – Databases	15
4.4	SOAP/HTTP Integration	16
4.5	Pattern Variant – B2B – External Partner	16
4.6	RMI Integration	17
4.7	File-based Integration	17
4.8	Pattern Variant – SFTP	17
4.9	Pattern Variant – Store and Forward	18
4.10	Pattern Variant – XCOM	18
4.11	Message-based Integration	19
4.12	Typical Broker Notification Pattern	19
4.13	Full, Delta and Key Notification Variants	20
4.14	Asynchronous Communication Between Layers	21
4.15	Pattern Variant – AND JOIN at Consumer	21
4.16	Pattern Variant – OR JOIN at Consumer	22
4.17	Pattern Variant – Asynchronous Response	22
4.18	Synchronous Communication Between Layers	23
4.19	File Transfer between Service Layers	24
4.20	Pattern Variant – FTP Polling at Target	24
5	Performance Tuning	25
5.1	Trigger Tuning	25
5.2	Trading Networks Run Time Performance Tuning Tips	26
5.3	How can I make best use of “watt” properties ?	27
6	Best Practices - Developer	28
6.1	Pipeline Management	28
6.2	Documents	30
6.3	XML Strings and Documents	30
6.4	Flow Services & Java Services	30
6.5	Input Validation	31
6.6	Publish-Subscribe model	31
6.7	General / Programming Guidelines	32

6.8	Error Handling Using Flow Sequences.....	34
7	Best PracticesS - webMethods Trading Networks.....	35
7.1	Trading Networks Profiles	35
7.1.1.1	Document Types.....	35
7.1.1.2	Processing Rules.....	36
8	TIPS.....	38
8.1	Tip on Publishing event and dimensional data to Optimize.....	38
8.2	Tip on Duplicate documents getting subscribed by JMS trigger.	38
8.3	Tip on Importing a profile in Trading Networks.....	39
8.4	Tip on Retry Mechanism.....	39
8.5	Tip on webMethods rosettanet	40
8.6	Tip on XML Schema Validation	41
8.7	Tip on webService in C#.....	41
8.8	Tip on Java Service	42
8.9	Tip on TN Convert To Values	42
8.10	Tip on EDI X12 Document.....	42
8.11	Tip on JDBC Adapter.....	43
8.12	Tip on JDBC Adapter.....	43
8.13	Tip on Java Service Development Restriction.....	43
8.14	Tip on TRY CATCH BLOCK.....	44
8.15	Tip on Publishable Document Type.....	45
8.16	Tip on loop over a document	46
8.17	Tip on Broker Connectivity	47
8.18	Tip on Schedulers.....	48
8.19	Tip on MyWebMethodsServer	48
9	Documentation.....	49

1 Introduction

The objective of this webMethods cookbook is to provide the insight into webMethods architecture and help the Architects, Developers, and Project Managers to implement the integration projects in webMethods platform. This document is not intended for the wM rookies.

2 WebMethods Architecture

2.1 webMethods Platform Usage

The webMethods overall platform is presented below:

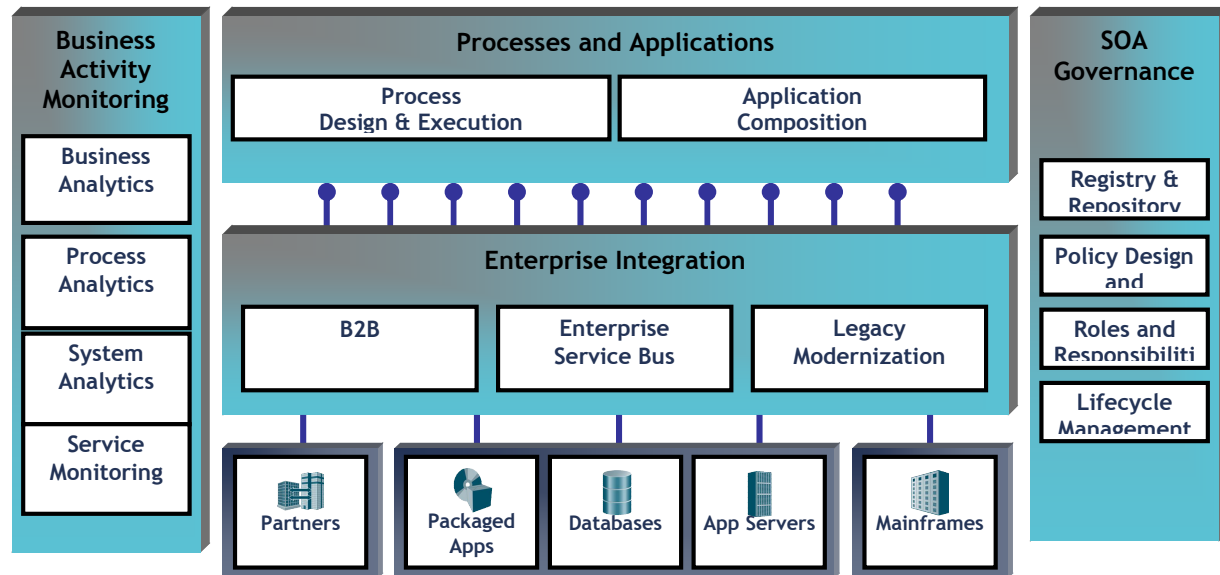


Figure 1.1

2.2 Points To Ponder

The ESB platform should be designed to support multiple business processes and integration with multiple applications on a common infrastructure. There are some key concepts to keep in mind while defining the architecture:

- All projects that are implemented on this platform should be in a position to re-use common components and functionality
- Adding a new integration should be done in such a way that future processes may re-use the interface
- New services that perform generic tasks should be created so they are re-usable by other future processes, and will expand the functionality of the platform.

2.3 Illustrative webMethods product stack mapping for SOA blueprint

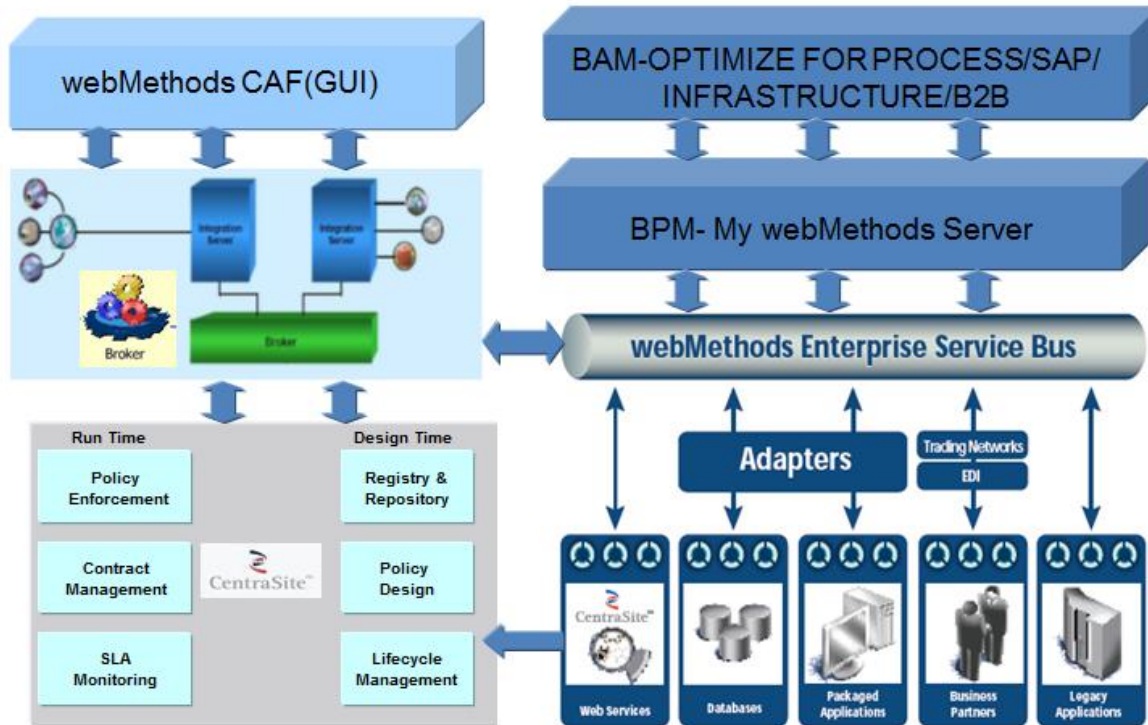


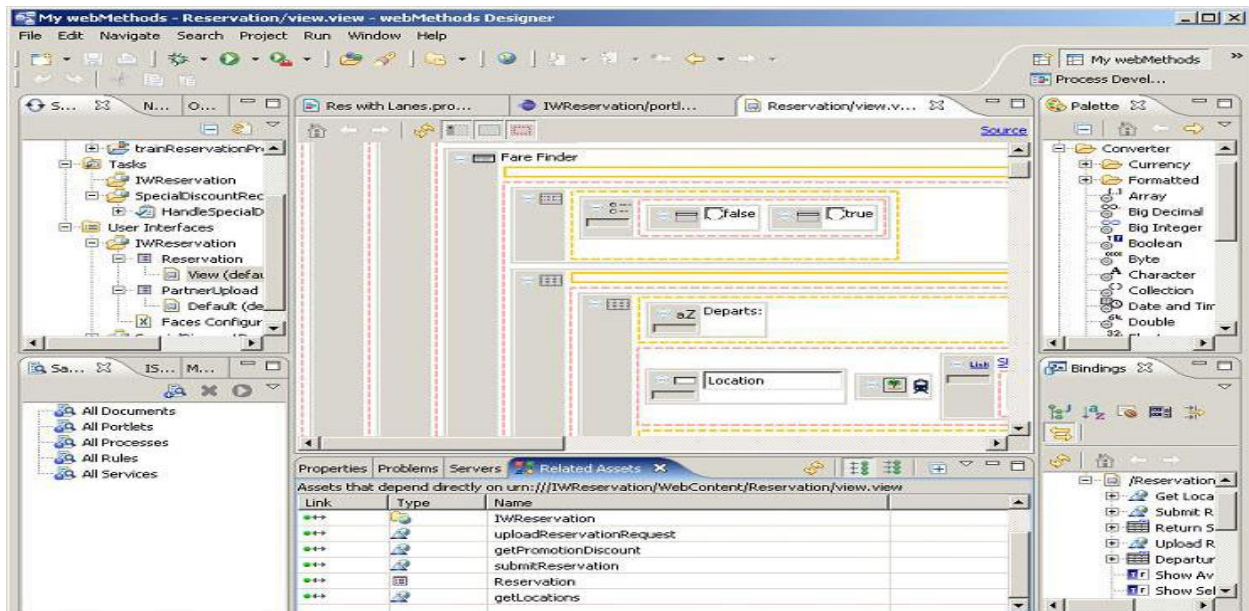
Figure 1.2

3 webMethods Components

3.1 webMethods Designer

webMethods CAF (GUI)

webMethods CAF allows you to create Web application projects for use with application servers and portlet application projects for use with My webMethods Server. A Web application is an application that a user accesses over the Web on either the Internet or an intranet. As created in the webMethods CAF, a Web application is made up of one or more views (pages) intended to be displayed by an application server, such as Apache Tomcat or JBoss. A portlet application is composed of one or more portlets, which in turn are made up of one or more views (pages) intended to be displayed by My webMethods Server.



3.2 webMethods BAM

BUSINESS ACTIVITY MONITORING -BAM

- **Optimize for Process** is a Business Activity Monitoring (BAM) solution that gives you real-time, actionable insight into process activity. You see how processes are performing across your entire enterprise. You're alerted to potential issues and can immediately identify the culprit. You even see how to correct the problem
- **Optimize for B2B** is a Business Activity Monitoring (BAM) software solution that helps you analyze and understand interactions with your trading partners. You'll gain visibility into transactions and trends across your trading network
- **Optimize for Infrastructure** is a Business Activity Monitoring (BAM) solution that gives you real-time insight into the performance of webMethods Integration Server and Broker—the servers they run on and the applications they're hosting. Optimize for Infrastructure: Mainframe Edition is an activity monitoring solution that gives you real-time insight into the performance of your Adabas and Natural systems as well as ApplinX and EntireX.
- **Optimize for SAP** leverages existing technology investments by integrating with SAP at the most granular business-event level, providing real-time metrics monitoring within and across SAP systems.

3.3 WebMethods BPM

webMethods Business Process Management Suite



KEY FEATURES

1. Collaborative Process Modelling and Orchestration
2. Process Simulation and Testing
3. Rapid Application and UI Development
4. Human Task Facilitation and Collaboration
5. Realize expected ROI with industry best practices and thorough business case analysis of your BPM project.

KEY BENEFITS

6. Achieve greater operational efficiency
7. Identify customer problems early and manage SLAs easily with business activity monitoring and proactive problem alerts.
8. Tackle changing market and competitive conditions by providing rules management for business users to make changes without affecting the underlying infrastructure.
9. Accelerate delivery of new business projects through drag-and-drop process modelling, advanced simulation and rapid user interface development.
10. Decrease the cost and time of deployment by intelligently searching and reusing existing assets such as existing processes and web services.

3.4 webMethods Integration Server



- Simplify with a single platform for developing your SOA and integrating all your internal and external applications, partners and suppliers.
- Support any integration pattern. You can support on-demand, scheduled or eventdriven interactions. Clicking a button on a Web form, scheduling a nightly batch transfer, updating a packaged application—any of these events can trigger the start of something inside the webMethods ESB.
- Build on your existing IT investments. Re-use existing software investments and expose those assets as business services by taking any integration logic service and creating a Web service or WSDL.
- Make changes easily. Point-to-point integrations are brittle and costly to maintain. Use the webMethods ESB, instead, to streamline integration projects, reducing their time and cost.
- Innovate faster for the business. Open, standards-based integration enables rapid use and deployment of Web services. Graphical service creation means you can build, edit and test integration logic without low-level coding.

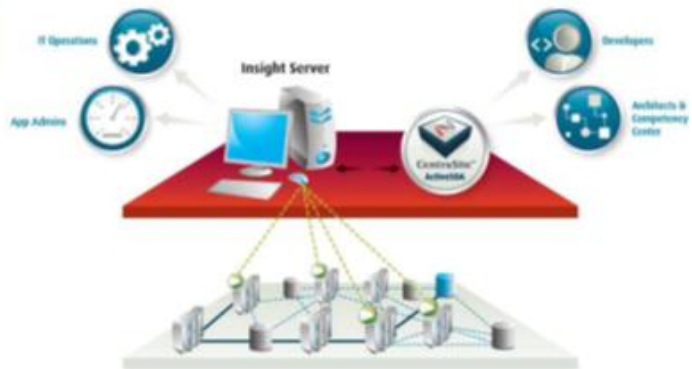
3.5 webMethods Centrasite

CentraSite integration—Improve transparency and accountability

webMethods Insight and CentraSite are fully integrated so you can improve transparency and accountability across all SOA stakeholders.

KEY BENEFITS

- ✓ Rogue service detection
- ✓ Easy access to run-time metrics
- ✓ Locating service consumers
- ✓ Last-mile security for Mediator



webMethods Insight Server is an SOA visibility tool that shows you what's happening in real-time with service transactions as they flow across any system in your SOA. Insight works and integrates with CentraSite to give you a full and accurate picture of all applications and services.

CentraSite™—The Business Service Repository for SOA and BPM that eliminates re-work, assures consistent quality and keeps your services aligned with business needs.

CentraSite™—CentraSite shall provide the infrastructure required for design-time and run-time governance of the webMethods platform.
Mediator – Mediator makes sure that requests from and responses to consumer applications conform to service policies defined in CentraSite. Mediator runs on Integration Server.

3.6 webMethods Broker Server

Guarantee fast message delivery with webMethods Broker.

webMethods Broker—A reliable ,high-performance backbone for messaging between your applications. Use Broker to execute real-time, message-oriented interactions in your event-driven architecture.



Capabilities

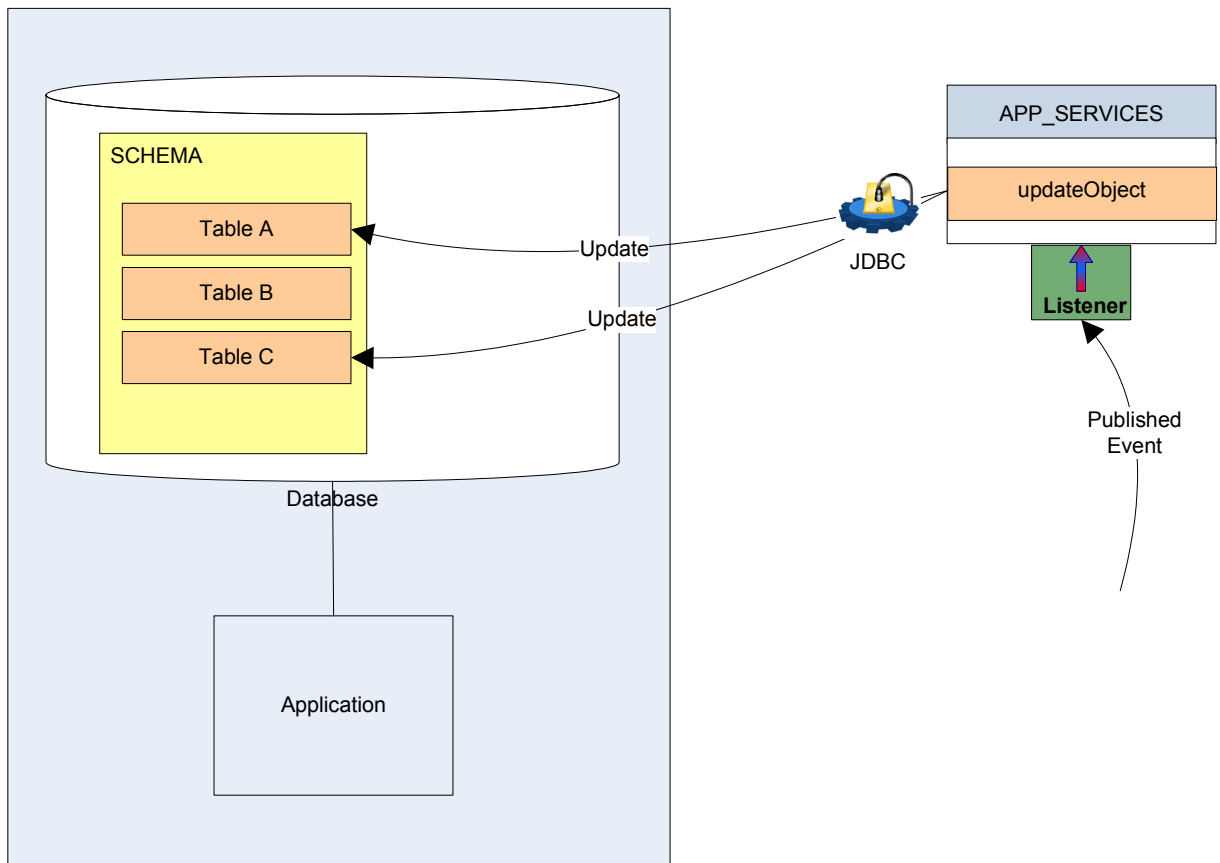
- ✓ Configurable quality of service
- ✓ Support for any messaging style
- ✓ Policy-based clustering

webMethods Broker—supports Broker to Broker connectivity via Territories and Gateways.

4 Integration Design Patterns

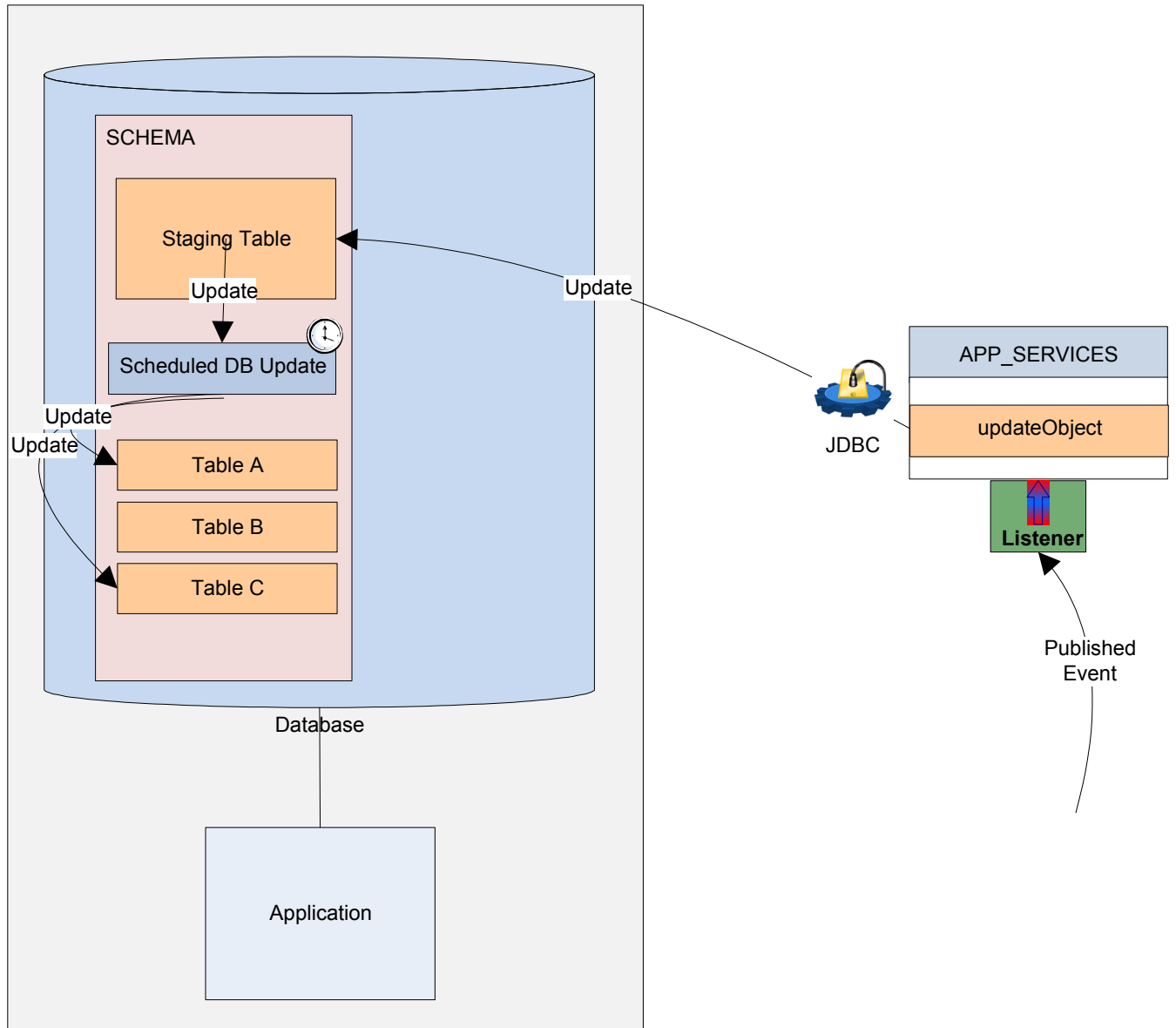
4.1 Database-centric Integration

This pattern follows the update of a target application's database (master tables) from the service bus. In the example diagram below it shows the trigger as a published event, however these types of integration can also be fronted by synchronous protocols such as SOAP.



4.2 Pattern Variant – Staging Tables

In this variant database updates are made to a staging table, and the application itself schedules updates to its master tables from the staging table.



4.3 Technical Implementation – Databases

All database reads can either be via standard webMethods adapter service (simple queries), or via a stored procedure (e.g. multiple table reads and joins). All database updates will be via a stored procedure to maintain data integrity in the target system.

All webMethods adapter services return a 'result' variable. This should be utilized to support alerting and logging. This result will either indicate the success of the query (single row queries), or return the number of rows for multiple row queries.

The recommendation for implementing database listeners (adapter notifications) in webMethods is to use the BasicNotification type. There are two types of adapter notifications in webMethods.

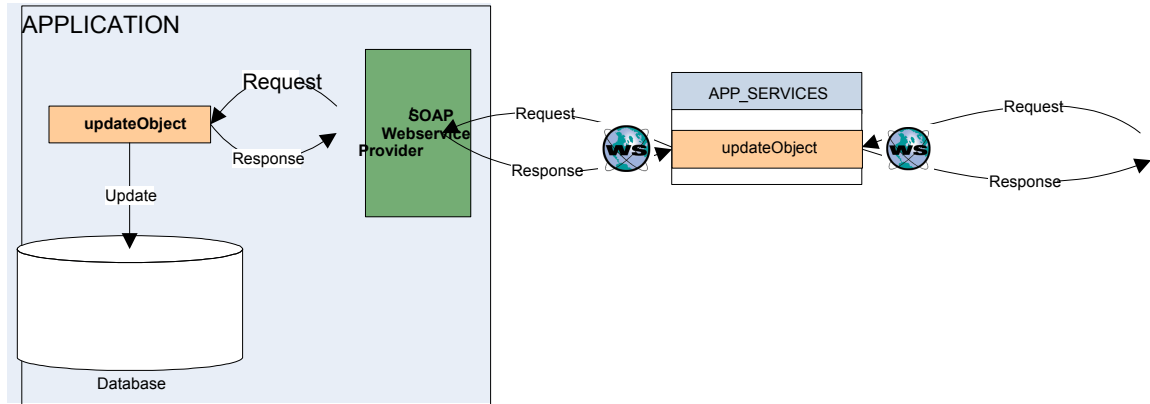
1. **Wizard-driven Notifications.** These are the INSERT, UPDATE and DELETE notifications. These set up their own buffer tables and DB triggers on the database (which is not always what a DBA would like). When these listeners are disabled, they remove these DB components. When they are activated, they re-create these components. Any loss in DB connectivity will result in these components becoming 'out-of-sync' with webMethods, and will cause issues.

2. **Basic Notifications.** This type of trigger connects to already existing DB trigger and buffer tables. This allows the DBA to set these components up and have control over them. When this type of webMethods trigger is disabled, the DB components are retained. Connectivity issues will not cause any synchronization issues with webMethods.

All database notifications will retrieve a record key, this key will be used to then call a stored procedure to collect all associated data required from one or many tables (i.e. this is a two stage process, the trigger does not return all the data).

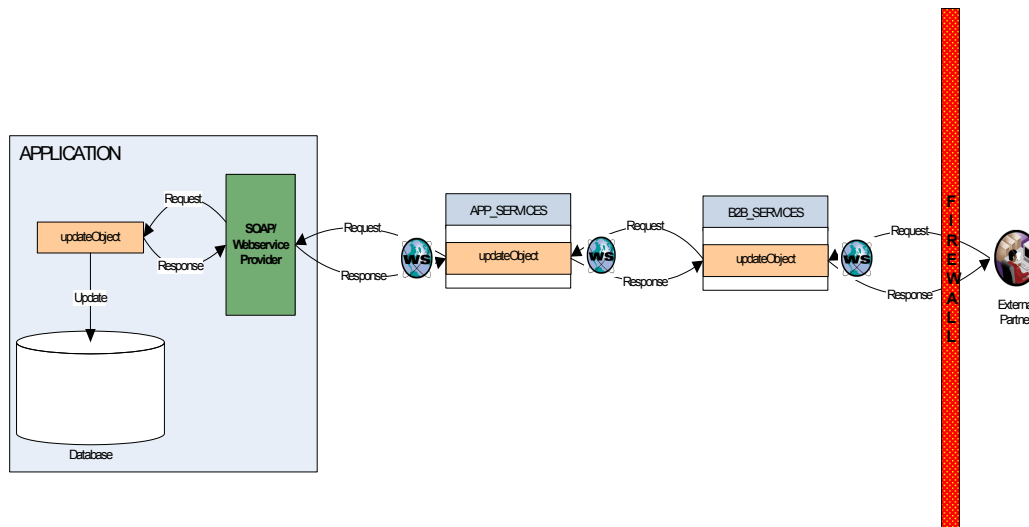
4.4 SOAP/HTTP Integration

Synchronous interactions between service layers and applications may use web-services or HTTP. The request and response will be within the same synchronous session.



4.5 Pattern Variant – B2B – External Partner

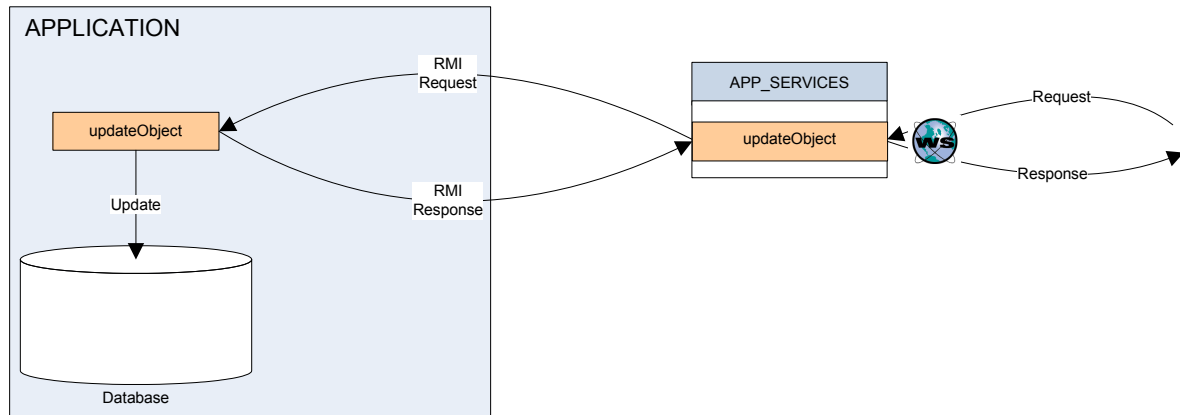
Synchronous interactions between the service bus (external service layer) and an external partner will typically be via SOAP. The request and response will be within the same synchronous web-service session.



Interactions with external partners will employ secure connection (HTTPS) and digital certificates. Payloads may be encrypted as required.

4.6 RMI Integration

Synchronous interactions between service layers and target applications can use an RMI mechanism (e.g. EJB call). This is a synchronous invocation of an exposed bean or service inside the target application from the service bus.

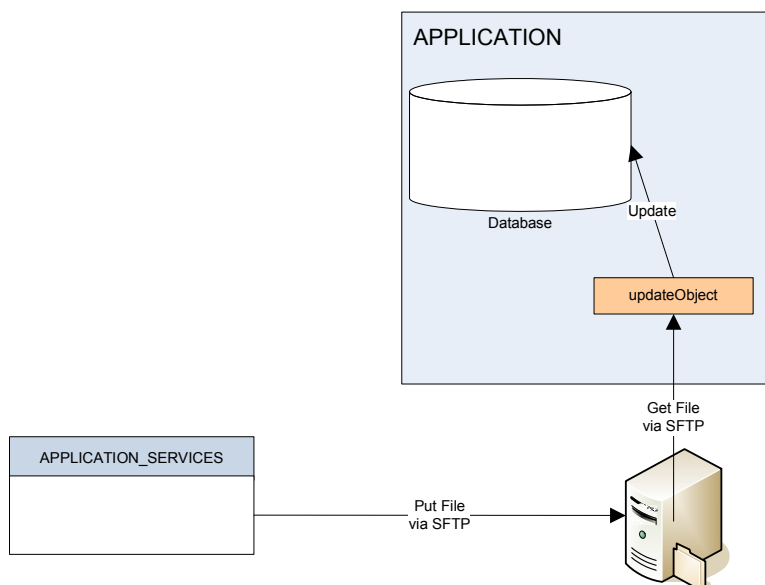


4.7 File-based Integration

File transfer between service layers and target applications will be performed either using SFTP or XCOM (depending on the target application, XCOM is typically used for Mainframe-based file transfer).

4.8 Pattern Variant – SFTP

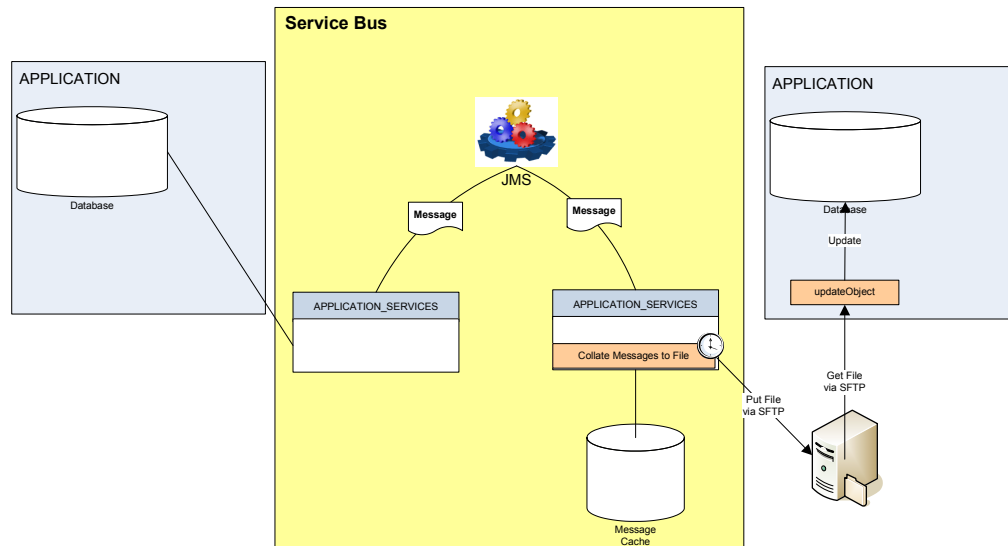
In this variation, the target system will expose an FTP file store to which a file can be SFTPed from the service bus. This interaction will not employ the guaranteed delivery mechanism via JMS used by the service bus.



In this scenario, the originating file typically exists on the ESB file store, and is retrieved by the bus for putting onto the application's SFTP store. The SFTP get from the application would typically be scheduled for polling on a regular interval or at a specific time of day.

4.9 Pattern Variant – Store and Forward

In this variation, the target system requires a file delivered using a file transfer mechanism, however the information is consumed in a message format. In this situation, the target service layer will store messages in a storage location (either disk or database), and perform a scheduled batching of messages into the target file format before delivering the file to the target application.



Messages will be received by the target service layer throughout the day (e.g. change notifications). These will be persisted by the service layer in a 'Message Cache'. This cache will be flushed and the messages collated into a file by a scheduled service, which will then deliver the file to the target system via SFTP, for example.

4.10 Pattern Variant – XCOM

CA-XCOM supports high-speed transfers of files between all supported systems. Users can send files from the local system to remote systems across an SNA or TCP/IP network and actively retrieve files from those systems. The remote system can initiate the same range of transfer capabilities as the local system.

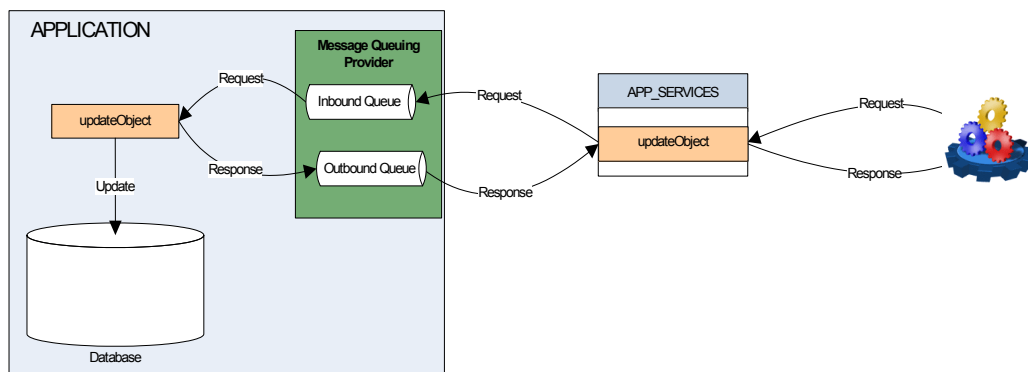
1. The local Windows user invokes CA-XCOM and requests a transfer.
2. CA-XCOM verifies the information contained in the request. Example: CA-XCOM checks to see if the local file exists on the local system.
3. CA-XCOM establishes a connection with the remote system and sends along the header record to the remote system. The header record contains transfer information regarding the request, for example, filenames and compression. The local system then waits for confirmation of the header record information.
4. Once the contents of the header record are verified, CA-XCOM begins sending data records to the remote system. CA-XCOM sends each data record, checking to see if any sending errors occur. If no errors are found, the local system checks for an end-of-file marker denoting the last record of the file. If no end-of-file marker is found, another data record is sent.

5. If an end-of-file marker is found, the local system sends a trailer record, which indicates the number of data records sent. The local system then awaits verification from the remote system that the number of records received is equal to the number of records sent. If the number of records matches, a successful send file message is sent to the user on the local system and the conversation ends. If notification was specified, CA-XCOM on the remote system notifies the user of a successful transfer.

XCOM will be the preferred file-based delivery mechanism for Windows-based applications. There is no provision for an SFTP client on such environments.
TO BE COMPLETED

4.11 Message-based Integration

Message-based integration with external parties and applications will typically be via an industry standard protocol or solution (e.g. JMS, MQ-Series, etc.)

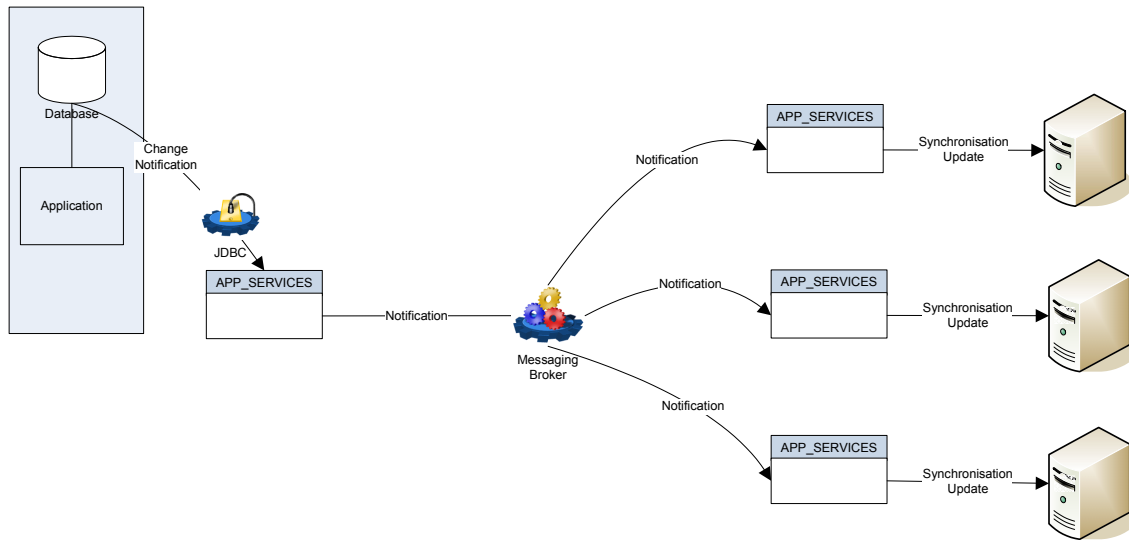


Typically requests and responses will be send in an asynchronous fashion, so if responses are required, a listener would exist in the APP_SERVICES layer to consume the response. This type of interaction lends itself to orchestrated business processes that can react to the successful processing of a task.

4.12 Typical Broker Notification Pattern

Below is a diagram representing a typical broker notification pattern. The notification is triggered from a database change (e.g. insert, update or deletion of a database table row). This notification event is consumed by the associated application service layer.

Some transformation may take place, and the notification event is published to the bus. There may be multiple consumers of the notification event, each services layer converting the notification to a synchronisation update on the back-end applications.



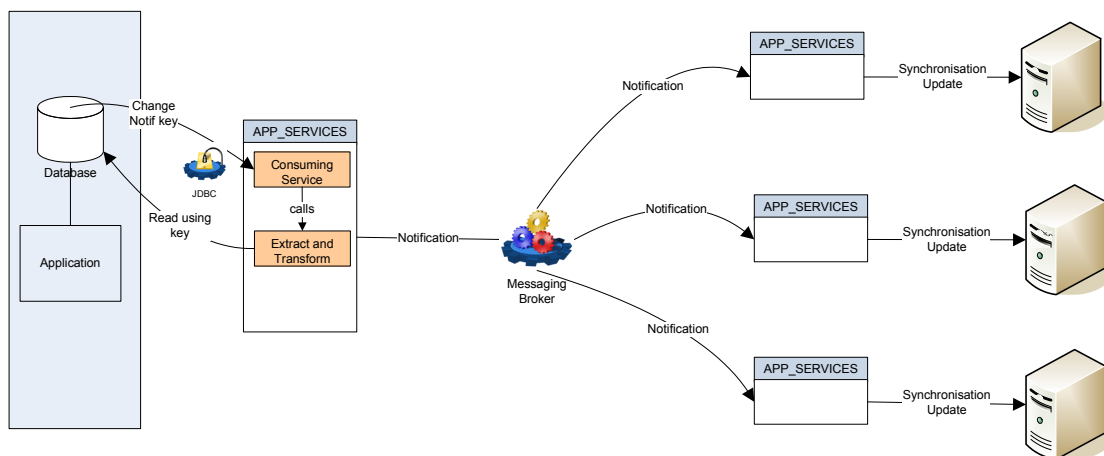
4.13 Full, Delta and Key Notification Variants

The content of the notification from the database can vary:

Full – this will contain all fields that represent the row that has changed

Delta – this will contain only the field values that have changed

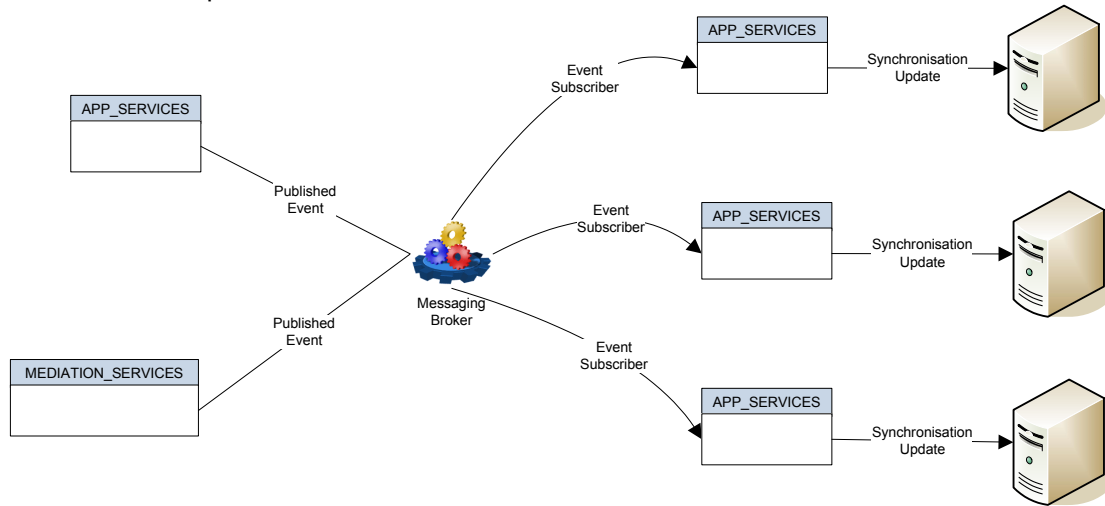
Key – this will contain only the key of the row that has changed, and a further DB read is required to retrieve the necessary data (for example, see below).



Logic resides in the Application Services layer to take the application notification and convert it to a notification event to publish to the bus. This notification event may be many depending on the logic in the services layer. Subscription may be via event type or content-based.

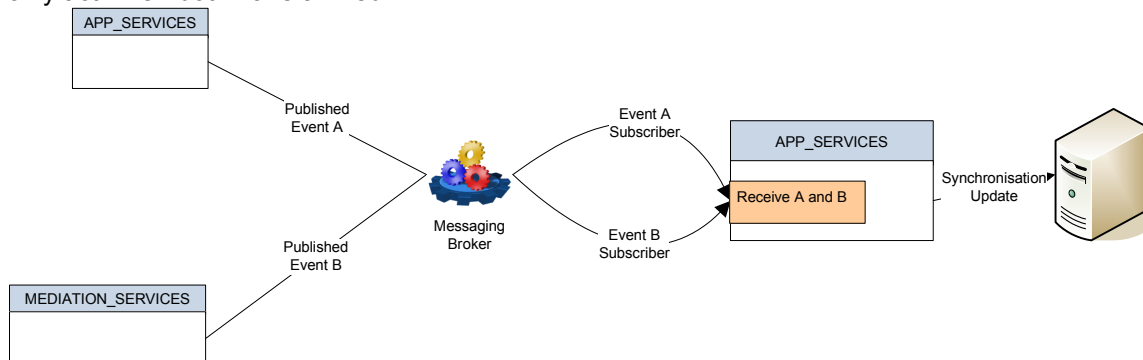
4.14 Asynchronous Communication Between Layers

This pattern follows the 'fire and forget' approach to publishing of documents to a messaging broker. It is assumed that the messaging broker will guarantee persistence and guaranteed delivery. There may be multiple subscribers to a published event.



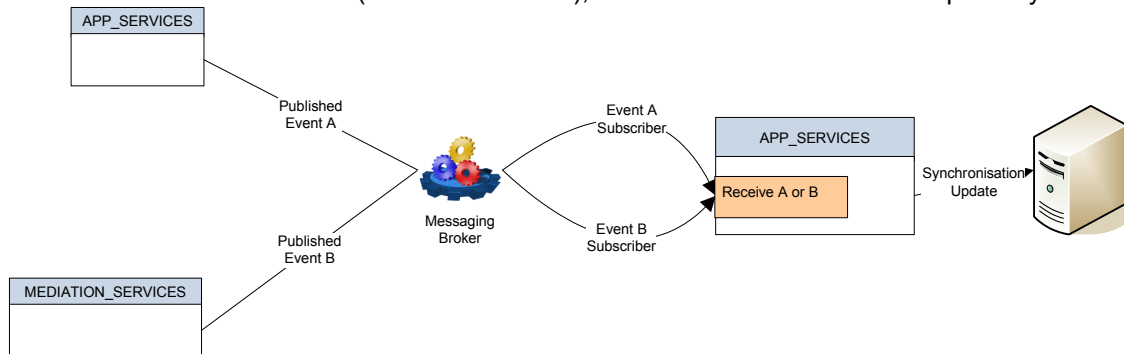
4.15 Pattern Variant – AND JOIN at Consumer

In this variant of the asynchronous pattern, there are two or more events published that require logically joining at the target. The target system consumes two related events (correlated in some way), and will only act when both have arrived.



4.16 Pattern Variant – OR JOIN at Consumer

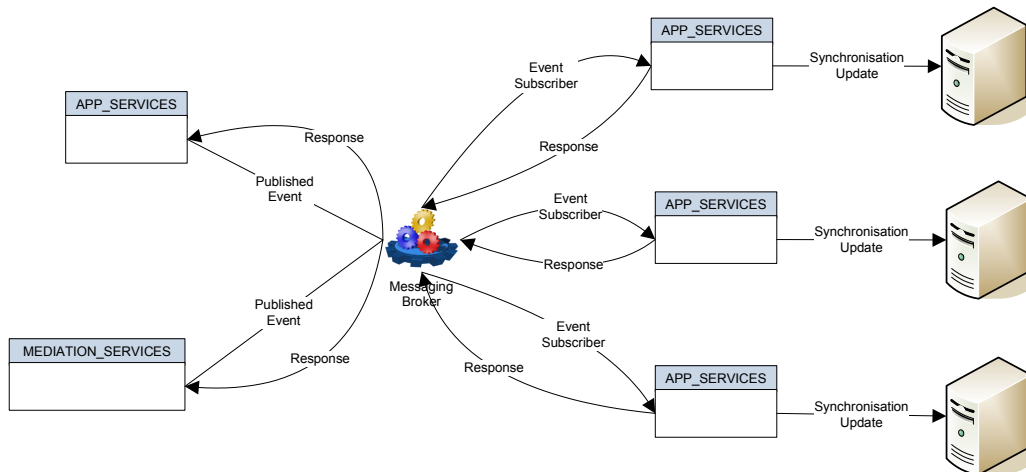
In this variant of the asynchronous pattern, there are two or more events. The target system consumes one of the two related events (whichever arrives), and will act on both events separately.



4.17 Pattern Variant – Asynchronous Response

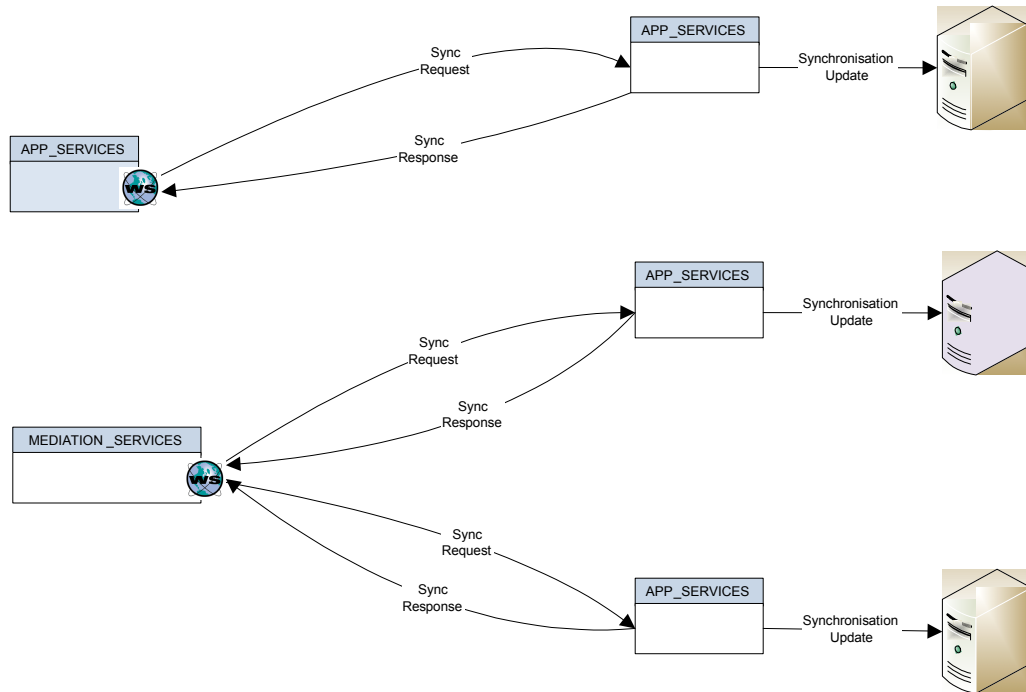
In this variation, the source service layer will expect an asynchronous response (ACK/NACK) from the target service layers, and will react accordingly. For example, this type of interaction is used within BPMS to allow steps in a business process to continue once an activity is completed, or react in an alternate way if the activity fails.

The response is not synchronous, so the source service layer needs to listen for response documents and process them.



4.18 Synchronous Communication Between Layers

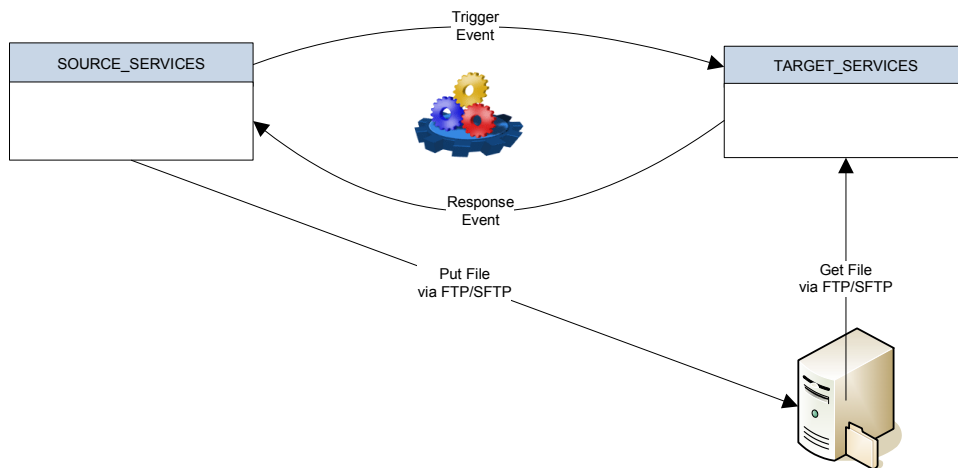
Synchronous interactions between service layers within the bus will use web-services. The request and response will be within the same synchronous web-service session. Services within the same layer can consume each other as required. The service bus will expose consumable services at the External, Mediation and Application layers.



4.19 File Transfer between Service Layers

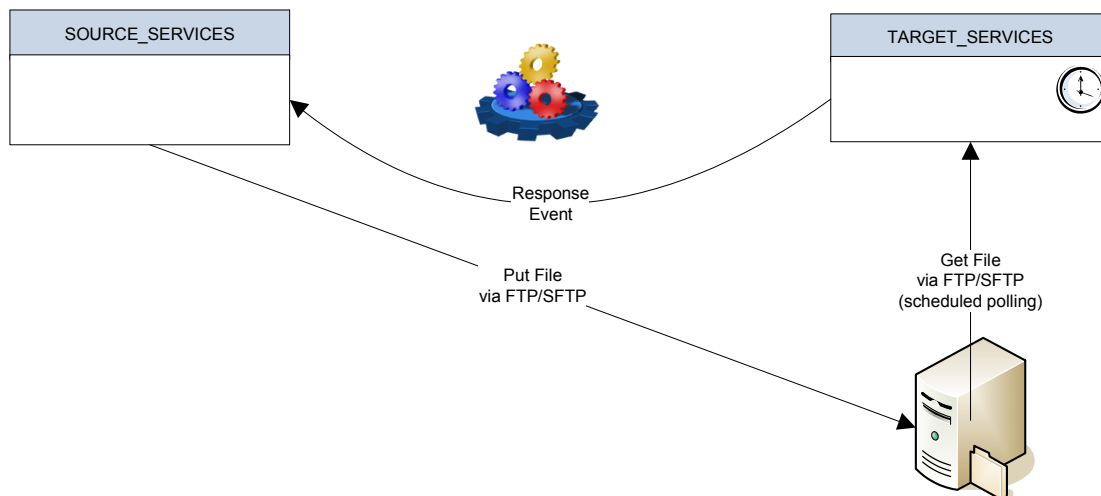
File transfer between service layers will incorporate a guaranteed delivery notification mechanism. The scenario is the file will be FTPed to the target, and then a trigger event will be published to the messaging broker to indicate that the FTP has completed. This will be consumed by the target service layer, and will trigger the receiving via FTP.

Once the receive has been processed successfully, a response trigger event will be published, for consumption by the source service layer to confirm the file has been processed at the target. This is an optional extension to this pattern.



4.20 Pattern Variant – FTP Polling at Target

In this variation, the target system does not respond to a trigger event from the source, but instead polls the FTP directories on a schedule, and then can optionally send an asynchronous response event.

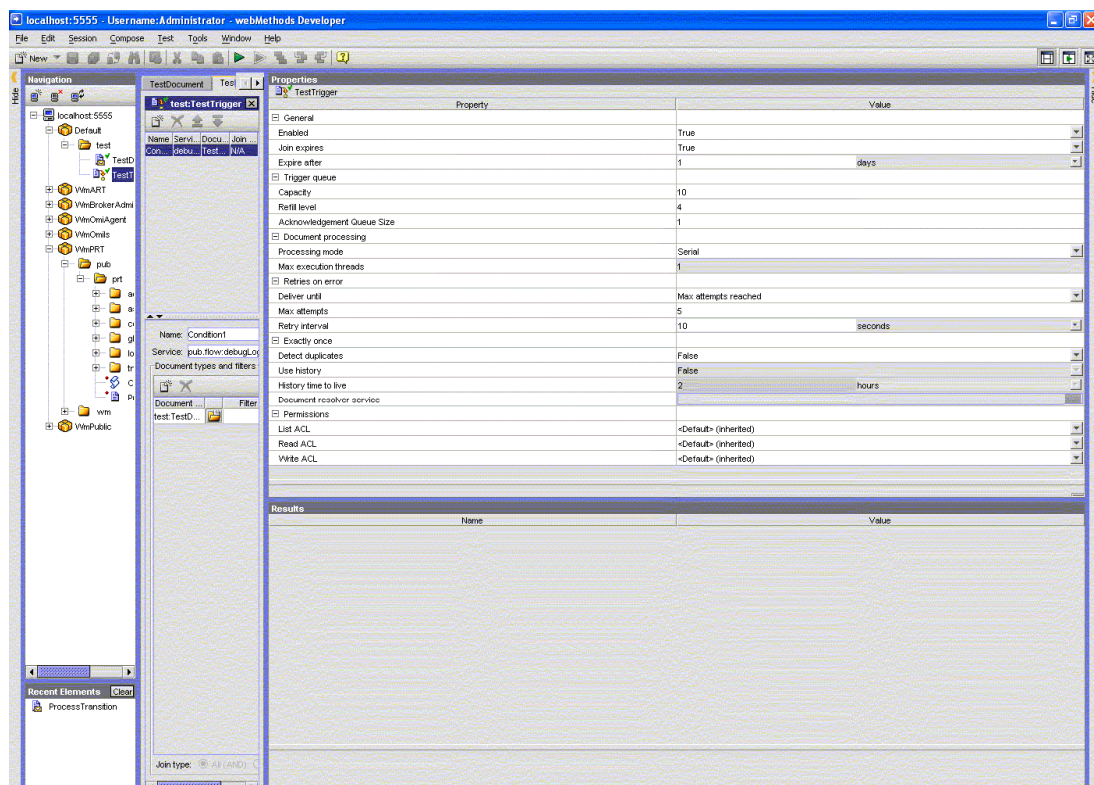


5 Performance Tuning

5.1 Trigger Tuning

Trigger Tuning. What are the different parameters and what will we use them for?

- **Trigger Queue:** This is the “read-ahead” queue (in-memory from 6.1) that consists of Publishable Documents that have been retrieved from the Broker, but not processed (the associated service has not been executed) or acknowledged by the Broker.
- **Capacity:** This is the upper limit of the number of documents that we have “read-ahead”.
- **Refill Level:** Once the number of documents reaches this level, the server starts retrieving documents from the broker again.



- If the time taken to process a document is much less compared to the time taken to retrieve it from the Broker, then the capacity level should be increased.
- Keeping the refill level closer to the capacity ensures that there is always work to do for the trigger execution threads.
- If the time taken to process a document is much larger than the time taken to retrieve a document from the Broker, then increasing the capacity might not make too significant an impact.
- Memory considerations also have to be kept in mind while increasing the capacity level.

Acknowledgement Queue Size:

- **Pros:** Increasing this to a higher number significantly increases performance. After each guaranteed document is processed, it is acknowledged to the Broker. This involves a network operation and a disk operation on the Broker side. By batching acknowledgements up, we can improve performance greatly.
- **Cons:** If the IS crashes, all documents that have not been acknowledgement will be redelivered by the Broker. This can lead to potential duplicate execution unless care is taken to check for duplicates with the Exactly Once settings.

Processing Mode:

- **Serial:** Use this mode if you want only one thread on one IS processing the document in order to ensure in order processing of documents. Even if the IS is in a cluster, only IS will be executing Documents published from the same Publisher. This corresponds to “Ordered by Publisher” for the Broker Client queue on the Broker.
- **Concurrent:** In this mode as many threads as specified in the “Max Execution Threads” settings can concurrently process the document. If the IS is in a cluster, then multiple IS might be processing the Document concurrently. Increasing the concurrency should help with the throughput, but this follows the law of diminishing returns depending upon the system resources available for the IS. This corresponds to “Ordering mode of None” for the Broker Client Queue.

Note: “Ordering mode of None” on the Broker is significantly faster for the Broker to handle.

5.2 Trading Networks Run Time Performance Tuning Tips

- Overall performance for Trading Networks is very tightly bound to the RDBMS performance. In nearly every test case, the database became the bottleneck. Sometimes the disk I/O is maxxed out. In other instances, the CPU in the database server becomes the bottleneck. To optimize Trading Networks, focus first on database optimization.
- Optimize connections to the Trading Networks database when using an external database. By default, the maximum size of the connection pool to the Trading Networks database is 10. If the backend database can handle more concurrent connections, you may be able to improve performance by increasing this parameter. You can configure this from the JDBC Pools page on the Integration Server Administrator.
- Adjust the value of the `tn.task.sweepTime` property, located in the `IntegrationServer_directory/packages/WmTN/config/properties.cnf` file. The `sweepTime` property measures how many seconds can elapse before Trading Networks looks for more jobs once it has cleared its backlog. If a backlog remains, then Trading Networks will continue to attempt to run delivery services for jobs. Do not set this value too low because Trading Networks will not stand by waiting for work to perform during idle periods. Instead, the Trading Networks Job Manager will be scheduled to run frequently at the interval you have provided, potentially wasting processing time. Too large a value can result in many tasks backing up awaiting delivery or routing. These tasks then must be handled in one large iteration. It can be more efficient to work with small iterations over a few jobs instead of large iterations over many jobs.
- Adjust the value of the `watt.net.timeout` property in the `server.cnf` file. Set this parameter to a number of seconds greater than zero. This is the amount of time connection establishment will wait before timing out. Setting this value prevents a known problem with Trading Networks outbound jobs not processing when two jobs are delivering to a host that is not responding.
- For internal Trading Networks document submissions, use `wm.tn.doc.route:routeXML` instead of `wm.tn:receive`. This will bypass the user check and is therefore slightly faster.

- Within Processing Rules, execute Flow services in asynchronous mode where appropriate or practical.
- Another parameter that can be adjusted is the percentage of the Integration Server threads to allocate for Trading Networks Task Manager. This is a real value that must be between 0 and 1. Set this value higher on a dedicated system. `tn.task.threadpool.pct=[0..1]`

5.3 How can I make best use of “watt” properties ?

- a. **watt.server.broker.producer.multiclient**: The IS uses the Default Broker client to publish to the Broker. If this property is set, then the IS creates a set of broker clients and one is used from it. Using multiple broker clients can help with ensuring greater concurrency and hence help performance under certain conditions, but there is a greater overhead in having them. Also the broker overhead increases as the number of clients connected with it increase.
- b. **watt.server.broker.replyConsumer.multiclient**: If this property is set, then the IS uses a set of Broker Clients to do the request reply operations.
- c. **watt.server.dispatcher.comms.connectionShareLimit**: This property decides how many broker clients share a socket connection to the Broker. Decreasing the sharing might help with throughput, but it will increase the number of resources and file descriptors used by the IS. A value of zero means that connection sharing is turned off and each broker client has its own socket to the Broker. By default this setting is 5.
- d. **watt.server.control.maxPersist**: This is the maximum number of documents that the user is allowed to persist in the outbound store while the broker connection is down. When this limit is hit, the publishing thread is blocked until the broker connection becomes available again. By default, this setting is **50000**.
- e. **watt.server.control.maxPublishOnSuccess**: This controls the maximum number of documents that can be stored in memory if the `publishOnSuccess` flag is set to “true” during publishing. This is the cumulative maximum, i.e. if there are say two services, both publishing with the flag set true, then this flag controls the maximum that can be stored in memory for both the services. Default is **10000**.
- f. **watt.server.control.triggerInputControl.delayIncrementInterval** and **watt.server.control.triggerInputControl.delays**: These properties in conjunction control the polling mechanism of trigger queues on the broker. In order to conserve system resources, the each trigger’s queue on the Broker is scanned according to these rules:

If the call to `getEvents` returns any events from the broker, then the queue is immediately scheduled for retrieval till we hit the capacity.

If the call to `getEvents` does not return any event, then this trigger is scheduled for Document Retrieval according to this algorithm:

Say the delays are [1000,2000,3000,4000], and the **delayIncrementInterval** is 1000, then the document retrieval will be scheduled like this:

$0/1000 = 0$, so we would take index zero of the array and we will sleep for 1000 msecs
 $(1000 + \text{scheduling overhead})/1000 = 1$ (we round to integer so it will be at least 1), so take index 1, which would be 2000 msecs

Note: The largest entry in the `watt.server.control.triggerInputControl.delays` is the longest lag between the document being stored on the Broker queue and the document being retrieved by the IS.

6 Best Practices - Developer

6.1 Pipeline Management

All services should clean up all variables in the pipeline that are not in either the input or output specification.

- Drop pipeline variables and documents as soon as they are no longer needed in the flow. This minimizes the Integration Server's memory consumption.
- When you call a service within a flow, remember that the variables created in the service are carried forward from the point of execution of that service. Also remember that input parameters of services that are set or mapped (except for transformers) enter the pipeline. This can cause problems where multiple services with the same inputs/outputs names are called in succession or with inside loops that call the same service multiple times. Therefore, drop unnecessary items immediately after calling a service.
- *pub.flow:clearPipeline* service can be used when necessary but not compulsory as a last step in the flow. Guideline is to drop the variables when not required.
- Invoke *pub.flow:clearPipeline* at the end of audited services to guarantee that only output variables are stored (use the "preserve" variable to enumerate those output parameters).
- When possible, use the same input and output parameter names in a sub-service definition as those from the caller service. Doing so prevents unnecessary objects from being copied and reduces the necessary variable drops.
- Disabled steps in flow services still consume CPU. Before going to production, remove all disabled steps from flow services.
- Use the Integration Server caching mechanism for the services that return static data (data that don't change during a given period of time) and that are called frequently. You should call cached services only through a transformer. Doing so guarantees that the pipeline used to make the comparison only contains the input parameters, that the size is minimal, and that the returned pipeline won't overwrite the caller pipeline. End cached services written in Flow with a call to *pub.flow:clearPipeline* so that only output parameters are cached.
- The Integration Server is flexible with pipeline contents at run time, assuming explicit validation is not used. Run-time pipelines can be different from the design-time pipeline, as viewed in the Developer, which leads to unexpected results and exceptions. For example, if we are mapping a document instance called "*customerInfo*" at design time that has a specific structure – *firstName* (String) and *lastName* (String), the map still attempts to execute *customerInfo* even if it is populated with an entirely different structure at run time – say *age* (String), etc.
- A default value for service input can be implemented by clearing the "*Override pipeline value*" option in the Set Value operation.

- It is possible to have multiple copies of a pipeline variable with the same name. This needs to be avoided, as it can cause confusion during mapping.
- Reducing pipeline size is a great way to increase performance. Not only will this require less information to be stored in memory, but when services are called, less data will need to be copied over. To accomplish this result, it is best to build out the entire service, and then walk up the flow from the bottom dropping variables as early as possible. This will help ensure that required data is not dropped too early.
- In webMethods there are two different ways to invoke a service. The first way is to call the service normally as an invoke step, the second is to use the service as a transformer in a map step. When using a service as a transformer, it only passes a reference of the mapped inputs to the called service, not the entire pipeline. This limits the amount of data passed and will speed up performance. Utility services should be used as a transformer service.
- Remove all commented out code and *debugLog* calls before migrating to a production environment. This extra code uses up memory and can affect performance. Also ensure that there are no save and/or restore services turned on. Saving pipelines to files or memory is a waste of time in a production environment, and restoring a pipeline severely increases the likelihood of error.

6.2 Documents

- Use document type references when using document types within Developer for mapping input/output and so on. This reduces mapping effort and improves Developer performance.
- Never reference a non-publishable document within a publishable document. Referencing a non-publishable document within a publishable document will lead to synchronization issues when more than one Integration Servers are involved.
- Always mark the “*Allow unspecified fields*” property of a publishable document as “*false*”. This ensures that the document is validated before getting published to the Broker.
- Divide complex document definitions into sub-documents to maximize reusability and flexibility.
- Avoid usage of temporary documents in the service. Create documents and reference them wherever they are used. This will ensure that the change to the document is done once and the same is reflected everywhere.
- The publishable documents should not contain object types as fields since Objects will not be unmarshaled by the subscribing service on another IS.

6.3 XML Strings and Documents

- When using `pub.xml:xmlNodeToDocument`, set `makeArrays="false"` and `documentTypeName=<input document type name>`. This ensures that the document that is output from the service has a structure consistent with `documentTypeName`.
- When using `pub.xml:documentToXMLString`, set `documentTypeName=<input document type name>`. By default, webMethods record fields are ordered by mapping order. Referencing the `documentTypeName` during the `pub.xml:documentToXMLString` call ensures that the tags of the output string are in the correct order.
- When using the `pub.xml:xmlStringToXMLNode` service, the `isXML` variable must be set to “*true*” for the Integration Server to parse the document correctly.

6.4 Flow Services & Java Services

- In general, try to use Flow Services instead of Java services, but use your best judgment when deciding. Writing flow services generally makes application logic more readable and easier to debug, modify, and extend for future needs. However, there are cases where the complexity of service logic suits itself to Java rather than flow. For example, if a service contains many nested loops, it may make sense to go with Java. Also keep in mind that a Java service, if coded well, will generally perform better than a flow service with the same functionality.
- Always specify the inputs and outputs for all services so that these will be exposed to a calling service. Scheduled and start up services are exceptional cases.

- It is good practice to ensure that custom-developed code is operating system-independent. For example, don't use “\” as a file separator. Instead, either use “/” (which works on both Windows and UNIX operating systems) or use the system property named ***file.separator***
- Be careful to close all file descriptors, input/output streams, and so on using the Java finally clause. This will ensure that objects are cleaned up even if an exception occurs.
- Enable resubmission capabilities (audit settings) only to top level services.

6.5 Input Validation

- When developing Java services, always check the “Validate input” box on the Input/Output tab. More often than not, the inputs to Java services are required and checking for their existence up front will allow you assume their presence within the service, resulting in more efficient code. Otherwise, the presence of each parameter will need to be validated and an error thrown if the variable is not found.
- Integration Server will take care of throwing the exception for the missing parameter when it validates the input. An error, similar to “Input validation for server failed: /parm1 W-001 Missing Object” will be thrown. Enabling input validation means it is important to mark optional parameters as such; otherwise erroneous errors will be thrown. The presence of these optional parameters will need to be verified by the service itself within if then structure similar to the first example above.

6.6 Publish-Subscribe model

- Use lexical expressions on trigger filters when possible. Lexical expressions are evaluated at the Broker level rather than the Integration Server level.
- Define all Integration Server publishes as non-local. Let the architecture (physical installation) drive how messages are sent.
- To configure guaranteed delivery, both the document type and the client group must be set to guaranteed delivery.
- Always specify a timeout value, appropriate to the documents time-to-live property, for request/reply (*publishAndWait*) services.
- Always mark the “*Allow unspecified fields*” property of a publishable document as “*false*”. This ensures that the document is validated before getting published to the Broker.
- Never reference a non-publishable document within a publishable document. Referencing a non-publishable document within a publishable document will lead to synchronization issues when more than one Integration Servers are involved.
- In case of IS cluster make sure that triggers are not modified in cluster mode. All the modifications to the trigger should be done in stand alone server and deployed to cluster instead of modifying them on cluster itself.

- Enable duplicate document check only when required
- A trigger can invoke to at-most one service. However, filters can be set such that it triggers one of the many services it is supposed to invoke. It should be ensured that the filter conditions are mutually exclusive.

6.7 General / Programming Guidelines

- Any parameters that are hard-coded within flow and will not change with environment must be documented in comments section of the flow.
- It is possible to accidentally move services (within a flow or between folders) in Developer without being prompted to confirm the move. Be careful with your mouse clicks when using Developer.
- Before making updates to a service, first right-click on the service and select "*Find Dependents*" to determine where it is used. This will show you all the places where the service is invoked so that you can be sure you will not hurt anything else when you make changes.
- Keep in mind that the Integration Server ships with many useful built-in services, which are documented in the webMethods Integration Server Built-In Services Reference guide located under <DEVELOPER_DIR>/doc. It is better to check if a service for the needed operation already exists rather than creating a new service for same functionality.
- Never change the contents of any webMethods-supplied package. These packages can be distinguished by the "Wm" prefix in the package name or by a root folder of "wm".
- Do not use or reference any of the webMethods sample services (such as those supplied in the *WmSamples*, *WmTNSamples*, *WmEDISamples*, or *PSUtilities* packages) directly. If these services are to be used, copy them to the **Common** package since these services change from release to release. These services are beta services and need to be tested thoroughly before using them.
- *\$default* is not needed in BRANCH statement if no logic at all. It is kind of like an "if-else" statement without "else" logic.
- Use database operations in jdbc adapter services as much as possible to perform conversion tasks so that middleware can gain as much performance as possible.
 - Eg. See example in "xyz.db:updateWithSQL"
 - Do not use "*pub.date:getCurrentDateString*" to get current string to pass to database. Use Sysdate in adapter service unless this date needs to be kept in the beginning of the logic.
 - Do not use "*pub.string:toLower*" or like function to convert field before passing to database. Rather, use database operation like "lower" in adapter service.
- The Integration Server caches certain package components when it loads a package. If any change has been made that is not reflected in Developer, this might reload the relevant

package(s). For example, when changing values in the properties file the package must be reloaded.

- Use appropriately versioned Developer client while connecting to the corresponding Integration Server. Some of the features (e.g. configuring retries for services) may not be available when there is a version mismatch.

Note: If there is a separate process wide exception handler service which the PRT would invoke automatically on an error (in case of Process Models designed using wM Modeler) the try/catch sequence can be avoided.

```

graph TD
    S1[SEQUENCE1 (main sequence, set to exit on SUCCESS)] --> S2[SEQUENCE2 (try sequence, set to exit on FAILURE)]
    S2 --> S1_1[Service1]
    S2 --> S1_2[Service2]
    S2 --> S1_3[...]
    S2 --> S3[SEQUENCE3 (catch sequence, set to exit on DONE or as required)]
    S3 --> S3_1[pub.flow:getLastError]

```

Following are best practices with regard to Trading Networks profiles, document types, and processing rules.

7.1 Trading Networks Profiles

Creating new extended fields does not retroactively apply to profiles already created. Make sure you define all extended fields before creating your profiles.

7.1.1.1 Document Types

1. You can bypass document recognition by supplying the document type name or ID in the pipeline at `/TN_parms/DoctypeID` or `/TN_parms/DoctypeName`. This causes Trading Networks to skip the recognition process and directly select the specified *doctype*.
2. Document types cannot be ambiguous. If a particular document submitted to Trading Networks matches more than one document type, Trading Networks neither selects a processing rule nor processes the document. Trading Networks considers the document an unknown document and logs an error.
3. If Trading Networks recognizes a document, the *createEnvelope* method on the *BizDocType* is called. This method creates and formats the *BizDocEnvelope*. After document recognition completes, a *bizdoc* is written to the pipeline that contains its *BizDocType*.
4. To identify an XML document, you can specify XQL identifying queries. When you do so using the Trading Networks Console, you can specify the query and its value. If you do not specify a value for the query, Trading Networks checks only the existence of the XML node. If you specify the value as a single fixed string, the query is successful only if the element contains that value in the document. Consider a case where you want to have a set of possible values instead of a single value. For example, if your *DocumentType* element can have only two values—PO or BO—you would keep the Value part blank and specify the XQL query as follows:

```
//PO/Header[DocumentType='PO' $or$ DocumentType='BO']
```

7.1.1.2 Processing Rules

1. Although you can create processing rules with the same name, use different names to avoid confusion and redundancy.
2. Trading Networks selects processing rules using a top-down approach. Be sure to document the processing rule order so that during migration or accidental alteration, the rules can be reorganized for the proper rule selection.
3. Trading Networks evaluates the matching criteria on each rule in order against the “bizdoc” until it finds the first matching rule. Once found, Trading Networks executes the rule. The matching criteria can be any of the following:
 - ❖ Sender
 - ❖ Receiver
 - ❖ DocType
 - ❖ User status
 - ❖ Presence of recognition errors
 - ❖ System and user attributes
4. Trading Networks 6.5 adds support for matching the sender and receiver on profile groups. This means you can create processing rules for groups of profiles. When you create a new partner, you can add the partner to one or more groups. Trading Networks automatically matches documents to and from that partner to rules for those profile groups.
5. Trading Networks 6.5 also adds the ability to bypass rule matching. If you know the rule you want to use, you can specify it in the pipeline as */TN_parms/processingRuleID* or */TN_parms/processingRuleName*. If you want to bypass routing altogether, the best way is to specify a *ruleID* or *ruleName* of “Default rule” or some other default processing rule you have created.

6. You can configure processing rules to execute services synchronously, asynchronously, or reliably:
 - *Synchronous execution* performs a *Service.doInvoke*, wrapped with some extra error handling and logging.
 - *Asynchronous execution* is more followed way of doing threaded service invocations.
 - *Reliable execution* creates a task and passes it to the Job Manager.

8 TIPS

8.1 Tip on Publishing event and dimensional data to Optimize

Version : Applicable to both webMethods 7.x and 8.x versions.

Scenario: Publishing event and dimensional data using the webservice data collector's dimensionalData interface.

Standard Procedure: The recommended method for publishing monitor data into Optimize is to use the Web Service Data Collector's dimensionalData interface.

The Web Service Data Collector contains web services for sending data as events and dimensions, where events capture metric data and dimensions capture metadata about the metric data. addEvents method is used to publish the event data in to Optimize.

Pain Point: If we follow the standard recommended procedure to publish the data i.e. creating the webservice connectors using the WSDimensionalDataCollector.wsdl and use the addEvents connector to send the data to Optimize . Here we will be able to create the corresponding event map, facts, dimensions and kpi's through MWS Optimize interface. But the data is not populated in to BAM_AGG table for that event map which is responsible for generating KPI monitors in MWS (monitoring section/Business/BusinessOverview) where the actual visualization of the data happens.

Alternate Approach: Instead we can use **pub.optimize.monitoring:addEvents** service directly to publish the data. Here the BAM_AGG table is also populated for that event map and corresponding kpi monitors are created on Business overview page (monitoring/Business/BusinessOverview).

8.2 Tip on Duplicate documents getting subscribed by JMS trigger.

Scenario when the issue occurred:

Method – JMS Publish/Subscribe

Broker Cluster Policy – Multi send Best Effort (Documents will be published to all the available brokers in the cluster. Once it receives the 1st acknowledgement for the message, it would send message to rest of the brokers in the cluster to discard the message)

Trigger Properties – Detect duplicate = True, Use History – True and History time to live = 15 minutes

Description of the issue:

The JMS publish, publishes the document to all the available brokers in the cluster and waits for the acknowledgement. Sometimes it might happen that after receiving the acknowledgement from the 1st broker, it would miss to communicate to the other brokers to discard the message. And so after few hours when the 2nd broker sends the acknowledgement, the document is subscribed again and creates duplicate document (In our case the 2nd subscription happens at 10 hrs time interval).

This will end up sending duplicate documents to the target system.

To Resolve:

Below properties were added in the **awbroker.cfg** under the Brokerdatastore folder to resolve the above issue.

```
pre-acknowledgement-timer-interval=40
tentative-pre-acknowledgement-timeout=80
confirmed-pre-acknowledgement-timeout=7200
```

On adding the above properties, if the broker doesn't send the acknowledgement in 7200 seconds (2 hrs), then it would discard the messages from rest of the brokers in case the communication between the brokers is missed.

8.3 Tip on Importing a profile in Trading Networks.

Problem Scenario: Importing a profile in Trading Networks.

Problem: Error is displayed while importing a profile having profile extended fields under the field group name "EDIINT" - AS1MDNNURL, AS2MDNNURL, Encryption Algorithm, S/Mime Type

Reason of error: When Partner data has a dependency on any built-in piece of TN data (e.g. the 'S/MIME type' and 'AS1MDNNURL' Field Definitions), the customer will never be able to deploy that data. Because the identifier of the built-in data is different on the source and target machines, the TN APIs are unable to figure out what to do.

Solution: The above error can be rectified by importing only the 'PartnerData' during the import (excluding the field definitions), and then manually populating the values for the profile extended fields once profile was created/imported.

8.4 Tip on Retry Mechanism

Problem Scenario: Sometimes a webMethods service fails due to network issues or failure to connect to a database. The service might execute successfully if the Integration Server waits and then retries the service.

Solution: Use the `pub.flow: throwExceptionForRetry` service to handle transient errors that might occur during service execution. A transient error is an error that arises from a condition that might be resolved quickly, such as the unavailability of a resource due to network issues or failure to connect to a database. If a transient error occurs, the service can catch this error and invoke `pub.flow: throwExceptionForRetry` to instruct the Integration Server to retry the service.

The `pub.flow:throwExceptionForRetry` service should be used for transient errors only. Only top-level services or trigger services can be retried. That is, a service can be retried only when it is invoked directly by a client request or by a trigger. The service cannot be retried when it is invoked by another service (that is, when it is a nested service).

We can invoke the `pub.flow:getRetryCount` service to retrieve the current retry count and the maximum specified retry attempts for a service.

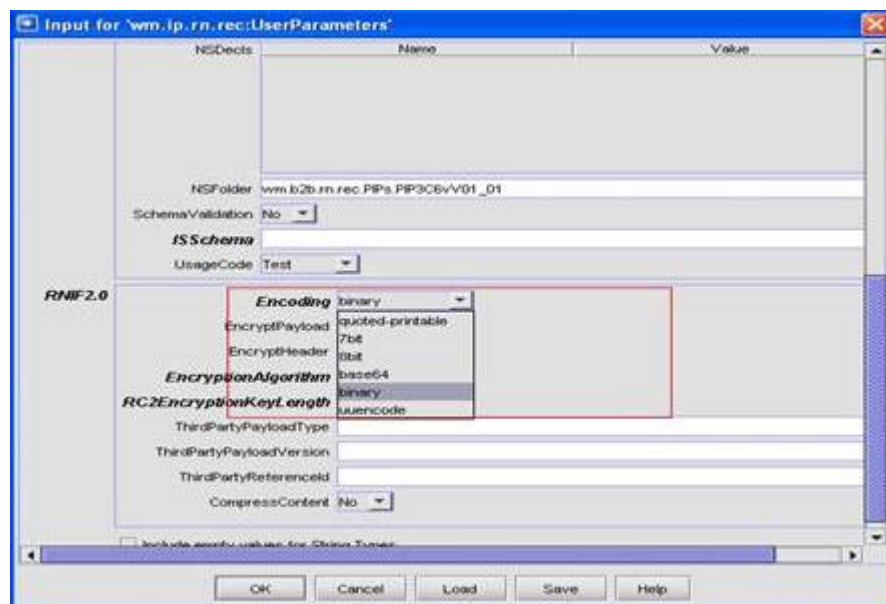
8.5 Tip on webMethods rosettanet

Problem Scenario: webMethods rosettanet module generates PIP message which is sent to the partner. The PIP message can have below issues along with several others.

- 1) In the xml header, a string “3D” is added before the version and the encoding format. Because of this the message will not be recognized by any xml parser tools.
- 2) Second issue is when the length of a line exceeds certain number of characters, the line is truncated with a “=” sign and the remaining text is continued in the next line

Reason: The reason can be wrong value set to encoding parameter (RNIF2.0 -> Encoding) in the PIP specific TPA.

Solution: This can be resolved by changing the RNIF2.0 -> Encoding parameter to binary. Attached is the screenshot for more clarity



8.6 Tip on XML Schema Validation

Problem Scenario:

To create a validation schema given an input XML string

Solution:

1. Create a Document Type "TestDoc" in Developer based on the layout.xsd -- Root Element
This will create a schema schema_TestDoc by webMethods
2. Create a Flow Service as below :
Convert the given input XML string to node using `xmlStringToXMLNode <-` passing in the XML as `xmlData`
Set the `isXML` type to true : `<- isXML` set to true.
3. Convert the `xmlNode` to document using `xmlNodeToDocument ->` output document to DocumentType "TestDoc".
4. Finally, Invoke `pub.schema:validate <-` passing in "TestDoc" to objects `<-` set `confirmsTo` to the documentType definition "TestDoc"

Validation: For this, set `ignoreContent = "true"` and validating it against the document Type would ensure that there are no errors

8.7 Tip on webService in C#

SCENARIO: To invoke a Web Service in C# that uses soap/rpc in webMethods

Problem/error generated : While invoking the service from .net into webMethods, if not called correctly an error is received 'Unable to process request as the object reference is not set properly.'

SOLUTION: We can generate a WSDL for the C# service and use the Web Service Connector wizard to generate a Flow service.

There are built-in services to work with soap headers. The service is invoked by using them in the following manner.

1. Create a document type from the element definition
2. Populate the document and convert it to a node.
3. Then use `pub.soap.utils:addHeaderEntry` and add it to the header of a soap message created using `pub.soap.utils:createSoapData`.

Generated Flow

4. Populate the input document, convert it to a string then a node and add it to the soap body
5. You will get a `soapResponseData` object as an output of this service

The Flow service in webMethods is thus generated by the web services connector

8.8 Tip on Java Service

Problem: While working on java services in webMethods, a situation could arrive where the service is throwing a run time exception i.e., the code is successfully compiled, but errors are appearing while execution. Debugging this code could be a tedious job.

Solution: To solve this problem **System.out.println()** statements in the service can be set at various places in the code, acting like break points. The content passed in these commands will appear in the **nohup.out** file in the **<IntegrationServer>** folder of the webMethods software in the server. Thus, by setting these breakpoints and passing value in them at runtime, flow of the code can be evaluated and errors being raised at runtime can be debugged.

8.9 Tip on TN Convert To Values

Context: The significance of the **skipWhiteSpace** parameter in **convertToValues** service.

The **skipWhiteSpace** parameter is a string type parameter which is optional. It decides whether the user wants the white space to be trimmed from the beginning of the records. This parameter is ignored by the fixed length record parser. It can have the following values:

Value of skipWhiteSpace	Meaning
true	Trim white spaces at the beginning of the record. This is default.
false	Record is used as it's identified. It is useful for positional data record.

As the default value of this parameter is **true**, so when we provide a flat file as an input the **convertToValues** service having a space at the beginning of the record the service will trim the white space and treat the subsequent value as the record identifier.

If the value of this parameter is set to false, then the developer will not trim

8.10 Tip on EDI X12 Document

Scenario: Installing an EDI X12 document in both the nodes of IS.

Problem: Error is displayed after installing in the first node of IS.

Reason of error: Servers which are clustered share the same TN DB.

Solution: The above error can be rectified in the following manner :

- 1) Install the EDI X12 document on Node1 from the IS admin page.
- 2) Using the Developer login to Node1, go to WmEDIforTN package and copy the EDIFFSchema.X12 folder from Node1 to Node2.

8.11 Tip on JDBC Adapter

Scenario: Using JDBC adapter in the Integration Server and then to run the stored procedure on the SQL Server2005 DB. Using a function here to convert date to char. This is a case of using Dynamic SQL in Stored Procedure.

Problem: Error is displayed in converting a string into a date data type.

Solution: The above error can be rectified in the following manner:

- 1) Change the "Input Field Type" of your DATE column from java.lang.String to java.util.Date in the adapter service
- 2) Convert the date string to a java.util.Date object prior to invoking the adapter service.

The way of converting the date to char in the database is given below.

Ex: select @SQL ='select * from Tablename where columnname ='convert (varchar,@Date) ' won't work and gives an error.

Instead select @SQL ='select * from Tablename where columnname =' + convert(varchar,@Date) + '' will work fine .

8.12 Tip on JDBC Adapter

Scenario: Inserting the date into Oracle database which has to be in date data type.

Problem : Error is displayed in converting a string into a date data type.

Solution: There are two solutions to the same:

- 1) We can use **pub.date:getCurrentDate** to get a **java.util.Date** object that represents the current time.
- 2) For getting the java.util.Date object that represents a time of your choosing, we can use **WmTransformationServices:StringToDate**.

8.13 Tip on Java Service Development Restriction

There is a restriction in developing java services. This restriction is that within a folder you cannot create java services and a sub folder that has more java services. Folder structure as below is not allowed.

```
Folder A
-> java Service1
-> java service2
-> Folder B
    -> java service3
```

-> java service4

The following is allowed:

Folder A

-> Folder B

-> java service1

-> java service2

This is also allowed:

Folder A

-> java Service1

-> java service2

-> Folder B

This is also allowed:

Folder A

-> java Service1

-> java service2

Folder B

-> java Service3

-> java service4

8.14 Tip on TRY CATCH BLOCK

Scenario:

The issue is when we use a TRY/CATCH block and handle error within the CATCH block (which is what is required), the Monitor shows that service as Completed (because error was handled). The auditing of that service is enabled. How to flag it as 'Failed' in Monitor?

Solution:

Add an **EXIT** from flow, signal FAILURE, as the last step of your CATCH sequence. This will set the status to **'failed'** in the Monitor.

Properties	
EXIT '\$flow'	
Property	Value
<input type="checkbox"/> General	
Comments	
Label	
Exit from	\$flow
Signal	FAILURE
Failure message	%lastError/error%
Exit from Specifies the flow step or service from which you want to exit. Select \$parent to exit from the nearest parent flow step. Select \$loop to exit from the nearest parent LOOP or REPEAT step. Select \$flow to exit from the entire flow service. To exit from the nearest ancestor flow step with a specific label, type the label of that ancestor flow step.	

8.15 Tip on Publishable Document Type

Scenario:

Two publishable document types are to be created (of the same structure) across different servers connected by the same broker and having common subscription clients.

Solution:

Create a document type on the first server and make it publishable "**true**" and note down the broker document type name of the same.

Make sure that broker document type of both the documents shares the same namespace. Now while creating the same document type on the other server, select "**All Options**" followed by "**Broker Document Type**".



It will give a list of all the broker document types present on the broker, select the required one from the list.

Any change made to the broker doc on one server instance has to be synched up with that on the other instance. This is done by performing "**Push to Broker**" from the first instance.



and "**Pull from Broker**" enabling the option of "**Overwrite existing elements ..** " from the other.



8.16 Tip on loop over a document

SCENARIO:

It is a very common situation where we loop over a document list and do processing and at the end of the processing the source document is appended to target document list using the **appendToDocumentList** service.

PROBLEM:

Using **appendToDocumentList** service in loop over a huge list can affect the performance by a great deal.

SOLUTION:

Vector approach can be chosen over appendToDocumentList. Vector services run much faster (upto x10).

Needed are two simple java services:

1. Service to append a document/documentList to a vector object for each iteration of the loop. This is to be executed multiple times (like in loops, repeats) until all the documents are appended.

INPUT:

- vector: new or existing vector that holds the items
- appendItem: item to append (i.e. String, Document)
- appendItemList: item list(array) to append (i.e String arrays, Document arrays)

OUTPUT:

- vector: vector with the items

```
IDataCursor idc = pipeline.getCursor();
Vector v = (Vector)IDDataUtil.get(idc, "vector");
Object o = IDDataUtil.get(idc, "appendItem");
Object [] oa = IDDataUtil.getObjectArray(idc, "appendItemList");
if (o != null){
    if (v == null) v = new Vector();
    v.add(o);}
}
if (oa != null){ // arrays
    if (v == null) v = new Vector();
    for (int i=0; i< oa.length; i++){
        v.add(oa[i]);
    }
}
if (v != null) IDDataUtil.put(idc, "vector", v);
idc.destroy();
```

2. Service to convert vector to the documentList. Need to use this service to convert the vector back to the proper objects.

INPUT:

- vector: the vector

OUTPUT:

- DocumentList: document list

- listSize: size of the list

```

IDataCursor idc = pipeline.getCursor();
Vector v = (Vector)IDataUtil.get(idc, "vector");
int vSize = 0;
if (v != null){
    vSize = v.size();
    if (vSize > 0){
        IData IDataList[] = new IData[vSize];
        for (int i = 0; i < vSize; i++){
            IDataList[i] = (IData)v.get(i);
        }
        IDataUtil.put(idc, "DocumentList", IDataList);
    }
}
IDataUtil.put(idc, "listSize", String.valueOf(vSize));
idc.destroy();

```

8.17 Tip on Broker Connectivity

Scenario

When the connection to the broker is lost, all guaranteed documents published will get buffered in the Outbound Document Store in the IS and delivered to the Broker when the connection is re-established.

If a document is immediately published after IS-Broker connection is re-established; following message appears in server log:

[Timestamp] Publishing delayed while outbound store is draining. Service: wm.server.publish:publish

Also, the publish step does not reach the broker but goes into the doc store.

Reason

When connection to the Broker is re-established, Integration Server starts draining out the existing documents from its Outbound Document Store to the Broker. Now, if publish to the Broker is done WHILE this process is happening, then the publish action waits until the draining is completed and the published documents also get saved in the Outbound Store and would eventually require draining out, thus slowing down the process. This is accompanied by the same message in server log for each document subsequently published.

The number of documents currently in Outbound Doc Store can be checked at **Administrator > Resources > Store Settings > Outbound Document Store > Current Documents in Outbound Store**. While the Broker is up, this will normally show zero.

Settings > Resources > Store Settings

Outbound Document Store	
Current Documents in Outbound Store	0
Maximum Documents to Send per Transaction	25

Solution

To assist Integration server in quickly draining the outbound store a batch size can be specified that removes documents from the store faster than publishing services that publish them i.e., by increasing the no of documents sent per transaction to broker by IS.

This can be done **Administrator > Resources > Store Settings > Edit Document Store Setting > Outbound Document Store > Maximum Documents to Send per Transaction**

8.18 Tip on Schedulers

If a scheduler is being used, and it is required to ensure that 2 scheduler services do not run in parallel (this can occur when the time to run the scheduled service is more than the scheduled time interval of the scheduler) then tick the check box **"Repeat from end of invocation"**.

This ensures that there is no overlap in execution of the scheduled service.

8.19 Tip on MyWebMethodsServer

If we need to validate Text Fields, (for example: To check if a field needs to be non-empty or if it has to contain only numbers) we can use the Portal built in validator control – **"Exact RegExp"** (available in the validators folder of the Palette)


The steps to use this control for checking if a field should contain 'only numbers' are given below,

1. Drag and Drop the **Exact RegExp** control onto the Text Field for which validation is required
2. In the properties for the control, under the Validators grouping do the following
 - a. Set the 'Regular Expression Pattern' value by
 - i. typing the Reg Exp - **^\d+\$** or
 - ii. open the Reg Exp editor and under the sample Regular Expressions choose 'Number' and then click on **'Use It'** and **'OK'**
 - b. Set the Custom Error Message as "<Field_Name> must be of numeric Type".
 - c. Set the Append CAF Error Message to false
3. Save and Publish

For more information on Regular Expression Syntax please refer to JavaDoc for the **java.util.regex** class.

9 Documentation

<http://documentation.softwareag.com/>



[SEARCH](#)

[SOLUTIONS](#) [PRODUCTS](#) [SERVICES](#) [CUSTOMERS](#) [COMMUNITY](#) [RESOURCES & EVENTS](#) [COMPANY](#)

Product Documentation > webMethods

[Adabas](#)
[ARIS](#)
[Natural](#)
[NaturalONE](#)
[webMethods](#)

webMethods Product Suite

Detailed information on trademarks and patents owned by Software AG and/or its subsidiaries is located at <http://documentation.softwareag.co>

- webMethods Product Suite 8.2
- webMethods Product Suite 8.0 and 8.1
- webMethods Product Suite 8.0 (Japanese)
- webMethods Product Suite 7.1.2 and 7.1.3 (includes Designer 7.1.2 and 7.2)
- webMethods Product Suite 7.1.2 and 7.1.3 (Japanese)
- webMethods Product Suite 7.1 and 7.1.1
- webMethods Product Suite 7.0
- webMethods Product Suite 7.0 (Japanese)

- webMethods Product Suite 6.5
- webMethods Product Suite 6.5 (Japanese)
- webMethods Product Suite 6.5 (Simplified Chinese)