Key Concepts

Following are definitions of some of the concepts and terminology associated with each of the profiles of WSO2 EI 6.6.0.

- Enterprise Service Bus (ESB) concepts
- Business process concepts
- Data services concepts

Enterprise Service Bus (ESB) concepts

Message routing

When there is an incoming message into WSO2 Enterprise Integrator, it is able to determine and route the message to the recipient. Routing can also be done based on some component of the message. This is known as content-based routing and is done using the Switch mediator.

Message filtering

The WSO2 Enterprise Integrator is able to filter out messages based on the message content using the Filter mediator. This feature allows you to perform complex logic, where you are able to filter out messages and send them in different mediation flows.
Message transformation

When sender and receiver messages do not have the same data format, the WSO2 Enterprise Integrator can be used to translate the messages between the sender and recipient. For information on how the Enterprise Integrator can be used for this, see the Message Translator pattern in the EIP Guide. The PayloadFactory mediator and Data Mapper mediator can be used to implement this. You can manipulate messages by adding and removing content from them, converting them to a completely different message format and even validating messages based on the available validation mechanisms of the message format.

For an example of how you can implement message transformation using the Data Mapper mediator, see Transforming Message Content.

Content enriching

You can use the Enrich Mediator to process a message based on a given source configuration and then perform a specified action on the message by using the target configuration. It gets an OMElement using the configuration specified in the source and then modifies the message by putting it on the current message using the configuration in the target.
Protocol switching

The WSO2 Enterprise Integrator has the capability to take messages that come in one protocol and then send the message out in a completely different protocol (e.g. HTTP to JMS). The protocol bridging technology in Enterprise Integrator takes the business content of a message that comes in from one protocol and sends this content out in a completely different format and protocol.

Service chaining

Service chaining (orchestration) is a popular use case in the Enterprise Integrator, where several services are exposed as a single service, aggregated service. Enterprise Integrator is used for the integration and sequential calling of these services so that the expected response can be provided to the client.

For information on implementing a simple service chaining scenario, see Exposing Several Services as a Single Service.
**Message storing and forwarding**

Store and forward messaging pattern is used in asynchronous messaging. This can be used when integrating with systems that accept message traffic at a given rate and handling failover scenarios. In this pattern, messages are sent to a Message Store where they are temporarily stored before they are delivered to their destination by a Message Processor. The Enterprise Integrator is shipped with a few message store implementations and also allows you to implement a custom message store implementation.

For information on implementing a store and forward pattern using the in-memory store of Enterprise Integrator, see the [Storing and Forwarding Messages](#).

For information on an example on implementing guaranteed delivery in Enterprise Integrator, see [Guaranteed Delivery with Failover Message Store and Scheduled Failover Message Forwarding Processor](#).

**Load balancing**

The load balancer automatically distributes incoming traffic across multiple WSO2 product instances. It enables you to achieve greater levels of fault tolerance in your cluster and provides the required balancing of load needed to distribute traffic. For more information, see [Clustering the ESB Profile](#).

Following are the key concepts with respect to the key constructs/artifacts of WSO2 EI.
Message entry points

Proxy services

Proxy services are virtual services that receive messages and optionally process them before forwarding them to a service at a given endpoint. This approach allows you to perform necessary transformations and introduce additional functionality without changing your existing service. Any available transport can be used to receive and send messages from the proxy services. A proxy service is externally visible and can be accessed using a URL similar to a normal web service address.

For more information, see Working with Proxy Services.

REST APIs

A REST API in Enterprise Integrator is analogous to a web application deployed in the Enterprise Integrator runtime. Each API is anchored at a user-defined URL context, much like how a web application deployed in a servlet container is anchored at a fixed URL context. An API will only process requests that fall under its URL context. A REST API defines one or more resources, which is a logical component of an API that can be accessed by making a particular type of HTTP call.

A REST API resource is used by the WSO2 Enterprise Integrator mediation engine to mediate incoming requests, forward them to a specified endpoint, mediate the responses from the endpoint, and send the responses back to the client that originally requested them. We can create an API resource to process defined HTTP request method/s that are sent to the service. The In sequence handles incoming requests and sends them to the back-end service, and the Out sequence handles the responses from the back-end service and sends them back to the requesting client.

REST APIs allow you to send messages directly into the Enterprise Integrator using REST.

For more information, see Working with APIs.

Inbound endpoints

An inbound endpoint is a message source that can be configured dynamically. In the Enterprise Integrator, when it comes to the existing Axis2 based transports, only the HTTP transport works in a multi-tenant mode. Inbound endpoints support all transports to work in a multi-tenant mode.

For more information, see Working with Inbound Endpoints.

Tasks

A task allows you to run a piece of code triggered by a timer. WSO2 Enterprise Integrator provides a default task implementation, which you can use to inject a message to the Enterprise Integrator at a scheduled interval. You can also write your own custom tasks by implementing a Java interface.

For more information, see Scheduling ESB Tasks.

Message processing units

Mediators
Mediators are individual processing units that perform a specific function, such as sending, transforming, or filtering messages. WSO2 Enterprise Integrator includes a comprehensive mediator library that provides functionality for implementing widely used Enterprise Integration Patterns (EIPs). You can also easily write a custom mediator to provide additional functionality using various technologies such as Java, scripting, and Spring.

For more information, see ESB Mediators.

**Sequences**

A sequence is a set of mediators organized into a logical flow, allowing you to implement pipes and filter patterns. You can add sequences to proxy services and REST APIs.

For more information, see Mediation Sequences.

**Message exit points**

A message exit point or an endpoint defines an external destination for a message. An endpoint can connect to any external service after configuring it with any attributes or semantics needed for communicating with that service. For example, an endpoint could represent a URL, a mailbox, a JMS queue, or a TCP socket, along with the settings needed to connect to it.

You can specify an endpoint as an address endpoint, WSDL endpoint, a load balancing endpoint and more. An endpoint is defined independently of transports, allowing you to use the same endpoint with multiple transports. When you configure a message mediation sequence or a proxy service to handle the incoming message, you specify which transport to use and the endpoint where the message will be sent.

For more information, see Working with Endpoints.

**Message stores and processors**

Message stores and message processors are used to store and forward messages while guaranteeing reliable message delivery. For more information, see Working with Message Stores and Message Processors.

**Templates**

Templates help you to manage your configurations without scattering or duplicating them by creating prototypes that you can use and reuse when required. Templates improve re-usability and readability of your ESB configurations (XML files). There are two types of templates available in the ESB profile as Sequence templates and Endpoint templates. For more information on templates, see Working with Templates.

**Connectors**

A connector is a collection of templates that define operations that can be called from the Enterprise Integrator and is used when connecting the Enterprise Integrator to external third party APIs. WSO2 Enterprise Integrator provides a variety of connectors via the WSO2 Connector Store.

For information on using a connector in your EI configuration, see Using the Gmail Connector.

**Transports**

A transport is responsible for carrying messages that are in a specific format. The Enterprise Integrator supports all the widely used transports including HTTP/s, JMS, VFS and domain-specific transports like FIX. You can easily add a new transport using the Axis2 transport framework and plug it into the Enterprise Integrator. Each transport provides a receiver, which the Enterprise Integrator uses to receive messages, and a sender, which it uses to send messages. The transport receivers and senders are independent of the Enterprise Integrator core.

For more information, see Carrying Messages.

**Message builders and formatters**

When a message comes into the Enterprise Integrator, the receiving transport selects a message builder based on the message's content type. It uses that builder to process the message's raw payload data and convert it into common XML, which the Enterprise Integrator mediation engine can then read and understand. WSO2 Enterprise Integrator includes message builders for text-based and binary content.

Conversely, before a transport sends a message out from the Enterprise Integrator, a message formatter is used to build the outgoing stream from the message back into its original format. As with message builders, the message formatter is selected based on the message's content type.

You can implement new message builders and formatters using the Axis2 framework. For more information, see Working with Message Builders and Formatters.

**Applying security to artifacts**

You can apply security to artifacts of the ESB profile using WSO2 Integration Studio. The Quality of Service (QoS) component implements security. For more information, see Applying Security to a Proxy Service.

**Logging messages**

You can use the Log mediator to log mediated messages. For more information on the usage of the log mediator, see Log Mediator.
Message tracing

Message tracing helps you to track issues after an integration process finishes and thereby, allows you to identify and fix issues by identifying the root cause. It is used to trace, track and visualize a body of a message in each intermediate stage of its transmission. This is useful for a number of reasons, including auditing and debugging.

Debugging mediation

Message mediation mode is one of the operational modes of WSO2 EI where EI functions as an intermediate message router. A unit of the mediation flow is a mediator. A sequence is a series of mediators, where each mediator is a unit entity that can input a message, carry out a predefined processing task on the message, and output the message for further processing. Debugging is where you want to know if these units, which function as separate entities are operating as intended, or if a combination of these units are operating as a whole as intended. For more information, see Debugging Mediation.

Enterprise Integration Patterns

Enterprise Application Integration (EAI) enables you to connect business applications with heterogeneous systems. The EIP patterns guide demonstrates how the patterns that are invented integration solution architects over the years can be simulated using various constructs in the ESB profile.

ESB tooling

You can use WSO2 Integration Studio to create various integration artifacts that you can build and deploy to the ESB profile of WSO2 EI in order to process requests.

Business process concepts

[ Business process ] [ Abstract and executable processes ] [ Orchestration vs. choreography ] [ Asynchronous and synchronous communication ] [ Business process modeling ] [ Process execution ] [ Business Process Modelling Notation (BPMN) ] [ BPMN Explorer ] [ Business Process Execution Language (BPEL) ] [ Human tasks ] [ Business process tooling ]

Business process

A business process is typically a collection of related and structured activities or tasks, that depicts a business use case and produces a specific service or output. A process may have zero or more well-defined inputs and an output. During the execution of the business process, it executes its sub-processes synchronously or asynchronously for producing the final output. During the execution, it may interact with both humans or applications.

For example, a banking customer requesting a bank loan is a simple process. The following diagram depicts this process.

Taking the above process as an example, following are the key workflow components of a typical business process.

Process Initiator: In the 'Bank Loan Request' process, a banking customer is the client who initiates a loan request.

Well-Defined Input: Banking customer provides the inputs required for the initialization of the process. It may contain the personal details of the customer, his financial information, account details, etc.

Request Processing: This is typically a sub-process that produces an output internally during the execution of the business process. It analyses the input data, verifies loan eligibility of the client through the execution of several logical expressions etc.

Human Task: This is where a human interaction is involved in the business process. In this particular example, a bank employee sends an acknowledgement to the bank customer regarding his loan request approval.
Final Output: Sends acknowledgement. This is the final output which is sent back to the client who initiated the business process.

Process Instance

An instance of a process is a specific example of a process workflow. For example, if a particular process defines a banking customer requesting a bank loan, then an example instance of this process is Chris requesting for a loan of USD 50,000 and getting approval for it. Every time a banking customer makes a request for a loan, that request triggers a new process instance in the EI-Business-Process runtime, which flows through the elements of the process workflow according to its design.

Abstract and executable processes

Based on the definition of the actual behavior required by a business process, it can design in two ways using WS-BPEL: abstract and executable. Abstract processes are intended to hide some operational details of the process. As a result, they do not include executable details like process flows. Executable business processes are used to model the actual implementation of the business process.

An abstract process is denoted under the http://docs.oasis-open.org/wsbpel/2.0/process/abstract namespace and an executable process is denoted under http://docs.oasis-open.org/wsbpel/2.0/process/executable. Additionally, there are syntactical differences between an abstract and an executable BPEL process.

Orchestration vs. choreography

Web services can be composed using two approaches: orchestration and choreography. In orchestration, there is a central director to coordinate the services. In contrast, choreography contains no central director and each contributing service should have an understanding of participant services.

For composing Web services for a business process, orchestration is a better option for reasons such as simpler process management, loose-coupling between web services, ease in error handling, standardization, etc.

Asynchronous and synchronous communication

BPEL processes can also be categorized based on how it invokes an operation of a partner service: synchronous and asynchronous. It is not possible to use both methods when invoking a partner service's operation, as it is dependent on the type of the partner service operation as well.

Asynchronous transmission - Assume a BPEL process invokes a partner service. After the invocation of the partner process, the BPEL process continues to carry on with its execution process while the partner service completes performing its operation. The BPEL process then receives a response from the partner service, when the partner service is completed.

Synchronous transmission - Assume a BPEL process invokes a partner service. The BPEL process then waits for the partner service's operation to be completed, and responded. After receiving this completion response from the partner service, the BPEL process continues to carry on its execution flow.

This transmission is not applicable to the In-Only operations defined in the WSDL of the partner service.

Usually asynchronous services are used for long-lasting operations and synchronous services for operations that return a result in a relatively short time. Typically, when asynchronous Web services are used, the BPEL process is asynchronous.

Business process modeling

Business process modeling includes identifying four key aspects: process boundaries, activities and events, the resources and their handover and the control flow. Identifying resources for business process modeling leads you to seek information about the systems involved, the people and systems who perform the tasks, the information required and the control flow of the tasks.

Process execution

The diagram below depicts the flow of running a business process in the Business Process profile of WSO2 EI.
Business Process Modelling Notation (BPMN)

The Business Process Management Initiative (BPMI) developed the standard Business Process Modeling Notation (BPMN), which is an executable graphical notation for business processes. The BPMN 2.0 specification was released to the public in January, 2011. A business process in BPMN is a collection of business activities that is focused on a particular business goal or a use case. A process deployment can have one or more such processes.

Activiti runs in any Java application, on a server, on a cluster or in the cloud. BPMN capabilities are integrated to WSO2 EI using the Activiti engine. The Activiti engine runs in any Java application, on a server, on a cluster or in the cloud. It is extremely lightweight and is based on simple concepts. The Business Process profile, which is the Business Process Management (BPM) platform of WSO2 EI is targeted at business owners, developers and system administrators. It is open-source and distributed under the Apache license.

With the graphical notation capability, business analysts can develop the processes and then technical personnel can build the executable process that can be executed and followed by the management. Also, you can visualize business processes graphically using the Activiti Designer Eclipse plugin, which is embedded in WSO2 Integration Studio.

For example, the below is a simple user approval process.

The process starts with a none start event followed by two user tasks, i.e., the registration form getting filled by a front officer followed by the approval from a user in a managerial position. The process ends in a none end event by approving or rejecting the user. Each of these steps are discussed below.
**Start event**

In order to start a process, a process definition must be deployed in the BPMN engine. Deployed processes can then be started as required. In this scenario, when there is a new arrival, a front officer can start a new process. So the none start event will create a new process instance and the engine will execute the process until it reaches a wait state. Now, a new task has been persisted in the system.

A none start event technically indicates that the start time of the task is not defined. There are other start events such as Timer Start Event, Message Start Event, Signal Start Event, etc.

**User task**

Once the event is started, process is at the ‘Fill registration Form’ step. This is referred to as a user task in BPMN. User tasks are tasks that should be completed by human users or an external party to the activity engine. A user task should have authorized users or user groups. So a candidate user can claim and complete the task when a task is created. In this scenario, the front officer can be assigned for the first user task to fill the registration form.

After he completes the task, the engine will continue and halt on the second user task. This task can be assigned to the manager user group so that any of the managers can claim the task and complete it. Once a task is claimed, it will disappear from the task lists of other users.

**End event**

An end event marks the end of a process instance. In this scenario, the process will end after the approval in a none end event. Since this is a none end event, the engine will not do anything other than finishing the process.

There are other types of end events such as Error End Event, that will throw an error, and Cancel End Event, that would cancel the BPMN transaction, etc.

**More constructs**

The scenario explained above is a very simple scenario to introduce the basics. There are a number of constructs available in BPMN 2.0, that could address complex business scenarios. The following are few more useful constructs.

- **Gateways**: act as decisions. This way, a manager could reject the new user registration, which could return to the Fill Registration Form user task.
- **Variables**: are used to store or refer user information so that they can be visualized in the form provided to managers for approval.
- **Service task**: sends the report to every shareholder at the end of the process.
- **Call activity**: calls a sub process when process execution arrives at the activity. Call activity refers to an independent process that is external to the process definition, whereas the sub process will be embedded in the original process definition. The independent process in the call activity is a reusable process that can be called from multiple other process definitions.
- **Manual Task**: is performed without the help of any business process execution engine or application. It models work that is done by an external person, which the engine doesn't need to know. The engine handles the manual task as a pass-through activity where the process is continued automatically when the process execution arrives into it.

E.g. Following is a manual task depicting a delivery boy delivering pizza.

| Place Order | Order Confirmed | Deliver Pizza | Status of Order Delivery |

- The user enters the details of the pizza he/she wants to order which includes the topping and the size.
- Next, the user has to confirm the order by entering the amount.
- Then, the pizza delivery boy will deliver the pizza to the user: Manual Task.
- Finally, the user then confirms the status of the delivery.

**BPMN Explorer**

BPMN Explorer enables you to interact with deployed BPMN applications. It's a Jaggery-based, lightweight web application that you can customize and deploy in a web server.

**Business Process Execution Language (BPEL)**

BPEL is the industry standard for business process orchestration. It is an XML-based language used for the definition and execution of business, as well as scientific workflows using Web services. In other words, BPEL is used to write business processes by composing Web services together with orchestration. The outcome is a composite Web service.

Although these business processes may interact with humans, the WS-BPEL standard does not specify human interactions. As a result, a business process defined by WS-BPEL alone can not have human interactions but only with Web services. The WSO2 BPS facilitates defining and using human tasks in business processes.

**Human tasks**
Business processes cannot always proceed in a fully automated manner. They need human interaction as a means of decision making, error handling and exception cases. For example, cancelling a flight due to a strike or bankruptcy, deciding whether to accept the claim based on the requested amount etc. Human tasks provide the specification to define tasks performed by human beings.

Within a BPEL processes, such tasks are modeled as outgoing service calls, but those service calls are intended for and processed by a human. For example, a loan-issuing business process may include a Human Task step in its workflow to let a Bank Manager/Executive review and approve a loan. Such a task can typically trigger an email/alert to the manager, allowing him to click on a link, review the loan, and approve it. While the approval is pending, the calling processes wait, and the approval or rejection triggers a message in the process, which takes the process to the next step of execution in its workflow.

In SOA, Human-Tasks management is generally facilitated by a Web service. For example, the human tasks server, which manages all human task-related operations, is defined as a web service by the Human Tasks specification. Human tasks are realized using two technical specifications: Web Services for Human Task (WS-HumanTask) specification and WS-BPEL Extension for People (BPEL4People) specification.

WS-HumanTask specification defines interfaces for a task server that enable the workflow engine to create tasks, enabling the organizations to map the tasks to humans and manage them. The BPEL4people specification extends workflow process definitions to include Human Tasks definitions.

The Business Process Profile implements the WS-Human-Task specification. It has the WS-Human Task API and Tooling UI, which expose the functionality of the Task Management API. It enables users to bundle a Human Tasks definition as a ZIP file and upload it, where the task's definition includes input and output message formats for the Human Task.

For more information, see Working with BPEL Processes and Human Tasks.

---

**Business process tooling**

You can use the business process tooling shipped with WSO2 EI (in WSO2 Integration Studio) to create and manage various artifacts with respect to business processes.

**Data services concepts**

- Data service
- Datasources
- RESTful data services
- OData services
- Data Federation
- Distributed transactions
- Batch processing
- Data transformation
- Managed data access
- Streaming
- Namespaces

**Data service**

The data in your organization can be a complex pool of information that is stored in heterogeneous systems, ranging from RDBMSs to Excel files, and Google spreadsheets, etc. Data services are created for the purpose of decoupling the data from its infrastructure. In other words, when you create a data service in WSO2 EI, the data that is stored in a storage system (such as an RDBMS) can be exposed in the form of a service. This allows users (that may be any application or system) to access the data without interacting with the original source of the data. Data services are, thereby, a convenient interface for interacting with the database layer in your organization.

A data service in WSO2 EI is a SOAP-based web service, by default. However, you also have the option of creating REST resources. Therefore, the applications and systems consuming the data service can have both SOAP-based, and RESTful access to your data.

**Datasources**

Your organization's data can be stored in various data storage systems, which are the datasources. Data services in WSO2 EI support the following datasources: Relational databases, CSV files, Microsoft Excel Sheets, Google Spreadsheets, RDF, MongoDB, Cassandra, and Web Resources. Additionally, you can also use JNDI datasources, and create custom datasources. Read about using various datasources with data services defined in WSO2 EI.

**RESTful data services**

A data service exposes your data (stored in various data stores) as a service. You can enable RESTful access to your data, by defining RESTful resources, for the relevant data, in your data service. REST resources in WSO2 EI support both JSON and XML media types out of the box. Therefore, a resource can receive requests, and send responses in either medium. Secure resources with HTTP(S) Basic Auth integrated to enterprise identity systems (via WSO2 Identity Server).

**OData services**

RESTful data services in WSO2 EI supports OData (OData protocol version 4 - OASIS standards), which makes RESTful data access easier. In a normal data service, you will write SQL queries for CRUD operations that will be performed on the data. In other words, to be able to GET, UPDATE, POST, or DELETE data in a database, the data service should have separate SQL queries written for each purpose. However, when you enable OData for your RESTful data service, these CRUD operations will be enabled automatically, which allows RESTful data access using CRUD operations out of the box.

Currently, OData support is now accessible from the following endpoints:

- For super tenant: http://localhost:9763/odata/(dataserviceName)/(datasourceId)/
- For normal tenants: http://localhost:9763/odata/t/{tenantId}/(dataserviceName)/(datasourceId)/

**Data Federation**
A data service defined in WSO2 EI has the ability to aggregate the data that is stored in various, disparate datasources, and present the aggregated data as a single output. For example, the data of employees in a company may be stored in various data stores (details of employment history, details of the physical office, contact information, etc.). Data federation allows users to consume all this data through a single request to the data service. The data service will aggregate the relevant data from each of the disparate datasources and present it as one response to the request. Data federation can be achieved in two ways:

- Expose multiple datasources using a single data service.
- Use Nested Queries in your data service. This will allow you to feed the result you get from one query as input to another query. That is, data can be combined into a single response or resource.

### Distributed transactions

A distributed transaction is a set of operations that should be performed on two or more, distributed RDBMS data stores. If the operation on one data store (node) fails, the entire set of operations will fail in all the data stores. In other words, a distributed transaction is an example of a batch process, where multiple requests are grouped into one server call and processed as one unit by the data service.

Data services in WSO2 EI support distributed transactions, which allows data consumers to perform such transactions easily by using one data service as the interface. Note that distributed transactions can only be performed for IN-ONLY operations that will insert, update, or delete data in the data stores. These are not applicable to operations for retrieving data.

A transaction manager is set up in the middle of these transactions for effective coordination and management. This feature uses Java Transaction API (JTA), which allows distributed transactions to be carried out across multiple XA resources in a Java environment. You can also override this transaction manager.

### Batch processing

A data service is an interface that receives requests from data consumers and performs the requested tasks in the relevant data stores. Batch processing allows a data service to group multiple requests into a batch and process it as a single request. Batch processing can only be used for IN-ONLY operations that will insert, update, or delete data in the data stores, and not for operations that retrieve data.

Data services in WSO2 EI support two scenarios of batch requesting: Client-side batch requests, and server-side batch requests.

For example, consider the task of entering details of new employees into a database table. Typically, the client consuming the data can do this by sending separate requests with each employee record. Alternatively, the client can group the individual requests into a single batch, and send one batch request to the data service. In this example, the data service will have one operation defined for inserting data into the database. However, when batch processing is enabled, it is possible to insert multiple records into that database, using this operation. Therefore, the client can invoke this operation using a single request, to insert multiple records. This is client-side batch requesting.

Consider another example, where the client needs to enter the employee's bank details along with the personal details, but the bank details should be inserted to a different data store. In this example, the data service will have two separate operations for inserting data into two separate data stores, and the client is invoking both operations, using a single request (also called a request box). This is server-side batch requesting.

Note that batch requests are transactional if the data store is an RDBMS, or another system that supports transactions. Transactional requests succeed or fail as a batch. That is, if one individual request fails, all the requests in the batch will fail to make sure that the data is synchronized. Server-side batch requests work for local transactions (performed on one node of the data store), as well as distributed transactions (performed on multiple nodes of the data store).

### Data transformation

XSLT transformation is used in data services to transform the result of an already defined operation into a different result. The user can define the transformation xslt and provide the url of the transformation file in the result element.

### Managed data access

Most businesses require secure and managed data access across these federated data stores.

### Streaming

Data service streaming helps manage large data chunks sent back to the client by the data service as the response to a request. When streaming is enabled, the data is sent to the client as it is generated, without memory building up in the server. By default, streaming is enabled in data services.

### Namespaces

The service namespace uniquely identifies a Web service and is specified by the `<targetNamespace>` element in the WSDL that represents the service. A data service is simply a Web service with specialized functionality. When developing a data service, you get to apply namespaces at various levels. As a data service implementation is based on XML, namespace handling is useful for making sure that there are no conflicting element names in the XML. Although namespaces are optional for data services, in some scenarios they are necessary.