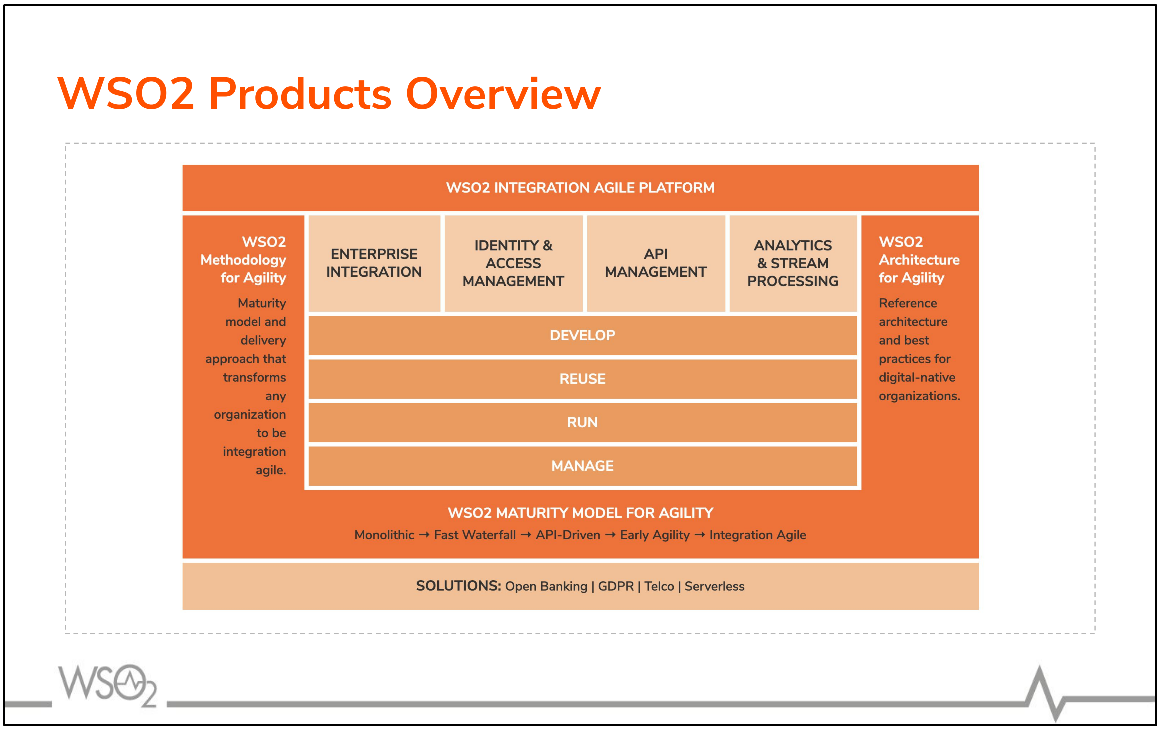
WSO2 Product Administration

WSO2 Training

Module 01 - Introduction



WSO2 Products Overview

WSO2 INTEGRATION AGILE PLATFORM
WSO2 Methodology for Agility
Maturity model and delivery approach that transforms any organization to be integration agile.
ENTERPRISE INTEGRATION
IDENTITY & ACCESS MANAGEMENT
API MANAGEMENT
ANALYTICS & STREAM PROCESSING
WSO2 Architecture for Agility
Reference architecture and best practices for digital-native organizations.
DEVELOP
REUSE
RUN
MANAGE
WSO2 MATURITY MODEL FOR AGILITY
Monolithic → Fast Waterfall → API-Driven → Early Agility → Integration Agile
SOLUTIONS: Open Banking | GDPR | Telco | Serverless
WSO2

Spans the entire breadth of Service Oriented Architecture (SOA), yet remain lean and easy to use. We have solutions for Security, Portals and Stores Device Management, Analytics, App Development and Management, API Management and Integration.

Let's look at the composition of the WSO2 middleware platform

The lowest level in the WSO2 Middleware platform , contains components that enable you to connect to legacy systems, SaaS applications and databases to access data and perform service orchestrations.

The integration components residing in the next level, access the services exposed by the bottom level components to integrate and facilitate communication among heterogeneous systems.

The API Management component is built on top of the integration component to exploit integration capabilities of the integration layer and exposes backend services in a secure and controlled manner.

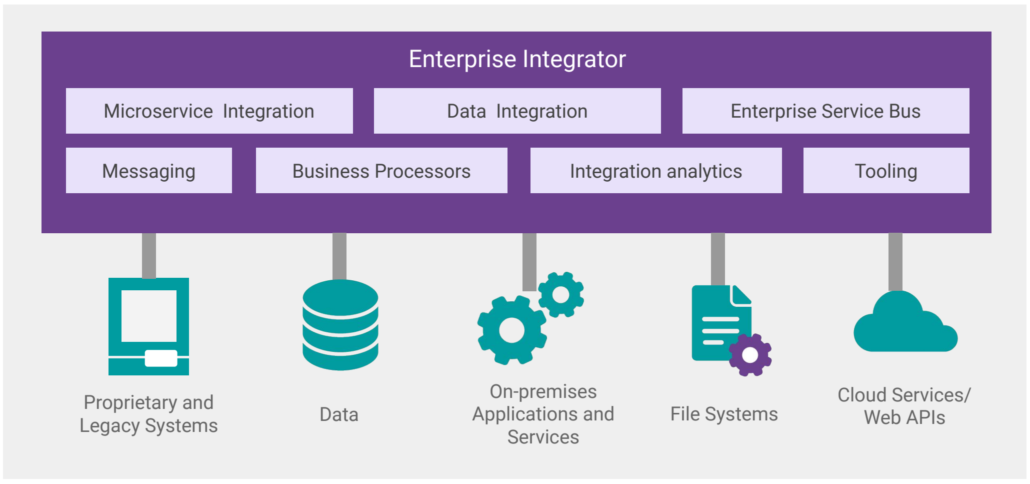
The middleware platform also consists of components featuring Application development and management.

API and Application Management components can be fronted with Portals and stores consisting of asset stores to support user friendly access.

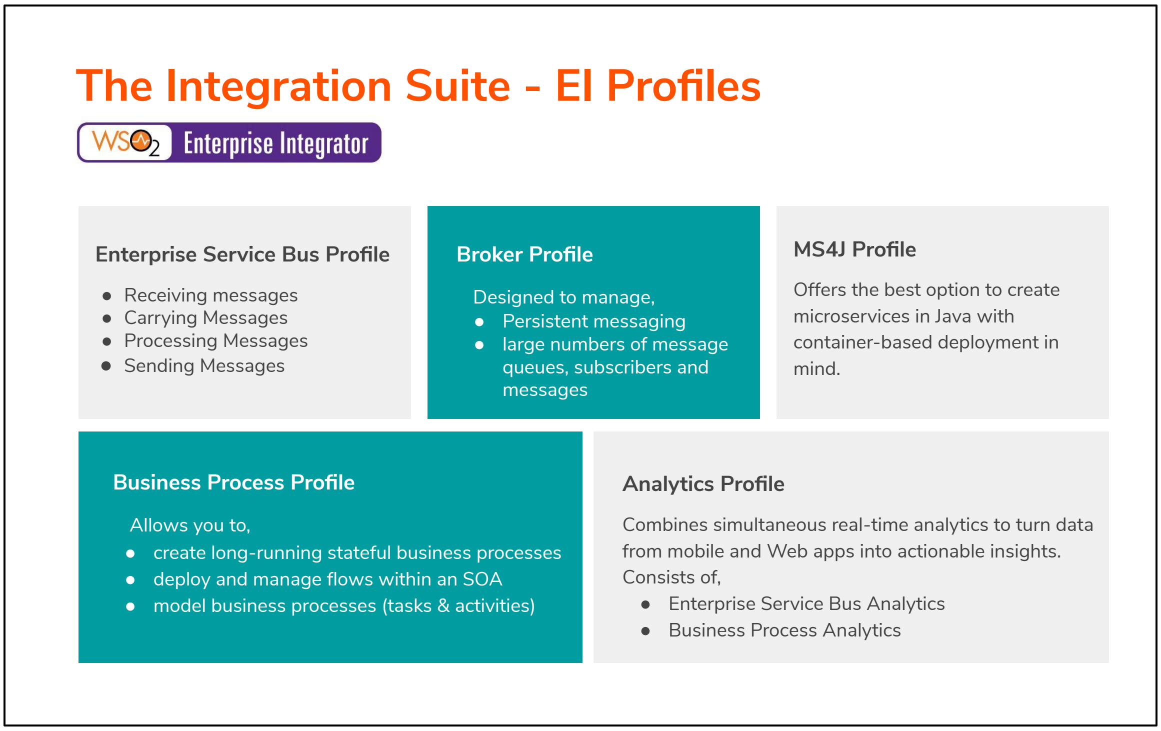
The platform also facilitates Mobile and IoT device management.

Finally as you can see on the diagram, there are Security and Analytics components that are common ingredients in all the products belonging to the platform.
Each product has built in security features to authenticate users, manage user stores and secure services.
The same way all products have built in features to enable analytics using our analytical platform.

The Integration Suite



WSO2 Enterprise Integrator is built to address the integration challenges faced in modern day enterprises. It provides a centralized integration middleware/ESB with data integration, process integration and B2B integration capabilities. In addition to short-running, stateless integration flows, it can be used to manage long-running, stateful business processes. It also comes with analytics for comprehensive monitoring, and message brokering capabilities that can be used for reliable messaging, and capabilities to run microservices for your integration flows.


The Integration Suite - EI Profiles

WSO2 Enterprise Integrator

Enterprise Service Bus Profile

- Receiving messages
- Carrying Messages
- Processing Messages
- Sending Messages

Broker Profile

Designed to manage,
- Persistent messaging
- large numbers of message queues, subscribers and messages

MS4J Profile

Offers the best option to create microservices in Java with container-based deployment in mind.

Business Process Profile

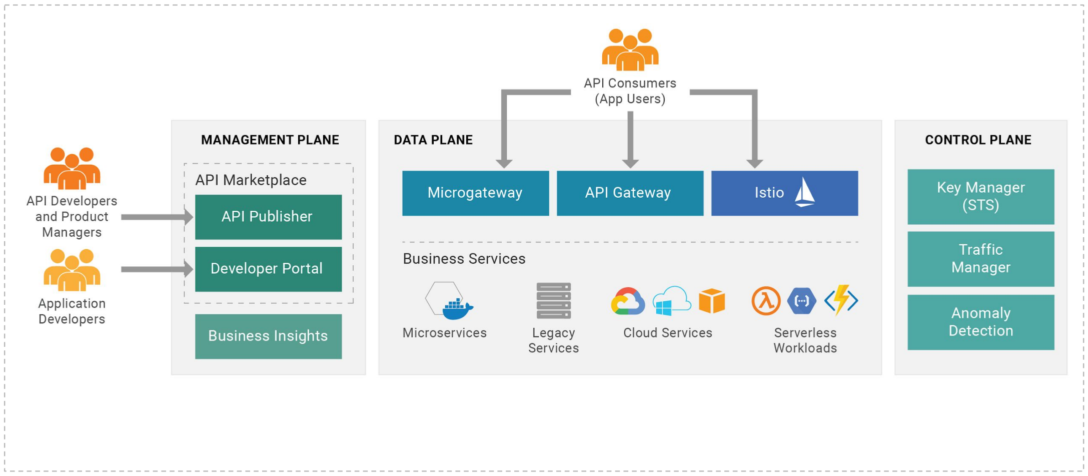
Allows you to,
- create long-running stateful business processes
- deploy and manage flows within an SOA
- model business processes (tasks & activities)

Analytics Profile

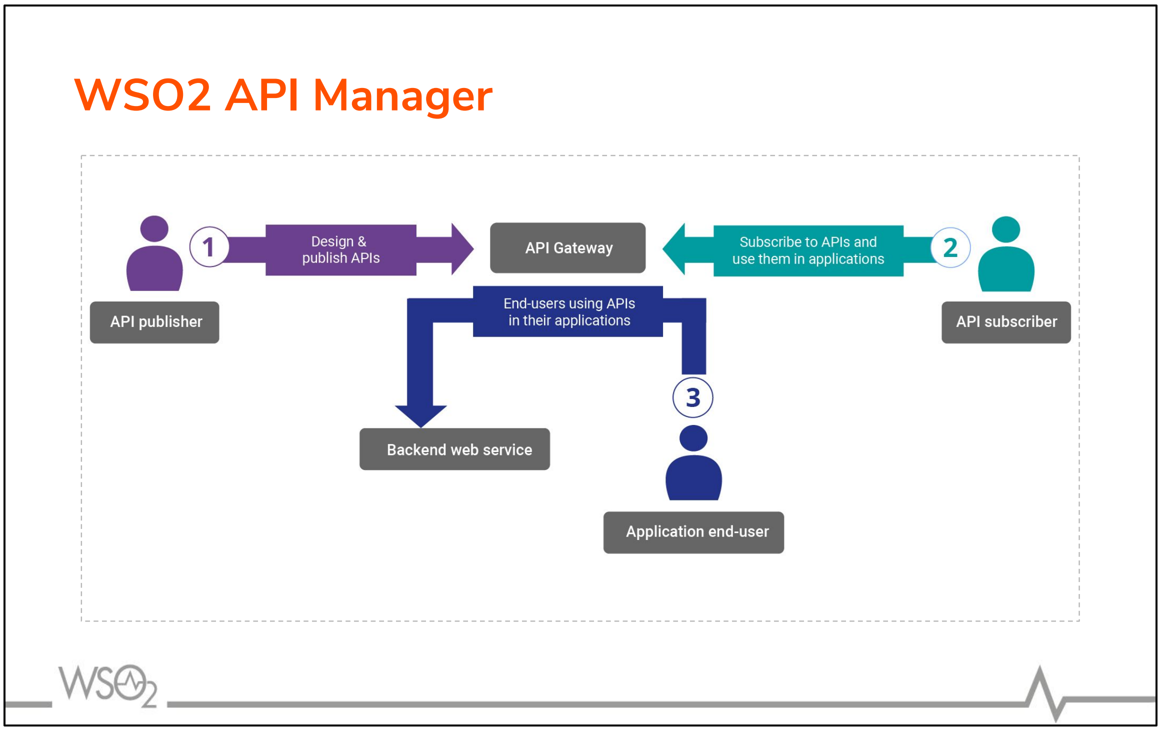
Combines simultaneous real-time analytics to turn data from mobile and Web apps into actionable insights. Consists of,
- Enterprise Service Bus Analytics
- Business Process Analytics

Here you can see a summary of features supported by products forming WSO2's integration suit discussed in the previous slide.

WSO2 API Manager



WSO2 API Manager supports API publishing, lifecycle management, application development, access control, rate limiting and analytics in one cleanly integrated system.



WSO2 API Manager

The WSO2 API Management Platform is a collaborative and secure platform with many capabilities including enterprise governance, integration, analytics and identity management as illustrated in this diagram.

This platforms provides portals for API publishers and consumers.
It allows API publishers to provision their APIs, publish API documentation, manage keys, and gather statistics on API usage.
Consumers can discover and subscribe to APIs, test them online, generate keys and consume.

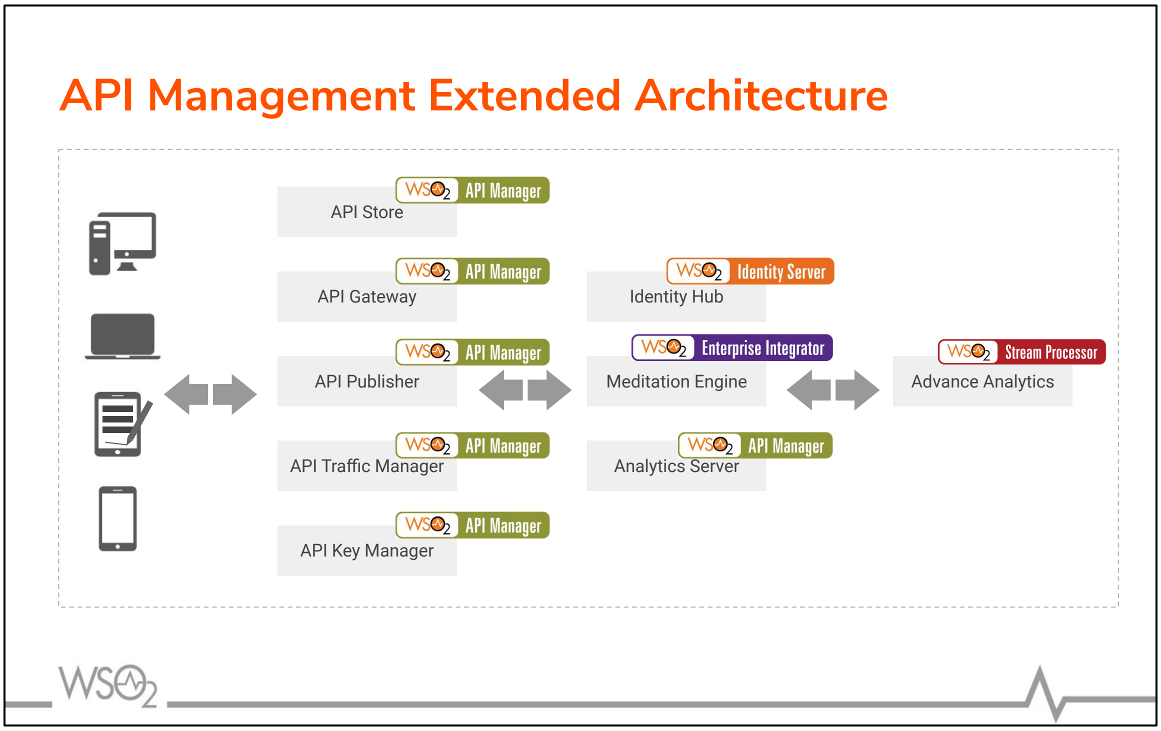
API Management platform also consists of a key server or the 'OAuth Server to enable its security capabilities for API authentications

The API Gateway enforces security, throttling and other Quality-of-service features.
It also handles basic mediation of API requests.

API analytics support helps the platform to illustrate and exploit the business value of APIs and monetization.

Extending the API Management Platform using the WSO2 Enterprise Service Bus , gives the platform the advantage of moving the integration logic on gateway to ESB.
This will offload the integration processing from gateway in situations where high traffic needs to be handled or when the integration flow has a complex logic.

API Management platform supports to build, publish and consume secure APIs, handle API lifecycles, enforce monetization strategies, throttling limitations Quality-of-service features establishing its enterprise governance capabilities.



API Management Extended Architecture

In a typical scenario involving an API call, the API Gateway forwards the request to the backend.

But if you need additional functionality such mediations, data manipulations, security checks and statistics, it may not be practical to handle within the Gateway, as it could have adverse effect on the Gateway's performance.

This is where you need an extended API Management Architecture.
Let's see in detail how WSO2 API Management platform enables this.

WSO2 API Manager provides a Storefront and a Gateway.
The store can be accessed by Application Developers and the Gateway initially deals with incoming API requests.

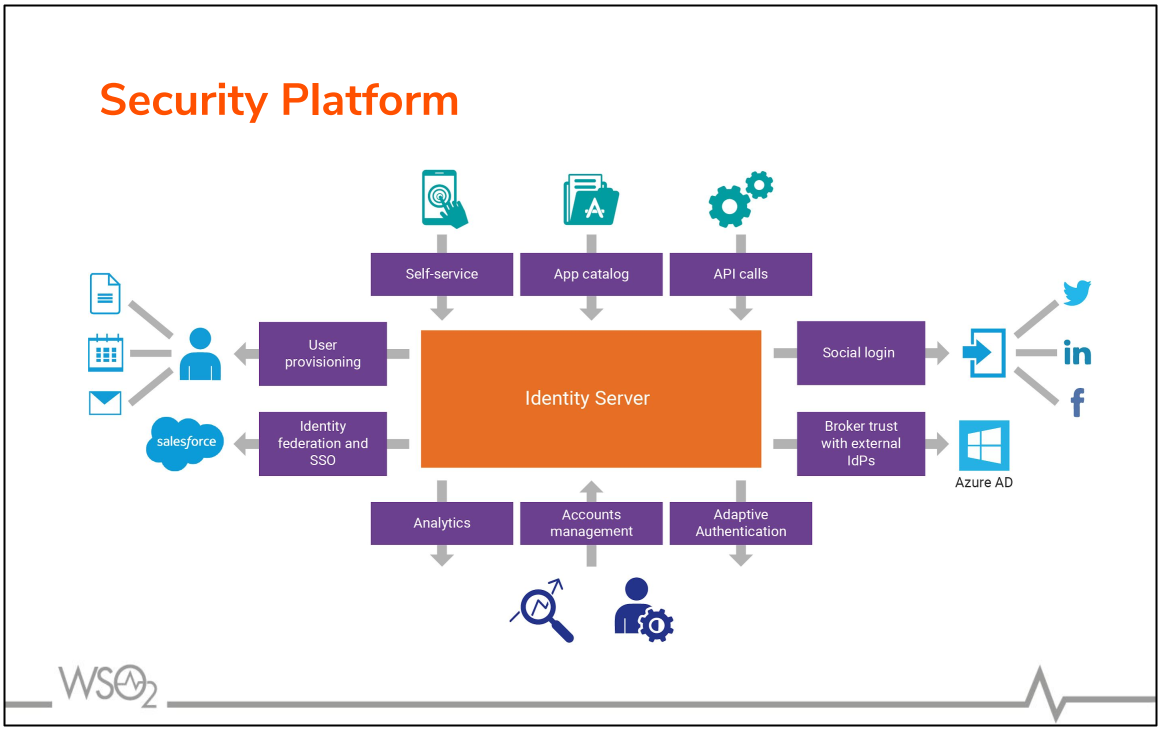
API creation, publishing and governance are managed separately using the API publisher component of WSO2 API Manager.

Security validations and authentication are handled by the WSO2 Identity Server.

WSO2 Enterprise Integrator is capable of handling any mediation required for incoming and outgoing messages.

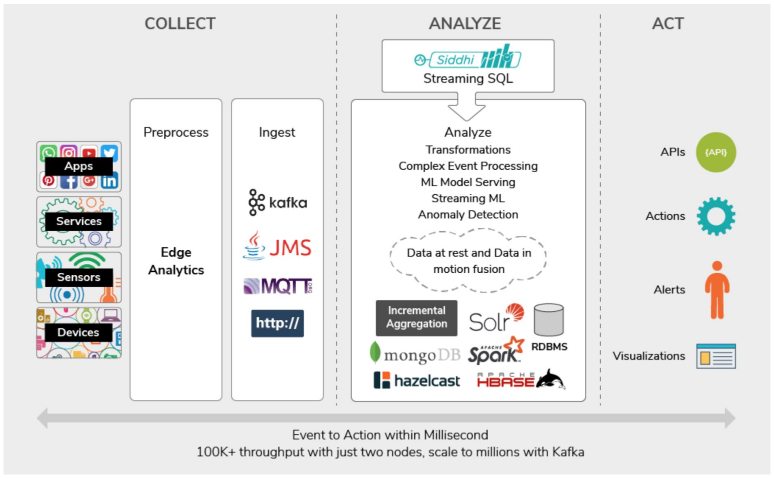
WSO2 API Manager has an inbuilt Analytics component using WSO2 Analytics but Advanced Analytics can be performed using the WSO2 Stream Processor.

These products together help build an extended, secure and a highly configurable platform for API Management.



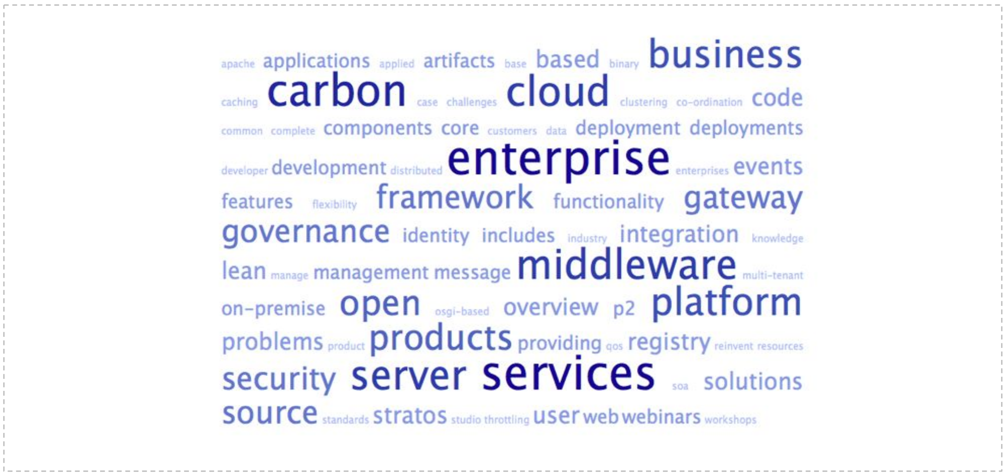
WSO2 Identity Server is an open source IAM product that specializes in access management, comprising of identity federation, SSO, strong & adaptive authentication, access control, account management & identity provisioning, API & microservices security, and privacy regulation. The product is highly extensible and offers the flexibility to write extensions with its rich set of connectors, which gives the ability to connect to other third-party and cloud systems and to support complex IAM use cases that fit any identity requirement.

Analytics Platform



Collect events through multiple transports and messaging formats. Use Streaming SQL to process streams, detect complex events and do prediction using machine learning models. Generate and notify alerts in real-time and visualize them with real time dashboards.

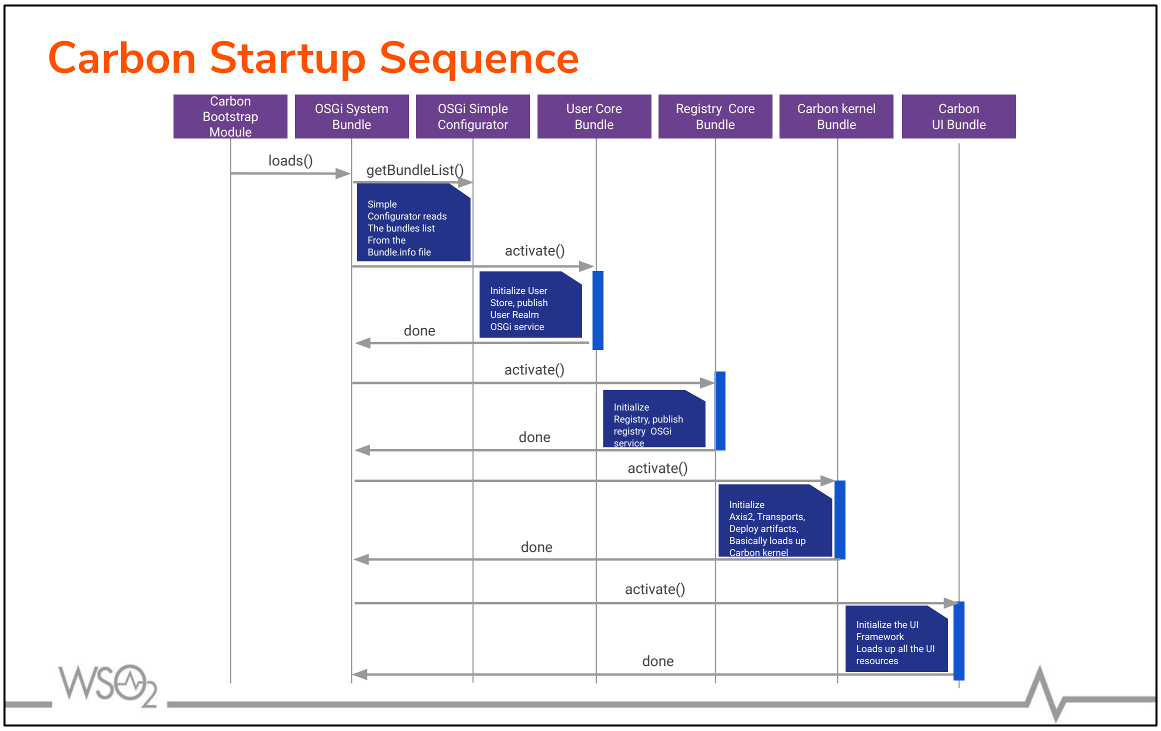
What is WSO2 Carbon?


apache applications applied artifacts base based binary business
caching carbon case challenges cloud clustering co-ordination code
common complete components core customers data deployment deployments
developer development distributed enterprise enterprises events
features flexibility framework functionality gateway
governance identity includes industry integration knowledge
lean manage management message middleware multi-tenant
on-premise open osgi-based overview p2 platform
problems product products providing qos registry reinvent resources
security server services soa solutions
source standards stratos studio throttling user web webinars workshops

WSO2 Carbon is the award-winning, component-based, service oriented platform for the enterprise-grade WSO2 middleware products stack. It is 100% open source and delivered under Apache License 2.0. The WSO2 Carbon platform is lean, high-performant and consists of a collection of OSGi bundles.

The WSO2 Carbon core platform hosts a rich set of middleware components encompassing capabilities such as security, clustering, logging, statistics, management and more. These are basic features required by all WSO2 products that are developed on top of the base platform.
All WSO2 products are a collection of Carbon components. They have been developed simply by plugging various Carbon components that provide different features. The WSO2 Carbon component manager provides the ability to extend the Carbon base platform, by selecting the components that address your unique requirements and installing them with point-and-click simplicity. As a result, by provisioning this innovative base platform, you can develop your own, lean middleware product that has remarkable flexibility to change as business requirements change.

Having a thorough knowledge on Carbon ensures that you are thorough with 60% of the relevant product and this will also help you troubleshoot any problems that you encounter in a very effective manner.


Carbon Startup Sequence
Carbon Bootstrap Module
OSGi System Bundle
OSGi Simple Configurator
User Core Bundle
Registry Core Bundle
Carbon kernel Bundle
Carbon UI Bundle
loads()
getBundleList()
Simple Configurator reads The bundles list From the Bundle.info file
activate()
Initialize User Store, publish User Realm OSGi service
done
activate()
Initialize Registry, publish registry OSGi service
done
activate()
Initialize Axis2, Transports, Deploy artifacts, Basically loads up Carbon kernel
done
activate()
Initialize the UI Framework Loads up all the UI resources
done
WSO2

When the startup script of a Carbon product is run, it will call the Carbon launcher or the Carbon bootstrap module which launches the OSGi framework.

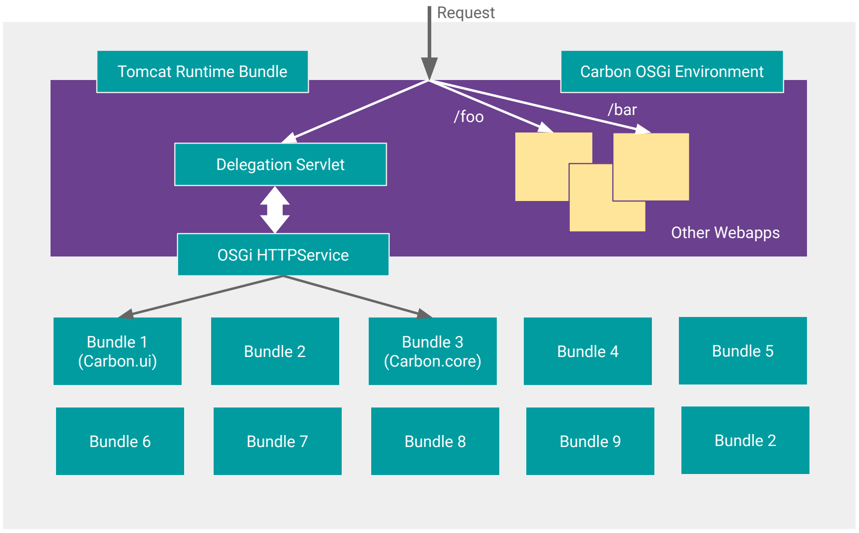
OSGi framework then loads system plugins and bundles and starts them.

Certain components require a start up order.

For example,

- User core component should be initialized before the registry core component.
- Registry core component should be initialized before the Carbon core.
- Likewise Carbon UI framework requires the initialization of Axis2 runtime and transports.

Carbon - Inner Architecture

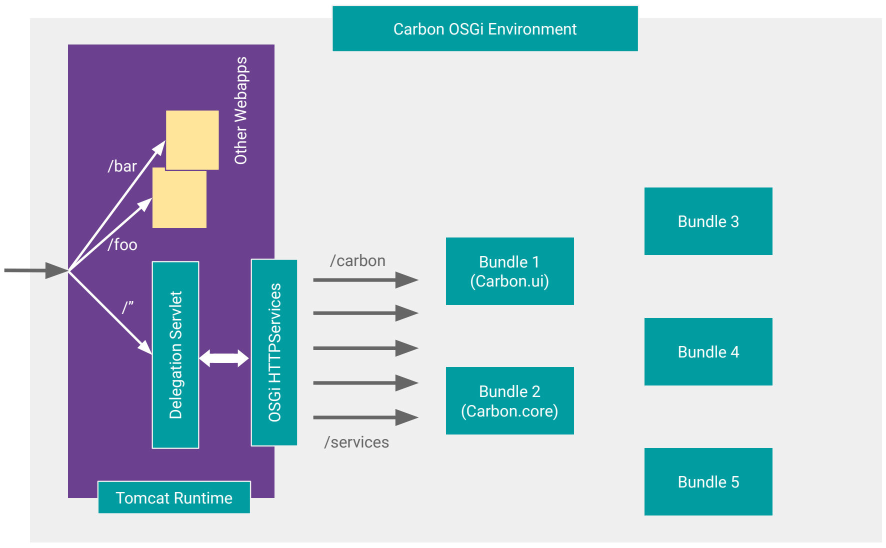


The diagram shows how service requests or Carbon UI request or any other requests get dispatched to the correct bundle by the Carbon servlet transport.

When the Carbon servlet transport receives a request, if the request is for the root context or the carbon context, the request is dispatched to the Delegation Servlet.
The Delegation Servlet then dispatches the request to the relevant OSGi bundle using the OSGI HTTPService.

If the request matches with the context of a deployed web application, it will be dispatched to the relevant web Application.

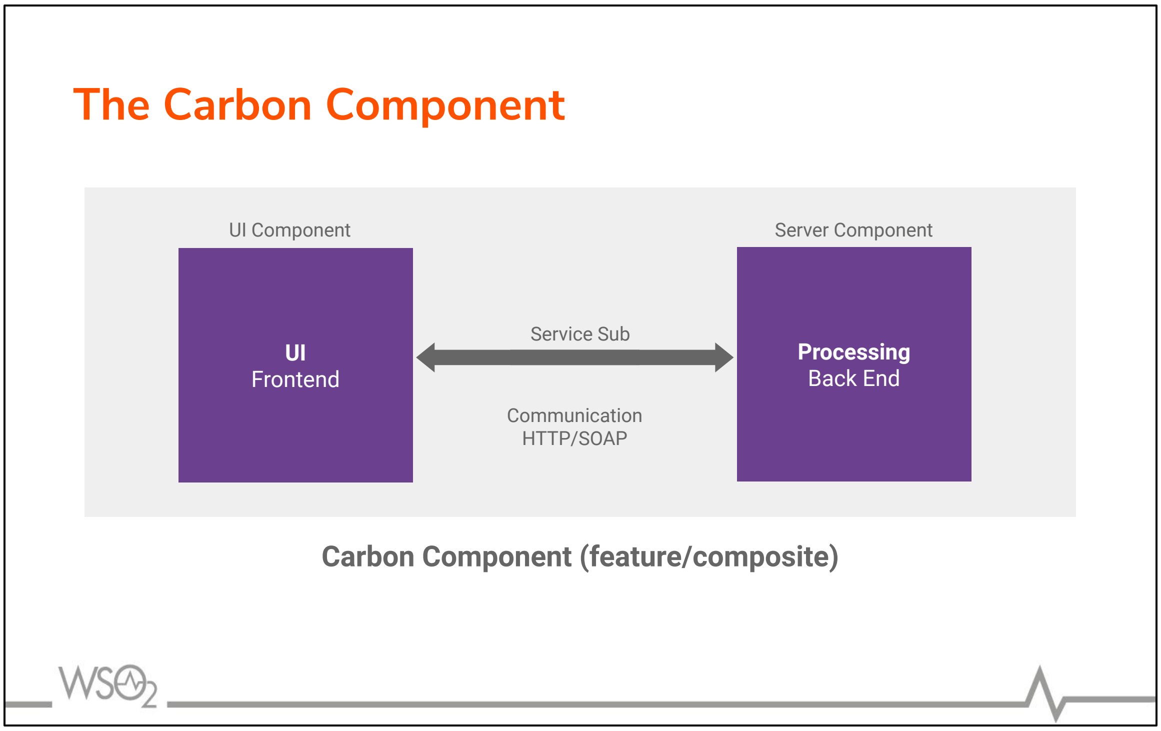
Carbon - Inner Architecture



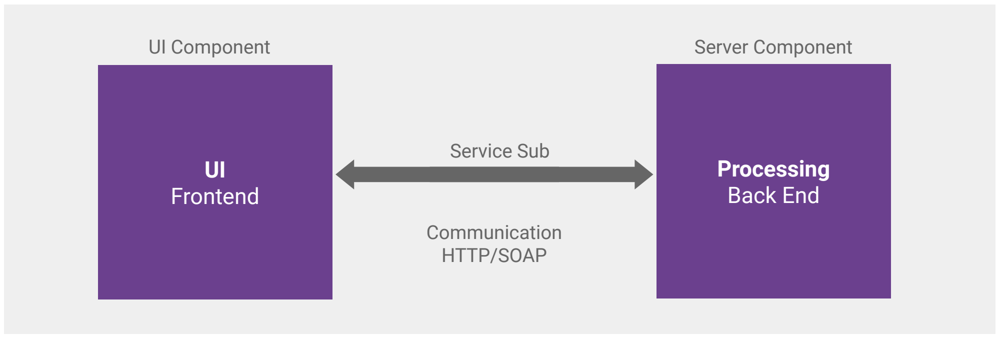
The diagram shows how service requests or Carbon UI request or any other requests get dispatched to the correct bundle by the Carbon servlet transport.

When the Carbon servlet transport receives a request, if the request is for the root context or the carbon context, the request is dispatched to the Delegation Servlet.
The Delegation Servlet then dispatches the request to the relevant OSGi bundle using the OSGI HTTPService.

If the request matches with the context of a deployed web application, it will be dispatched to the relevant web Application.



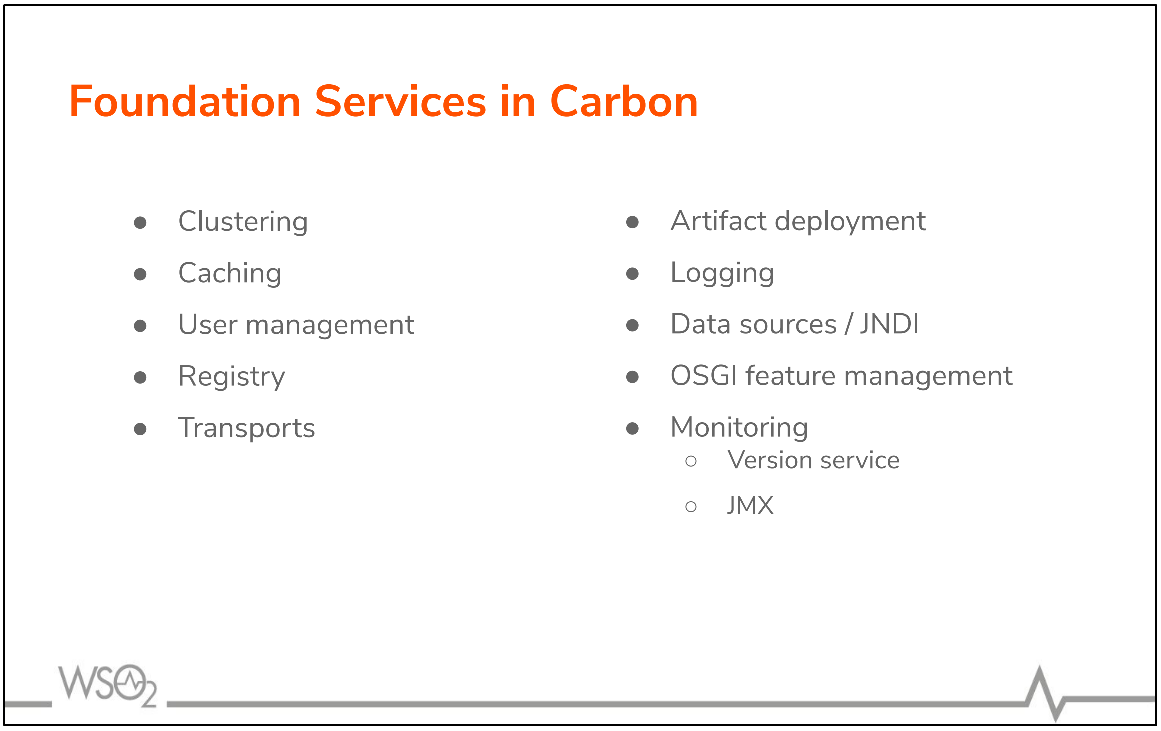
The Carbon Component



Carbon Component (feature/composite)

If we look at the composition of a typical Carbon component, it consists of a backend component and a front end component.

Backend component carries out all heavy duty processing and exposes web services in order for the front end UI component to interact with it.
The UI component communicates with the server component via the Service Stub using HTTP or SOAP.


Foundation Services in Carbon

- Clustering
- Caching
- User management
- Registry
- Transports
- Artifact deployment
- Logging
- Data sources / JNDI
- OSGI feature management
- Monitoring
  - Version service
  - JMX

There is a set of services that make up the foundation of Carbon Platform.

Clustering enables multiple instances of products to divide up the work, yet act as a single instance.
The requests coming to a cluster are distributed among the servers, thereby making another instance seamlessly handle requests in case of a failure of one instance.

This improves the performance of the overall deployment.

The Carbon platform also supports caching at different levels to reduce data access and response times with the purpose of increasing performance.

User management is bundled within the Carbon Platform to facilitate the management and control of users and roles.
Carbon based Products expose a set of user management APIs for component developers and they can also be configured to use different types of internal and external user stores.

Registry allows to govern and monitor the deployment, and the platform exposes core registry APIs.

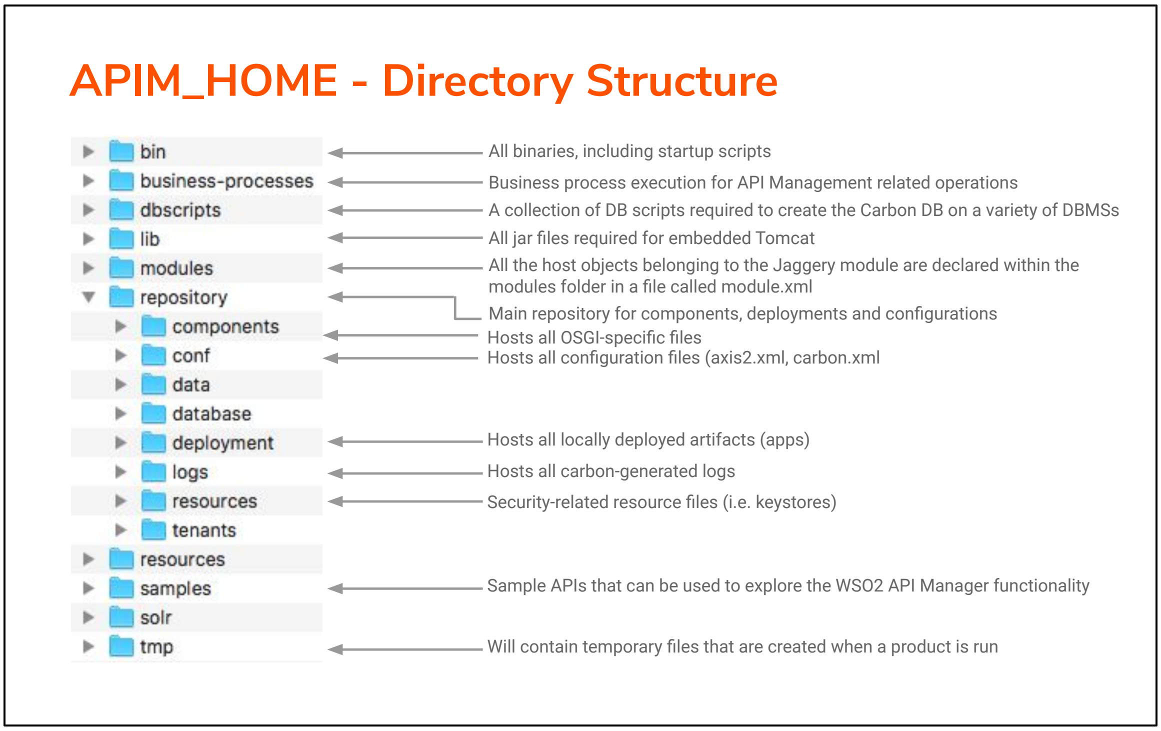
Carbon platform supports a number of transports directly and indirectly based on the Apache Axis2 transport framework.
This framework exposes interfaces to Transport Listener and Sender of each transport, thereby enabling carbon based products to send and receive messages to and from a multitude of transports and application level protocols.

Products based on Carbon platform support deployment of different types of artifacts such as Axis2 services, data services , synapse configurations, proxy services, mediators and registry resources.
These artifacts can be bundled in Composite Applications and deployed on Carbon servers at runtime.

WSO2 products are shipped with log4j logging capabilities to generate server side logs which are crucial for identifying errors, security violations and usage patterns.
In addition to logs that come from libraries using Log4j, logs based on the Java logging framework are also written to the same log files.
These cover logs from tomcat and hazelcast related libraries as well.

Carbon platform contains a datasource management feature that allows configuring data sources and connecting databases or external data stores.
You can also expose data sources as JNDI Data Sources.

As a result of being based on the OSGI framework, the carbon platform contains independent components, known as OSGI bundles that form carbon features.
These bundles can be added or removed from a solution dynamically.

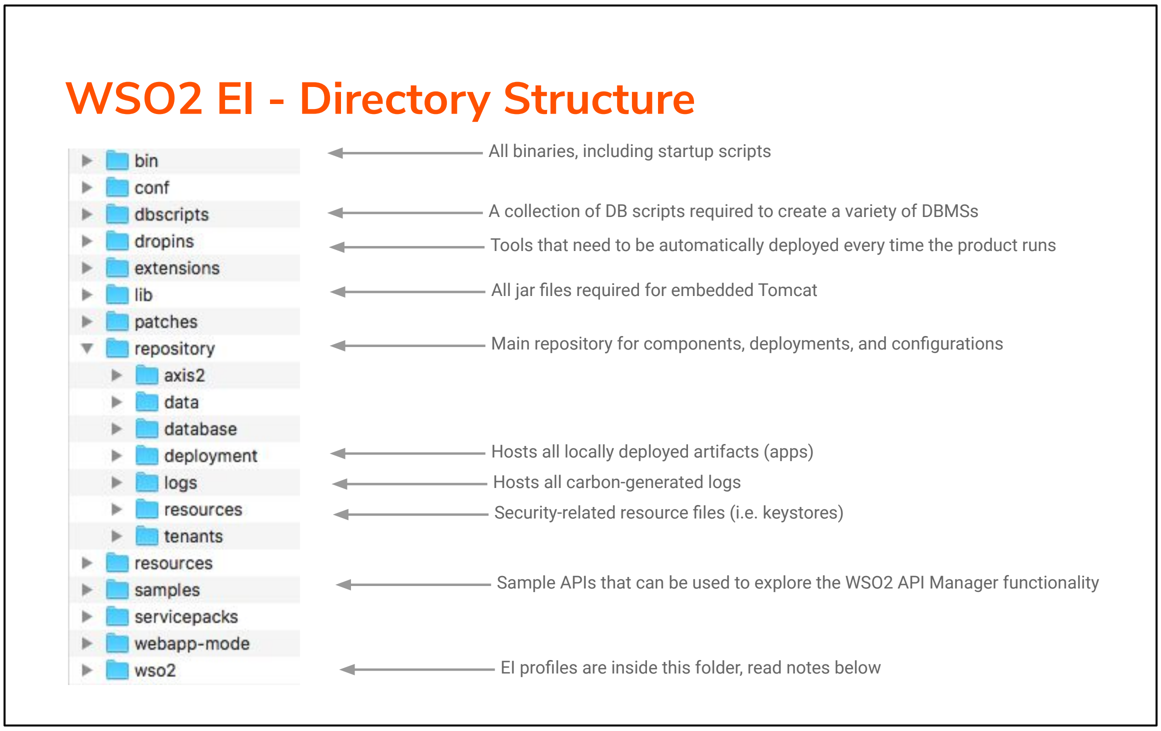


APIM_HOME - Directory Structure

bin — All binaries, including startup scripts
business-processes — Business process execution for API Management related operations
dbscripts — A collection of DB scripts required to create the Carbon DB on a variety of DBMSs
lib — All jar files required for embedded Tomcat
modules — All the host objects belonging to the Jaggery module are declared within the modules folder in a file called module.xml
repository — Main repository for components, deployments and configurations
components — Hosts all OSGI-specific files
conf — Hosts all configuration files (axis2.xml, carbon.xml
data
database
deployment — Hosts all locally deployed artifacts (apps)
logs — Hosts all carbon-generated logs
resources — Security-related resource files (i.e. keystores)
tenants
resources
samples — Sample APIs that can be used to explore the WSO2 API Manager functionality
solr
tmp — Will contain temporary files that are created when a product is run

The directory structure of Carbon is common to all products based on a particular Carbon version except WSO2 Enterprise Integrator.

For example, in product_home/ you can find product specific text files, directories containing samples , database scripts and libraries.
bin/ directory contains the startup script and repository/ contains artifacts that are utilized during server run time.
Within the 'repository/ directory you get OSGI specific files in the components/ directory, deployed artifacts in the deployment/ directory and server log files in the logs/ directory.
resources/ contain security artifacts such as key stores and truststores.
Tenant specific artifacts are store within the tenants' directory.



WSO2 EI - Directory Structure

bin	All binaries, including startup scripts
conf	
dbscripts	A collection of DB scripts required to create a variety of DBMSs
dropins	Tools that need to be automatically deployed every time the product runs
extensions	
lib	All jar files required for embedded Tomcat
patches	
repository	Main repository for components, deployments, and configurations
axis2	
data	
database	
deployment	Hosts all locally deployed artifacts (apps)
logs	Hosts all carbon-generated logs
resources	Security-related resource files (i.e. keystores)
tenants	
resources	
samples	Sample APIs that can be used to explore the WSO2 API Manager functionality
servicepacks	
webapp-mode	
wso2	EI profiles are inside this folder, read notes below

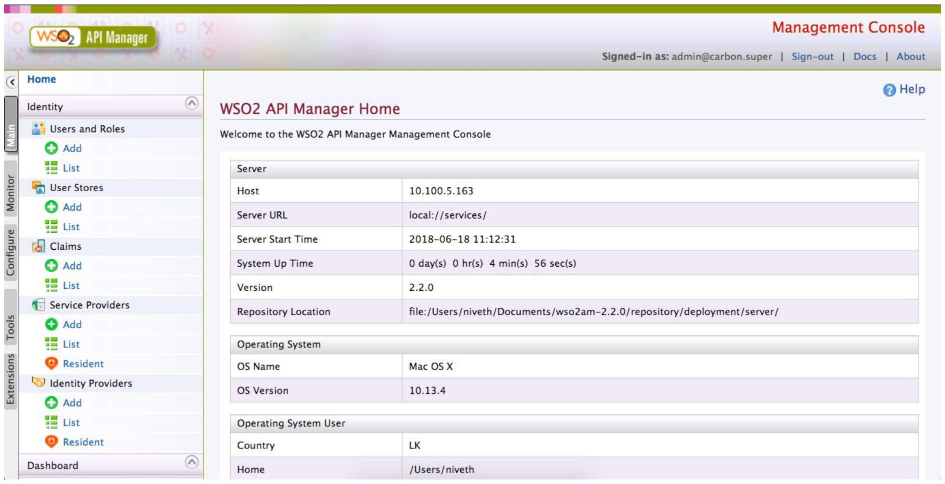
Unlike other products, EI has a folder called wso2 within its HOME Directory. This 'wso2' folder contains all the EI Profiles which were discussed on the 4th slide (Enterprise Integration Suite)

The 'lib' folder in the HOME Directory includes the ESB profile lib files, specific to ESB only. Meanwhile for the other EI profiles, their lib files can be found within the HOME > wso2 > BrokerProfile > lib.

Configuration Files

- **carbon.xml** - The carbon server configuration file
- **user-mgt.xml** - The User Manager configuration file used for configuring user management details.
- **axis2.xml** - Used to do configurations specific to a variety of web services. (eg. clustering, transports)
- **registry.xml** - The carbon registry configuration file. This will be used when the WSO2 Embedded Registry is used.

UI Management Console



WSO2 API Manager Home

Welcome to the WSO2 API Manager Management Console

Server	
Host	10.100.5.163
Server URL	local://services/
Server Start Time	2018-06-18 11:12:31
System Up Time	0 day(s) 0 hr(s) 4 min(s) 56 sec(s)
Version	2.2.0
Repository Location	file:/Users/niveth/Documents/wso2am-2.2.0/repository/deployment/server/

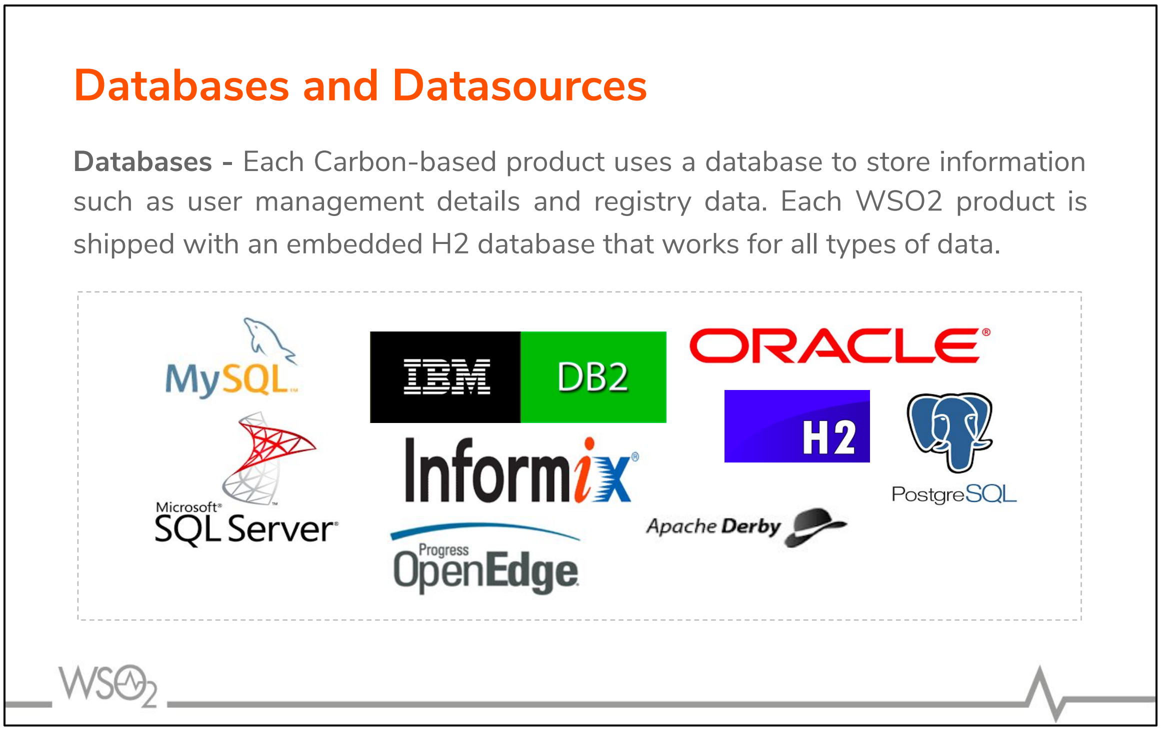
Operating System	
OS Name	Mac OS X
OS Version	10.13.4

Operating System User	
Country	LK
Home	/Users/niveth

https://localhost:9443/carbon (admin:admin)

Let's try it out!

Installing WSO2 API Manager and Management Console Basics



Databases and Datasources

Databases - Each Carbon-based product uses a database to store information such as user management details and registry data. Each WSO2 product is shipped with an embedded H2 database that works for all types of data.

Databases - All nodes in the cluster must use one central database for config and governance registry mounts

Databases and Datasources

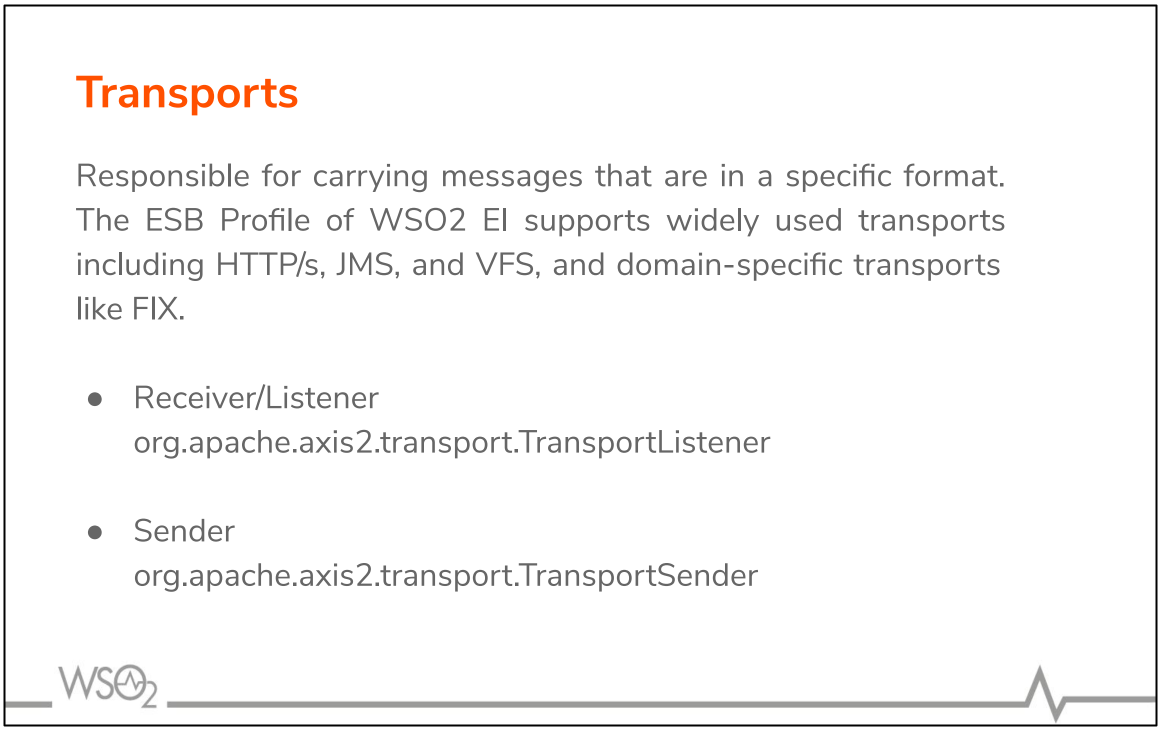
Datasources - A datasource provides information that a server can use to connect to a database. Either a RDBMS or a Custom Datasource can be created through the WSO2 management console.

Datasource management is provided by the following feature in the WSO2 feature repository:

Name : WSO2 Carbon - datasource management feature
Identifier: org.wso2.carbon.datasource.feature.group

Let’s try it out!

Connecting to a Database



Transports

Responsible for carrying messages that are in a specific format. The ESB Profile of WSO2 EI supports widely used transports including HTTP/s, JMS, and VFS, and domain-specific transports like FIX.

- Receiver/Listener
  org.apache.axis2.transport.TransportListener

- Sender
  org.apache.axis2.transport.TransportSender

All EI transports are directly or indirectly based on the Apache Axis2 transports framework.

Because each transport implementation has to implement the above two interfaces, each transport generally contains a transport receiver/listener implementation and a transport sender implementation.

Let’s try it out!

Transport Support using the Enterprise Integrator

THANK YOU

wso2.com