webMethods Adapter for Apache Cassandra Installation and User's Guide

Version 10.2

April 2018

WEBMETHODS

This document applies to webMethods Adapter for Apache Cassandra 10.2 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Document ID: ADAPTER-CAS-IUG-102-20221202

Table of Contents

About this Guide

This guide describes how to install, configure, and use Adapter for Apache Cassandra. It provides an overview of how Apache Cassandra operates and explains common tasks you can perform using Adapter for Apache Cassandra.

To use this guide effectively, you should be familiar with:

- Other software applications and protocols you can use with Apache Cassandra.
- The setup and operation of webMethods Integration Server.
- Have a general idea about how to perform basic tasks with Software AG Designer.

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
Narrowfont	Identifies service names and locations in the format *folder.subfolder.service*, APIs, Java classes, methods, properties.
Italic	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
`Monospace font`	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Product Documentation

You can find the product documentation on our documentation website at https://documentation.softwareag.com.

In addition, you can also access the cloud product documentation via https://www.softwareag.cloud. Navigate to the desired product and then, depending on your solution, go to "Developer Center", "User Center" or "Documentation".

Product Training

You can find helpful product training material on our Learning Portal at https://knowledge.softwareag.com.

Tech Community

You can collaborate with Software AG experts on our Tech Community website at https://techcommunity.softwareag.com. From here you can, for example:

- Browse through our vast knowledge base.
- Ask questions and find answers in our discussion forums.
- Get the latest Software AG news and announcements.
- Explore our communities.
- Go to our public GitHub and Docker repositories at https://github.com/softwareag and https://hub.docker.com/u/softwareag and discover additional Software AG resources.

Product Support

Support for Software AG products is provided to licensed customers via our Empower Portal at https://empower.softwareag.com. Many services on this portal require that you have an account. If you do not yet have one, you can request it at https://empower.softwareag.com/register. Once you have an account, you can, for example:

- Download products, updates and fixes.
- Search the Knowledge Center for technical information and tips.
- Subscribe to early warnings and critical alerts.
- Open and update support incidents.
- Add product feature requests.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

1 Overview of the Adapter

About the Adapter

Adapter for Apache Cassandra is an add-on to webMethods Integration Server that enables you to interact with Apache Cassandra using DataStax java driver. Apache Cassandra is a distributed NoSQL database management system designed to handle large amount of data across many commodity servers, providing high availability with no single point of failure. Cassandra offers robust support for clusters spanning multiple datacenters, with asynchronous masterless replication allowing low latency operations for all clients.

Using Adapter for Apache Cassandra, you can create and run adapter services to perform operations such as insert, update, delete and query data from a Cassandra table or Column Family.

Apache Cassandra Concepts

This section discusses the following concepts:

Keyspace

A keyspace is also called as a schema which is similar to a database or schema in RDBMS. It is the unit for Cassandra's access control mechanism and also contains a set of tables.

Table

A table is known as a column family. It defines the column names and data types. Also, it is a map of rows. The client application provides rows that conform to the schema. Each row has the same fixed set of columns.

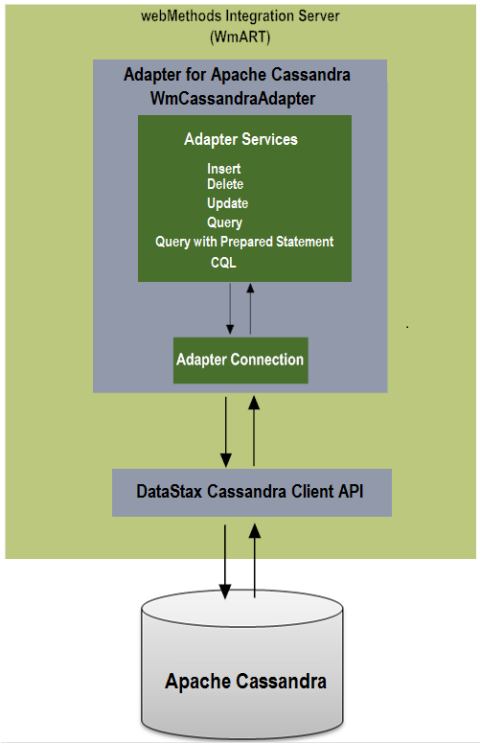
Rows

Cassandra supports tables defined with composite primary keys. The first column in a composite key definition is used as the partition key. Remaining columns are automatically clustered. Rows that share a partition key are sorted by the remaining components of the primary key.

Architecture and Components

Adapter for Apache Cassandra provides a set of user interfaces and services that enable you to create integration with Datastax. The adapter is provided as a single package that must be installed on Integration Server. For detailed installation instructions, see "Overview of installing and uninstalling Adapter for Apache Cassandra" on page 20. For software requirements, see *webMethods Adapters System Requirements* .

The following diagram shows at a high level, how the adapter components connect to Apache Cassandra.



- **webMethods Integration Server.** Adapter for Apache Cassandra is installed and runs on Integration Server.
- **(WmART).** The WmART package provides a common framework for webMethods Adapter for Apache Cassandra version 10.2 and later to use Integration Server's functionality, making Integration Server the run-time environment for Adapter for Apache Cassandra. The WmART package is installed with Integration Server and it provides logging, error handling for the adapter, connections, and services.
- **Adapter for Apache Cassandra.** The Adapter for Apache Cassandra is delivered as a single package called WmCassandraAdapter. The adapter installation includes templates from which all adapter connections, and adapter services can be created. The adapter provides:
  - Integration Server Administrator user interfaces that will enable you to configure and manage adapter connections
  - Software AG Designer user interfaces that will enable you to configure and manage adapter services
- **Adapter Service.** Adapter services enable the Integration Server to initiate and perform database operations on Datastax. You configure adapter services using adapter services templates, which are provided with Adapter for Apache Cassandra. For more information about adapter services, see "Overview of Adapter Services" on page 42.
- **Adapter Connection.** Adapter connections enable the Integration Server to connect to Datastax at runtime. You must configure an adapter connection before you can configure adapter services. For a detailed description of adapter connections, see "Overview of Adapter for Apache Cassandra Connections" on page 30.

- **DataStax Cassandra Client API.** DataStax Cassandra Client API is a client jar provided by DataStax that connects to the Cassandra cluster .
- **Apache Cassandra.** Apache Cassandra is a high-performance, extremely scalable, fault tolerant (i.e., no single point of failure), distributed non-relational database solution. Cassandra combines all the benefits of Google Bigtable and Amazon Dynamo to handle the type of database management requirements that the traditional RDBMS vendors cannot support. DataStax is the leading worldwide commercial provider of Cassandra products, services, support, and training.

Package Management

Adapter for Apache Cassandra is available as a package called WmCassandraAdapter. You can manage the WmCassandraAdapter package like any package on Integration Server.

You can set up and manage the packages on Integration Server using the following considerations:

- Create user-defined packages for your connections and adapter services. For details, see "Adapter for Apache Cassandra Package Management" on page 24.
- Understand how package dependencies work so that you can decide on how to manage your adapter services. For details, see "Package Dependency Requirements and Guidelines" on page 25.
- You must have a control on which development groups have access to which adapter services. For details, see "Group Access Control" on page 27.

Adapter Connections

Adapter for Apache Cassandra connects to DataStax through Client jar API at run time. You create one or more connections at design time to use in integrations.

Note:
Enabling or Disabling the connection will not have any effect as pooling is not supported by the adapter.

For instructions on configuring Adapter for Apache Cassandra connections, see "Overview of Adapter for Apache Cassandra Connections" on page 30. For information about setting user privileges, see the *webMethods Integration Server Administrator's Guide* for your release.

Built-In Services for Connections

Integration Server provides built-in services that enable you to programmatically control connections. You can use them to enable and disable a connection, and to return usage statistics and the current state (Enabled or Disabled) and error status for a connection. These services are located in the WmART package, in the pub.art.connection folder.

The setAdapterServiceNodeConnection built-in service enables you to change the connection associated with an adapter service respectively. For more information, see "Change the Connection Associated with an Adapter Service at Design Time" on page 15.

For details, see the *webMethods Integration Server Built-In Services Reference* for your release.

Adapter Services

To use Adapter for Apache Cassandra, you create adapter services. Adapter services allow you to connect to the adapter's resource and initiate an operation on the resource from Integration Server

You call adapter services from flow or Java services to interact with . The adapter services perform Cassandra table operations by calling Datastax Cassandra Client API. Integration Server then uses adapter connections that you defined earlier to execute the adapter services. For details, see "Adapter Service Processing" on page 15.

Adapter services are based on templates provided with Adapter for Apache Cassandra. Each template represents a specific operation on a resource, such as using the Insert service template to inserts specified information to the table.

An adapter service template contains all the code necessary for interacting with the resource but without the data specifications. You provide these specifications when you create a new adapter service.

Creating a new service from an adapter service template is straightforward. Using Software AG Designer, you assign the service a default adapter connection.

After you select the connection for the adapter service, you select the adapter service template and supply the data specifications using Designer. Some familiarity with using Designer is required. For more information, see the *webMethods Service Development Help* for your release.

Adapter for Apache Cassandra provides the following adapter service templates:

Adapter Service Template	Description
Insert	Inserts the records into the table. For instructions about configuring the service, see "Configuring Insert Service" on page 42 .
Update	Updates the existing records in a table by updating the value of the column and deletes the existing value in the column without removing the columns. For instructions about configuring the service, see "Configuring Update Service" on page 50 .
Delete	Deletes the records from the table. For instructions about configuring the service, see "Configuring Delete Service" on page 45.
Query	Retrieves the records from the table.

Adapter Service Template	Description
	For instructions about configuring the service, see "Configuring Query Service" on page 55.
Query with Prepared Statement	Retrieves records from the table using prepared statement. For instructions about configuring the service, see "Configuring Query with Prepared Statement Service" on page 60.
CQL	Executes the configured CQL statement. For instructions about configuring the service, see "Configuring CQL Service" on page 62.

Using Adapter Services

The following table lists the tasks required to use the adapter services.

For this task...	Use these tools...
1. Create an adapter connection. For details, see "Overview of Adapter for Apache Cassandra Connections" on page 30.	Integration Server Administrator
2. Select the appropriate adapter service template and configure the adapter service. Depending on the type of adapter service, you specify: ■ The adapter connection ■ The input fields and types as needed ■ The output fields and types as needed For more information about configuring adapter services, see "Overview of Adapter Services" on page 42.	Designer
3.If you plan to use an Integration Server flow or Java service to invoke the adapter service, design the flow or Java service to use this adapter service.	Designer
4. Manage the adapter service.	Designer Integration Server

Change the Connection Associated with an Adapter Service at Design Time

Integration Server provides built-in services that you can use at design time to change the connection associated with an adapter service. The built-in services are provided in the WmART package's pub.art.service folder. Using the service setAdapterServiceNodeConnection, you can change the specific connection associated with an adapter service at design time so that you need not create and maintain multiple adapter services.

Note:
The setAdapterServiceNodeConnection can run at design time only. Do not use it within an Integration Server flow or Java service. You must run the services directly from Designer by selecting a service and running it.

For details, see the *webMethods Integration Server Built-In Services Reference* for your release.

Other built-in services enable you to control connections. For more information, see "Built-In Services for Connections" on page 12.

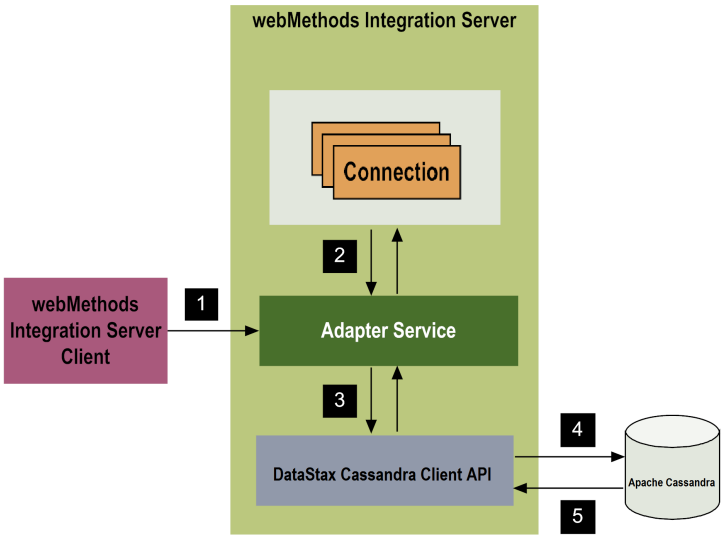
Change the Connection Associated with an Adapter Service at Run Time

Integration Server enables you to dynamically select the connection that a service uses to interact with the adapter's resource. This feature enables one service to interact with multiple, and similar backend resources.

For example, a service can be defined to use a default connection that interacts with your company's production database. However, at run time you can override the default connection and instead use another connection to interact with the company's test database.

Adapter Service Processing

The following diagram illustrates how Adapter for Apache Cassandra processes adapter services at run time.



Step	Description
1	An Integration Server client, typically using a flow or Java service, invokes a Adapter for Apache Cassandra service on Integration Server to perform an operation on a Apache Cassandra. You configured the adapter service earlier using Designer.
2	The adapter service gets a connection from the service's connection pool. Adapter connections contain connection information of Apache Cassandra.
3	The adapter service uses the DataStax Cassandra Client API to connect to Apache Cassandra.
4	All adapter services performs the CRUD operation against a data in Apache Cassandra. ■ For Insert, Delete, Update, and Query services, the adapter service performs the corresponding operation against the data in Apache Cassandra.
5	Depending on the adapter service type, such as a Query service, the adapter service may return data to Integration Server. ■ If the operation is successful, the service returns the output from the Client jar API, if applicable. ■ If the operation is unsuccessful, the service returns an error such as an Adapter Exception. If the Apache Cassandra throws an exception while performing the adapter service's operation, the adapter logs the exception in the Integration Server log.

Viewing the Adapter's Update Level

You can view the list of updates that have been applied to the adapter. The list of updates appears in the **Updates** field on the adapter's About page in Integration Server Administrator.

Controlling Pagination

When using the adapter on Integration Server 10.1 and later, you can control the number of items that are displayed on the adapter Connections screen. By default, 10 items are displayed per page. Click **Next** and **Previous** to move through the pages, or click a page number to go directly to a page.

To change the number of items displayed per page, set the watt.art.page.size property and specify a different number of items.

To set the number of items per page

1. From Integration Server Administrator, click **Settings**>**Extended** .

2. Click **Edit Extended Settings**. In the Extended Settings editor, add or update the watt.art.page.size property to specify the preferred number of items to display per page. For example, to display 50 items per page, specify:

   ```
   watt.art.page.size=50
   ```

3. Click **Save Changes**. The property appears in the Extended Settings list.

   For more information about working with extended configuration settings, see the *webMethods Integration Server Administrator's Guide* for your release.

2 Installing and Uninstalling Adapter for Apache Cassandra

Overview of installing and uninstalling Adapter for Apache Cassandra

This chapter explains how to install, and uninstall webMethods Adapter 10.2 for Apache Cassandra. The instructions use the Software AG Installer and Software AG Uninstaller wizards. For complete information about the wizards or other installation methods, or to install other webMethods products, see the *Installing webMethods Products On Premises* for your release.

Requirements

For a list of operating systems, and webMethods products supported by Adapter for Apache Cassandra, see *webMethods Adapters System Requirements* .

Adapter for Apache Cassandra has no hardware requirements beyond those of its host Integration Server.

The Integration Server Home Directory

Beginning with Integration Server 9.8, you can create and run multiple Integration Server instances under a single installation directory. Each Integration Server instance has a home directory under *Integration Server_directory* \instances*instance_name* that contains the packages, configuration files, log files, and updates for the instance.

For more information about running multiple Integration Server instances, see the *webMethods Integration Server Administrator's Guide* for your release.

This guide uses the packages_directory as the home directory in Integration Server classpaths. For Integration Server 9.8 and above, the packages_directory is *Integration Server_directory* \instances*instance_name*\packages directory.

Installing Adapter for Apache Cassandra

1. Download Installer from the Empower Product Support website.

2. If you are installing the adapter on an existing Integration Server, shut down the Integration Server.

3. Start the Installer wizard.

4. Choose the webMethods release that includes the Integration Server on which you want to install the adapter. For example, if you want to install the adapter on Integration Server 10.2, choose the 10.2 release.

5. Specify the installation directory as follows:

   - If you are installing on an existing Integration Server, specify the webMethods installation directory that contains the host Integration Server.

- If you are installing on an existing Integration Server and the adapter, specify the installation directory to use.

6. In the product selection list, select **Adapters** > **webMethods Adapter 10.2 for Apache Cassandra**.

   If you are using Integration Server 10.1 and above, you can choose to install the package in the default instance. In this case, Software AG Installer installs the adapter in both locations, *Integration Server_directory* \packages and the default instance packages directory located in *Integration Server_directory* \instances\default\packages.

7. To download the documentation for the adapter, go to Software AG Documentation website.

8. After the installation completes, close the Installer and start the host Integration Server.

Note:
Be sure that Apache Cassandra driver jars are installed while you are installing webMethods Adapter for Apache Cassandra.

Uninstalling Adapter for Apache Cassandra

To uninstall Adapter for Apache Cassandra

1. Shut down the host Integration Server. You do not need to shut down any other webMethods products or applications that are running on your machine.

2. Start Software AG Uninstaller, selecting the webMethods installation directory that contains the host Integration Server.

3. In the product selection list, select **Adapters** > **webMethods Adapter 10.2 for Apache Cassandra**. You can also choose to uninstall documentation.

4. After Uninstaller completes, restart the host Integration Server.

   Note:
   Uninstaller removes all Adapter for Apache Cassandra related files that were installed. However, Uninstaller does not delete files created after you installed the adapter (for example, user-created or configuration files), nor does it delete the adapter directory structure. You can go to the *Integration Server_directory* \packages directory and *Integration Server_directory* \instances\default\packages directory. Delete the WmCassandraAdapter directory.

3 Package Management

Overview of Package Management

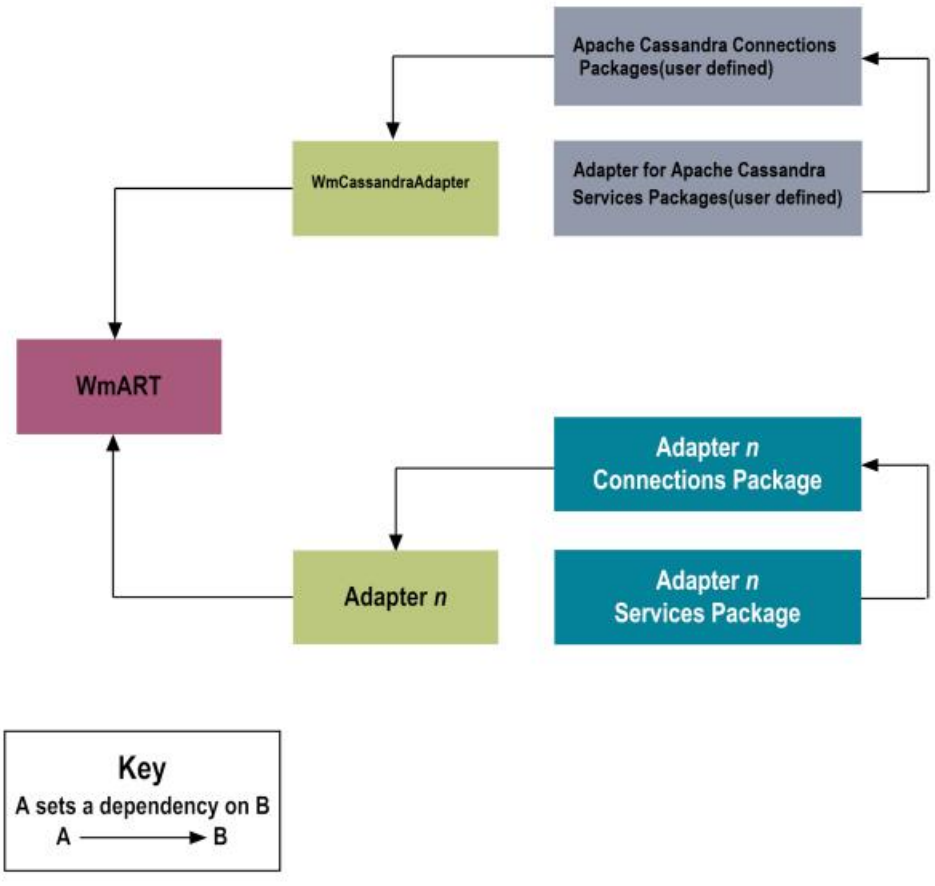
The following sections describe how to set up and manage your webMethods Adapter for Apache Cassandra packages and to set up Access Control Lists (ACLs).

Adapter for Apache Cassandra Package Management

Adapter for Apache Cassandra is provided as a package called WmCassandraAdapter. You can manage the WmCassandraAdapter package as you would manage any package on webMethods Integration Server.

When you create connections, and adapter services, define them in user-defined packages rather than in the WmCassandraAdapter package. Doing so will allow you to manage the package more easily.

As you create user-defined packages in which to store connections, and adapter services, use the package management functionality provided in Software AG Designer and set the user-defined packages to have a dependency on the WmCassandraAdapter package. That way, when the WmCassandraAdapter package loads or reloads, the user-defined packages load automatically. See the following diagram:



Package management tasks include:

- Setting package dependencies (see)

Package Dependency Requirements and Guidelines

This section contains a list of dependency requirements and guidelines for user-defined packages. For instructions for setting package dependencies, see the *webMethods Service Development Help* for your release.

- A user-defined package must have a dependency on its associated adapter package, WmCassandraAdapter. (The WmCassandraAdapter package has a dependency on the WmART package.)
- Package dependencies ensure that at start-up the Integration Server automatically loads or reloads all packages respectively: the WmART package, the adapter package, and the user-defined packages. The WmART package is automatically installed when you install Integration Server. You should manually reload the WmART package.
- If the connections and adapter services of an adapter are defined in different packages, then:
  - A package that contains the connections must have a dependency on the adapter package.
  - Packages that contain adapter services must have a dependency on their associated connection package.
- Keep connections for different adapters in separate packages so that you do not create interdependencies between adapters. If a package contains connections for two different adapters, and you reload one of the adapter packages, the connections for both adapters will reload automatically.
- Integration Server will not allow you to enable a package if it has a dependency on another package which is disabled. That is, before you can enable your package, you must enable all packages on which your package depends. For information about enabling packages, see .
- Integration Server will allow you to disable a package even if another package that is enabled has a dependency on it. Therefore, you must manually disable any user-defined packages that have a dependency on the adapter package before you disable the adapter package. For information about disabling packages, see .
- You can provide same name for the connections, and adapter services, if they are in different folders and packages.

Enabling Packages

All packages are automatically enabled by default. Use the following procedure when you want to enable a package that was previously disabled.

To enable a package

1. Open Integration Server Administrator if it is not already open.

2. In the **Packages** menu of the navigation area, click **Management.**

3. Click **No** in the **Enabled** column. The server displays **Yes** in the **Enabled** column.

Note:
Enabling an adapter package will not cause its associated user-defined packages to be reloaded.

Important:
Before you manually enable a user-defined package, you must first enable its associated adapter package (WmCassandraAdapter).

Disabling Packages

When you want to temporarily prohibit access to the elements in a package, disable the package. When you disable a package, the server unloads all of its elements from memory. Disabling a package prevents Integration Server from loading that package at startup.

Important:
If your adapter has multiple user-defined packages, and you want to disable some of them, disable the adapter package (WmCassandraAdapter) first. Otherwise, errors will be issued when you try to access the remaining enabled user-defined packages.

To disable a package

1. Open Integration Server Administrator if it is not already open.

2. In the **Packages** menu of the navigation area, click **Management**.

3. Click **Yes** in the **Enabled** column for the package that you want to disable. The server issues a prompt to verify that you want to disable the package. Click **OK** to disable the package. When the package is disabled, the server displays **No** in the **Enabled** column.

A disabled adapter will:

- Remain disabled until you explicitly enable it using Integration Server Administrator.
- Not be listed in **Designer**.

Importing and Exporting Packages

You can import and export packages using Designer. Exporting allows you to export the package to a .zip file and save it to your hard drive. The .zip file can then be imported for use by another package.

Important:

Do not rename packages that you export; the rename function is comparable to moving a package, and when you import the renamed package, you can lose any triggers, and connections, associated with this package.

For details about importing and exporting packages, see the *webMethods Service Development Help* for your release.

Group Access Control

To control which groups have access to which adapter services, use access control lists (ACLs). For example, you can use ACLs to prevent one development group from inadvertently updating the work of another group, or to allow or deny access to services that are restricted to one group but not to others.

For information about assigning and managing ACLs, see the *webMethods Service Development Help* for your release.

4 Adapter for Apache Cassandra Connections

Overview of Adapter for Apache Cassandra Connections

This chapter describes how to configure and manage Adapter for Apache Cassandra connections. For more information about how adapter connections work, see "Adapter Connections" on page 12.

Before Configuring or Managing Adapter Connection

Perform the following steps before configuring or managing adapter connections.

To prepare to configure or manage adapter connections

1. Install webMethods Integration Server and Adapter for Apache Cassandra on the same machine. For details, see "Overview of installing and uninstalling Adapter for Apache Cassandra" on page 20.

2. Make sure you have Integration Server administrator privileges so that you can access Adapter for Apache Cassandra's administrative screens. For information about setting user privileges, see the *webMethods Integration Server Administrator's Guide* for your release.

3. Start your Integration Server and Integration Server Administrator, if they are not already running.

4. Using Integration Server Administrator, make sure the WmCassandraAdapter package is enabled. For instructions, see "Enabling Packages" on page 25.

5. Using Designer, create a user-defined package to contain the connection, if you have not done so. For more information about managing packages for the adapter, see "Overview of Package Management" on page 24.

Configuring Adapter for Apache Cassandra Connection

When you configure Adapter for Apache Cassandra, you specify information that Integration Server uses to connect to Apache Cassandra. You can configure Adapter for Apache Cassandra connections manually using the Integration Server Administrator screen.

To configure an adapter connection

1. In the **Adapters** menu of Integration Server Administrator's navigation area, click **webMethods Adapter for Apache Cassandra**.

2. On the Connections screen, click **Configure New Connection**.

3. On the Connection Types screen, click **Apache Cassandra Connection** to display the Configure Connection Type screen.

4. In the **webMethods Adapter for Apache Cassandra** section, use the following fields:

Field	Description/Action
Package	The package in which to create the connection. You must create the package using Designer before you can specify the package using this parameter. For general information about creating packages, see the *webMethods Service Development Help* for your release. **Note:** Configure the connection in a user-defined package rather than in the adapter's package. For other important considerations when creating packages for Adapter for Apache Cassandra, see "Adapter for Apache Cassandra Package Management" on page 24
Folder Name	Specifies the folder in which you create the connection.
Connection Name	Specifies the name you want to give to the connection. Connection names cannot have spaces or use special characters reserved by Integration Server and Designer. For more information about the use of special characters in package, folder, and element names, see the *webMethods Service Development Help* for your release.

5. In the **Connection Properties** section, use the following fields:

Field	Description/Action
Cluster Client Name	The name of the client connection. **Note:** ■ This client connection name is not the Cassandra cluster name, but rather a name assigned to the Cassandra Client Connection. ■ You cannot create multiple connections with same Cluster Client Name.
Contact Point(s)	Defines the points to which the list of Cassandra hosts connects. The format is provided as follows: `host1:port1,host2:port2,….` **Note:** Be sure not to use the same cluster name in more than one connection.
Enable Metrics	Enables the metrics for the connections.
Enable JMX Metrics	Enables the JMX reporting metrics for the connections.
Read Pooling Options from Config file	Reads the pooling options from connection.properties file.

Field	Description/Action
Read Protocol Options from Config file	Reads the protcol options from connection.properties file.
Read Load balancing Options from Config file	Reads the load balancing options from connection.properties file.
Read Policies from Config file	Reads the policies options from connection.properties file.
Read Socket Options from Config file	Reads the socket options from connection.properties file.
Read Query Options from Config file	Reads the query options from connection.properties file.

For more information regarding the supported connection properties for Cassandra, see "Supported Backed Connection Properties" on page 34

6. In the **Connection Management Properties** section, use the following fields:

Field	Description/Action
Enable Connection Pooling	Enables the connection to use connection pooling. For more information about connection pooling, see "Adapter Connections" on page 12. **Note:** If you plan to enable connection pooling in a clustered environment, consider the connection pool size.
Minimum Pool Size	If connection pooling is enabled, this field specifies the number of connections to create when the connection is enabled. The adapter will keep open the number of connections you configure here regardless of whether these connections become idle.
Maximum Pool Size	If connection pooling is enabled, this field specifies the maximum number of connections that can exist at one time in the connection pool.
Pool Increment Size	If connection pooling is enabled, this field specifies the number of connections by which the pool will be incremented if connections are needed, up to the maximum pool size.
Block Timeout	If connection pooling is enabled, this field specifies the number of milliseconds that Integration Server will wait to obtain a connection before it times out and returns an error. For example, you have a

Field	Description/Action
	pool with **Maximum Pool Size** of 20. If you receive 30 simultaneous requests for a connection, 10 requests will be waiting for a connection from the pool. If you set the **Block Timeout** to 5000, the 10 requests will wait for a connection for 5 seconds before they time out and return an error. If the services using the connections require 10 seconds to complete and return connections to the pool, the pending requests will fail and return an error message stating that no connections are available. If you set the **Block Timeout** value too high, you may encounter problems during error conditions. If a request contains errors that delay the response, other requests will not be sent. This setting should be tuned in conjunction with the **Maximum Pool Size** to accommodate such bursts in processing.
Expire Timeout	If connection pooling is enabled, this field specifies the number of milliseconds that an inactive connection can remain in the pool before it is closed and removed from the pool. The connection pool will remove inactive connections until the number of connections in the pool is equal to the **Minimum Pool Size**. The inactivity timer for a connection is reset when the connection is used by the adapter. If you set the **Expire Timeout** value too high, you may have a number of unused inactive connections in the pool. This consumes local memory and a connection on your backend resource. This could have an adverse effect if your resource has a limited number of connections. If you set the **Expire Timeout** value too low, performance could degrade because of the increased activity of creating and closing connections. This setting should be tuned in conjunction with the **Minimum Pool Size** to avoid excessive opening/closing of connections during normal processing.
Startup Retry Count	The number of times that the system should attempt to initialize the connection pool at startup if the initial attempt fails. The default is `0`.
Startup Backoff Timeout	The number of seconds that the system should wait between attempts to initialize the connection pool.

7. Click **Save Connection**.

The connection you created appears on the adapter's Connections screen and in Designer.

You can enable a connection only if the parameters for the connection are valid.

Note:

- For connection pooling the Adapter for Apache Cassandra does not create these many Cassandra client Objects even if you configure these options. Hence, the DataStax's

team recommends you to have only one Cassandra client Object per JVM as the connection object is heavy.

- If you need any different configuration settings for multiple Adapter for Apache Cassandra connections, then you have to prepend the cluster name to respective configuration properties in the connection.properties file. For example, *clustername*.cassandra.codecRegistry.typecodecs = xxxxxxxxxxxxxx.

Supported Backed Connection Properties

CodecRegistry

Properties	Action/Description
cassandra.codecRegistry.typecodecs=	Configuration directory.

PoolinglOptions(Options related to connection pooling)

Properties	Action/Description
cassandra.poolingOptions. hearbeatIntervalSeconds	Sets the heart beat interval. Later, a message is sent on an idle connection to make sure it is still alive.
cassandra.poolingOptions. idleTimeoutSeconds=	Sets the timeout before an idle connection is removed.
cassandra.poolingOptions. poolTimeoutMillis=	Sets the timeout when trying to acquire a connection from a host's pool.
cassandra.poolingOptions. coreConnectionsPerHost.local=	Sets the core and maximum number of connections per host with HostDistance set to LOCAL in one call.
cassandra.poolingOptions. maxConnectionsPerHost.local=	
cassandra.poolingOptions. coreConnectionsPerHost.remote=#	Sets the core and maximum number of connections per host with HostDistance set to REMOTE in one call.
cassandra.poolingOptions. maxConnectionsPerHost.remote=	

ProtocolOptions(Options of the Cassandra native binary protocol)

Properties	Action/Description
cassandra.protocolOptions.port=	Port to be used for binary protocol.
cassandra.protocolOptions. authProvider=	AuthProvider to be used for authentication against the Cassandra nodes.
cassandra.protocolOptions. username=	Username for the com.datastax.driver.core. PlainTextAuthProvider.
cassandra.protocolOptions. password=	Password for the com.datastax.driver.core. PlainTextAuthProvider.
cassandra.protocolOptions. sslOption=	Specify the implementation class of com.datastax.driver.core.SSLOptions interface.
cassandra.protocolOptions.version=	Specify the protocol version. If the version is null, then the biggest version supported by the first node to which the driver connects to is used.
cassandra.protocolOptions. compression=	Sets the compression to be used. By default, the compression is not used.
cassandra.protocolOptions. maxSchemaAgreement=	Specify the maximum wait time in seconds for schema agreement before returning from a DDL query.

QueryOptions(Options related to defaults for individual queries)

Properties	Action/Description
cassandra.queryOptions.consistency=	Sets the default consistency level to use queries.
cassandra.queryOptions.serialConsistency=	Sets the default serial consistency level to use queries.
cassandra.queryOptions.fetchSize=	Sets the default fetch size value to use SELECT queries.
cassandra.queryOptions.idemPotence=	Sets the default idempotence for queries.

Properties	Action/Description
cassandra.queryOptions.metadata=	Specifies if the client-side token and schema metadata is to be enabled or not. By default, this feature is enabled. Some applications might want to disable it in order to eliminate the overhead of querying the metadata and building its client-side representation.
cassandra.queryOptions. maxPendingRefreshNodeListRequests=	Sets the default window size in milliseconds, which is used to debounce the node list refresh requests.
cassandra.queryOptions. maxPendingRefreshNodeRequests=	The default window size in milliseconds used to debounce schema refresh requests.
cassandra.queryOptions. maxPendingRefreshSchemaRequests=	Sets the maximum number of schema refresh requests that the control connection can accumulate before executing them.
cassandra.queryOptions. refreshNodeListIntervalMillis=	Sets the default window size in milliseconds which is used to debounce the node list refresh requests.
cassandra.queryOptions. refreshNodeIntervalMillis=	Sets the default window size in milliseconds which is used to debounce node refresh requests.
cassandra.queryOptions. refreshSchemaIntervalMillis=	Sets the maximum number of node refresh requests that the control connection can accumulate before executing them.
cassandra.queryOptions.reprepareOnUp=	Specify whether the driver should re-prepare all cached prepared statements on a host when it marks it as back up. By default, this feature is enabled.
cassandra.queryOptions.prepareOnAllHosts=	Specify whether the driver should prepare statements on all hosts in the cluster.

SocketOptions(Options to configure low-level socket options for the connections kept to the Cassandra hosts)

Properties	Action/Description
cassandra.socketOptions. connectTimeoutMillis=	Sets the connection timeout in milliseconds.
cassandra.socketOptions.	Sets the per-host read timeout in milliseconds.

Properties	Action/Description
readTimeoutMillis=	
cassandra.socketOptions.reuseAddress=	Specifies whether to enable reuse-address. It can be either true or false.
cassandra.socketOptions.keepAlive=	Specifies whether to enable TCP keepalive. It can be either true or false.
cassandra.socketOptions.soLinger=	Sets the linger-on-close timeout. By default, this option is not set by the driver. The actual value is the default from the underlying Netty transport (Java NIO or native epoll).
cassandra.socketOptions.tcpNoDelay=	Specifies whether to disable Nagle's algorithm. By default, this option is set to true (Nagle disabled).
cassandra.socketOptions.receiveBufferSize=	Sets a hint to the size of the underlying buffers for incoming network Input/Output. The value given is an integer value.
cassandra.socketOptions.sendBufferSize=	Sets a hint to the size of the underlying buffers for outgoing network Input/Output. The value given is an integer value.

Policies(Policy that decides how often the reconnection to a dead node is attempted)

Properties	Action/Description
cassandra.policies.reconnection.type=	Type of ReconnectionPolicy either com.datastax.driver.core. policies.ConstantReconnectionPolicy or com.datastax.driver.core. policies.ExponentialReconnectionPolicy.
cassandra.policies.reconnection.delay=	The constant/base delay in milliseconds to be used for this policy.
cassandra.policies.reconnection. maxDelay=	The maximum delay in milliseconds between the number of reconnecting attempts for this policy.
cassandra.policies.adresstranslator=	Translates IP addresses received from Cassandra nodes into locally queriable addresses. Specify the implementation class of com.datastax.driver.core.

Properties	Action/Description
	policies.AddressTranslator.
cassandra.policies.retry=	A policy that defines a default behavior to adopt when a request fails. Specify the implementation class of com.datastax.driver.core.policies.RetryPolicy.
cassandra.policies.timestamp=	Generates client-side, microsecond-precision query timestamps. Specify the implementation class of com.datastax.driver.core.TimestampGenerator.
cassandra.policies.speculative=	The policy that decides if the driver will send speculative queries to the next hosts when the current host takes too long to respond.
cassandra.policies.speculative. const.DelayMillis=	The delay between each speculative execution. The value must be strictly positive.
cassandra.policies.speculative. const.maxExecutions=	The number of speculative executions. The value must be an integer and strictly positive.
cassandra.policies.speculative. percentile.tracker=	The component that will record latencies. It will get registered with the cluster when this policy initializes.
cassandra.policies.speculative. percentile.percentile=	The percentile that a request's latency must fall into is considered slow. For example, 99.0.
cassandra.policies.speculative. percentile.maxExecutions=	The maximum number of speculative executions that is triggered for a given request. The request does not include the initial, normal requests. The Value must be strictly positive.
cassandra.policies.speculative. percentile. highestTrackableLatencyMillis=	The highest expected latency. If a higher value is reported, it will be ignored and a warning will be logged.
cassandra.policies.speculative. percentile. numberOfSignificantValueDigits=	Sets the number of significant decimal digits to which histograms maintains the value resolution and separation.
cassandra.policies.speculative. percentile.minRecordedValues=	Sets the minimum number of values that is recorded for a host before you consider the sample size significant.

Properties	Action/Description
cassandra.policies.speculative. percentile.intervalMs=	Sets the time interval over which samples are recorded.

Policies/LoadBalancing

Properties	Action/Description
cassandra.policies.loadbalancing=	The policy that decides the respective Cassandra hosts to contact for each new query. Specify implementation classes of com.datastax.driver.core.policies. LoadBalancingPolicy with comma separated.

com.datastax.driver.core.policies.DCAwareRoundRobinPolicy

Properties	Action/Description
cassandra.policies.loadbalancing. dcaware.localdc=	Sets the name of the datacenter that is considered "local" by the policy.
cassandra.policies.loadbalancing. dcaware.hostPerRemoteDc=	Sets the number of hosts per remote datacenter that the policy should consider.
cassandra.policies.loadbalancing. dcaware.allowhostPerRemote=	Allows the policy to return remote hosts when building query plans for queries having consistency level LOCAL_ONE or LOCAL_QUORUM.

com.datastax.driver.core.policies.TokenAwarePolicy

Properties	Action/Description
cassandra.policies.loadbalancing. token.shufflereplicas=	Specifies whether to shuffle the replicas returned by getRoutingKey. Note that setting this parameter to true might decrease the effectiveness of caching (especially at consistency level ONE), since the same row will be retrieved from any replica (instead of only the "primary" replica without shuffling). On the other hand, shuffling will better distribute writes, and can alleviate hotspots caused by "fat" partitions.

com.datastax.driver.core.policies.WhiteListPolicy

Properties	Action/Description
cassandra.policies.loadbalancing. whilelist.addresses=	Specify the white listed hosts.

com.datastax.driver.core.policies.LatencyAwarePolicy

Properties	Action/Description
cassandra.policies.loadbalancing. latency.exclusionThreshold=	Sets the exclusion threshold to use for the resulting latency aware policy.
cassandra.policies.loadbalancing. latency.scale=	Sets the scale for the resulting latency aware policy.
cassandra.policies.loadbalancing. latency.retryPeriod=	Sets the retry period for the resulting latency aware policy.
cassandra.policies.loadbalancing. latency.minMeasure=	Sets the minimum number of measurements per-host to consider the resulting latency aware policy.
cassandra.policies.loadbalancing. latency.updateRate=	Sets the update rate for the resulting latency aware policy.

com.datastax.driver.core.policies.ErrorAwarePolicy

Properties	Action/Description
cassandra.policies.loadbalancing. error.retryPeriodNanos=	Defines the time during which a host is excluded by the policy once it has exceeded. By default, the value for the retry period is 2 minutes.
cassandra.policies.loadbalancing. error.maxErrorPerMinute=	Defines the maximum number of errors allowed per minute for each host. By default, the value for the threshold is 1.

5 Adapter Services

Overview of Adapter Services

This chapter describes how to configure and manage the services of Adapter for Apache Cassandra. For detailed descriptions of the available Adapter for Apache Cassandra services, see "Adapter Services" on page 13.

Before Managing Adapter Services

Perform the following steps before configuring or managing adapter services.

To prepare to configure or manage Adapter for Apache Cassandra services

1. Start the Integration Server and Integration Server Administrator, if they are not already running.

2. Make sure you have administrator privileges to access the Adapter for Apache Cassandra administrative screens. For information about setting user privileges, see the *webMethods Integration Server Administrator's Guide* for your release.

3. Using Integration Server Administrator, make sure the WmCassandraAdapter package is enabled. For instructions, see "Enabling Packages" on page 25.

4. Using Integration Server Administrator, configure an adapter connection with the adapter service. For instructions, see "Overview of Adapter for Apache Cassandra Connections" on page 30.

   Note:
   Integration Server provides built-in services that you can use at design time to change the connection associated with an adapter service. For more information, see "Change the Connection Associated with an Adapter Service at Design Time" on page 15.

5. Start Designer if it is not already running.

6. Using Designer, create a user-defined package to contain the services, if you have not already done so. When you configure adapter services, you must always define them in user-defined packages rather than in the WmCassandraAdapter package. For more information about managing packages for the adapter, see "Overview of Package Management" on page 24.

Configuring Insert Service

An Insert service inserts a new record into the table. You configure Insert service using Designer. For more information about adapter services, see "Using Adapter Services" on page 14.

Read the section "Before Managing Adapter Services" on page 42, before you configure adapter services.

To configure an adapter service using the Insert template

1. In Designer, right-click the folder in which the service is to be contained and select **New** > **Adapter Service**.

2. Select the parent namespace, type a name for the adapter service, and click **Next**.

3. Select **webMethods Adapter for Apache Cassandra** as the adapter type and click **Next**.

4. Select the appropriate **Adapter Connection Name** and click **Next**.

5. From the list of available templates, select the **Insert** template and click **Finish**.

   The adapter service editor for the selected adapter service appears. You can select the **Adapter Settings** tab at any time to confirm Adapter Properties such as the **Adapter Name**, **Adapter Connection Name**, and **Adapter Service Template** as necessary.

6. Select the **Table** tab to configure the table and set the fields as follows:

Field	Field Description/Action
Keyspace Name	Specifies the keyspace which is auto-populated based on the created connection.
Table Name	Specify the name of the table.

7. Use the **Columns** tab to provide values for the following fields:

   a. Use the  icon to populate a single column in the table. You can use the  icon to display all columns created in an existing table.

Field	Description/Action
Column Name	The column name in the table.
Column Type	The column data type defined in the corresponding **Column Name** of the table.
Type Definition	Specifies the string representation of Column Type.
Input Field	The output field name.
Input Field Type	The data type of the output field.

8. Use the **Options** tab to provide the appropriate values for the following fields:

Field	Description/Action
Consistency Level	Defines to manage availability versus data accuracy. You can configure consistency for a session or per individual read or write operation. The write consistency levels in strongest-to-weakest order is given as follows: ■ `ALL`: Provides the highest consistency and the lowest availability of any other level. ■ `EACH QUORIUM`: Maintains a strict consistency level at the same level, when used in multiple datacenter clusters. ■ `QUORIUM`: Maintains a strong consistency across the cluster, when used in either single or multiple datacenter clusters. Use `QUORIUM`, if you can tolerate some level of failure. ■ `LOCAL QUORIUM`: Maintains consistency locally. `LOCAL QUORIUM` is used in multiple datacenter clusters with a rack-aware replica placement strategy, such as (NetworkTopologyStatergy), and a properly configured snitch. Can be used with SimpleStrategy. ■ `LOCAL ONE`: In a multiple datacenter clusters, the consistency level cross-DC traffic is not desirable when compared to the consistency level of ONE. `LOCAL ONE` accomplishes this. Use this consistency level in an offline datacenter to prevent automatic connection to online nodes in other datacenters if an offline node goes down. ■ `THREE`: Satisfies the needs of most users because consistency requirements are not stringent. ■ `ANY`: Provides low latency and a guarantee that a write never fails. Delivers the lowest consistency and highest availability. The read consistency levels in strongest-to-weakest order is given as follows: ■ `ALL`: Provides the highest consistency of all levels and the lowest availability of all levels. ■ `QUORIUM`: Maintains strong consistency across the cluster, when used in single or multiple datacenter clusters. Ensures strong consistency if you can tolerate some level of failure. ■ `LOCAL QUORIUM`: Used in multiple datacenter clusters with a rack-aware replica placement strategy (NetworkTopologyStrategy) and a properly configured snitch. Fails when used with SimpleStrategy. ■ `THREE`: Provides the highest availability of all the levels if you can tolerate a comparatively high probability of stale data being read. The

Field	Description/Action
	replicas contacted for reads may not always have the most recent write. ■ `LOCAL ONE`: In a multiple datacenter clusters, the consistency level cross-DC traffic is not desirable when compared to the consistency level of ONE. `LOCAL ONE` accomplishes this. Use this consistency level in an offline datacenter to prevent automatic connection to online nodes in other datacenters if an offline node goes down. ■ `SERIAL`: Use `SERIAL` to read the latest value of a column after a user has invoked a lightweight transaction to write to the column. Cassandra then checks the inflight lightweight transaction for updates and, if found, returns the latest data. ■ `LOCAL SERIAL`: Used to achieve linearizable consistency for lightweight transactions.
Enabling Tracing	Define to enable and disable the tracing for transactions on all nodes in the cluster. Tracing is enabled to troubleshoot performance problems.
If Not Exists	Checks if there are any duplicates with same primary key column value.

9. To verify input or output information for the service, use the **Input/Output** tab as required.

   The below fields are auto-populated in the input section of **Input/Output** tab:

Field	Description/Action
timestamp	Specify the time of insertion of the record.
Time to live(ttl)	Specifies the setting time for data in a column to expire. The value for the field is accepted in milliseconds.

Note:
The remaining fields in the input section are auto-populated based on the columns selected in **Columns** tab.

10. From the **File** menu, select **Save**.

Configuring Delete Service

A Delete service deletes the record from the Cassandra table. You configure Delete service using Designer. For more information about adapter services, see "Using Adapter Services" on page 14.

Read the section "Before Managing Adapter Services" on page 42, before you configure adapter services.

To configure an adapter service using the Delete template

1. In Designer, right-click the folder in which the service is to be contained and select **New** > **Adapter Service**.

2. Select the parent namespace, type a name for the adapter service, and click **Next**.

3. Select **webMethods Adapter for Apache Cassandra** as the adapter type and click **Next**.

4. Select the appropriate **Adapter Connection Name** and click **Next**.

5. From the list of available templates, select the **Delete** template and click **Finish**.

   The adapter service editor for the selected adapter service appears. You can select the **Adapter Settings** tab at any time to confirm Adapter Properties such as the **Adapter Name**, **Adapter Connection Name**, and **Adapter Service Template** as necessary.

6. Select the **Table** tab to configure the table and set the fields as follows:

Field	Field Description/Action
Keyspace Name	Specifies the keyspace which is auto-populated based on the created connection.
Table Name	Specify the name of the table.

7. Use the **Columns** tab to provide values for the following fields:

   a. Use the  icon to populate a single column in the table. You can use the  icon to display all columns created in an existing table.

Field	Description/Action
Column Name	The column name in the table.
Column Type	The column data type defined in the corresponding **Column Name** of the table.
Type Definition	Specifies the string representation of Column Type.

8. Use the **Options** tab to provide the appropriate values for the following fields:

Field	Description/Action
Consistency Level	Defines to manage availability versus data accuracy. You can configure consistency for a session or per individual read or write operation.

Field	Description/Action
	The write consistency levels in strongest-to-weakest order is given as follows: ■ `ALL`: Provides the highest consistency and the lowest availability of any other level. ■ `EACH QUORIUM`: Maintains a strict consistency level at the same level, when used in multiple datacenter clusters. ■ `QUORIUM`: Maintains a strong consistency across the cluster, when used in either single or multiple datacenter clusters. Use `QUORIUM`, if you can tolerate some level of failure. ■ `LOCAL QUORIUM`: Maintains consistency locally. `LOCAL QUORIUM` is used in multiple datacenter clusters with a rack-aware replica placement strategy, such as (NetworkTopologyStatergy), and a properly configured snitch. Can be used with SimpleStrategy. ■ `LOCAL ONE`: In a multiple datacenter clusters, the consistency level cross-DC traffic is not desirable when compared to the consistency level of ONE. `LOCAL ONE` accomplishes this. Use this consistency level in an offline datacenter to prevent automatic connection to online nodes in other datacenters if an offline node goes down. ■ `THREE`: Satisfies the needs of most users because consistency requirements are not stringent. ■ `ANY`: Provides low latency and a guarantee that a write never fails. Delivers the lowest consistency and highest availability. The read consistency levels in strongest-to-weakest order is given as follows: ■ `ALL`: Provides the highest consistency of all levels and the lowest availability of all levels. ■ `QUORIUM`: Maintains strong consistency across the cluster, when used in single or multiple datacenter clusters. Ensures strong consistency if you can tolerate some level of failure. ■ `LOCAL QUORIUM`: Used in multiple datacenter clusters with a rack-aware replica placement strategy (NetworkTopologyStrategy) and a properly configured snitch. Fails when used with SimpleStrategy. ■ `THREE`: Provides the highest availability of all the levels if you can tolerate a comparatively high probability of stale data being read. The replicas contacted for reads may not always have the most recent write.

Field	Description/Action
	■ LOCAL ONE: In a multiple datacenter clusters, the consistency level cross-DC traffic is not desirable when compared to the consistency level of ONE. LOCAL ONE accomplishes this. Use this consistency level in an offline datacenter to prevent automatic connection to online nodes in other datacenters if an offline node goes down. ■ SERIAL: Use SERIAL to read the latest value of a column after a user has invoked a lightweight transaction to write to the column. Cassandra then checks the inflight lightweight transaction for updates and, if found, returns the latest data. ■ LOCAL SERIAL: Used to achieve linearizable consistency for lightweight transactions.
EnableTracing	Define to enable and disable the tracing for transactions on all nodes in the cluster. Tracing is enabled to troubleshoot performance problems.
If Exists	Checks if columns value selected in WHERE clause is available and deletes the record and returns the success value as true else returns the value as false.

9. Use the **WHERE** tab to specify the conditions for selecting information:

   a. Select the icon to define the new WHERE clause fields.

   b. Select the **AND** logical operator and specify the following fields:

Field	Name Description/Action
AND	The logical operator.
Column	The name of the column you want to use in the WHERE clause.
Operator	The following are the operators used: ■ = ■ IN
Input Field	The default value is ?, which acts as a placeholder for the variable so that you can set the input variable for that column at run time, or get input external to this adapter service. You can also type a fixed value in this field now or at run time. If you type a fixed value, be sure that it is valid, or an exception will be generated at run time.

   The adapter automatically generates values for the following fields:

Field	Description/Action
Parameter	The unique key for those rows in the above table which takes input at runtime.
Column	The name of the column you need to use in the WHERE clause.
CQL Type	The CQL data type defined in the corresponding **Column Name** of the table.
Input Field Type	The corresponding data type of the **Input Field**.
Input Field	The name of the input field. By default the name combines the values of **Parameter** and **Column** fields. However, you can also choose to specify any custom value.

10. Use the **If Clause** tab to specify the if condition for the following:

    a. Select the icon to define the new If Clause fields.

    b. Select the **AND** logical operator and specify the following fields:

Name	Name Description/Action
AND	The logical operator.
Column	The name of the column you want to use in the If Clause.
Operator	The following are the operators used: ■ = ■ < ■ > ■ <= ■ >= ■ IN ■ CONTAINS ■ CONTAINS KEY
Input Field	The default value is ?, which acts as a placeholder for the variable so that you can set the input variable for that column at run time, or get input external to this adapter service. You can also type a fixed value in this field now or at run time. If you type a fixed value, be sure that it is valid, or an exception will be generated at run time.

The adapter automatically generates values for the following fields:

Field	Description/Action
Parameter	The unique key for those rows in the above table which takes input at runtime.
Column	The name of the column you need to use in the WHERE clause.
CQL Type	The CQL data type defined in the corresponding **Column Name** of the table.
Input Field Type	The corresponding data type of the **Input Field**.
Input Field	The name of the input field. By default the name combines the values of **Parameter** and **Column** fields. However, you can also choose to specify any custom value.

Note:
If Not Exists field cannot be used with **If Clause** tab at the same time.

11. To verify input or output information for the service, use the **Input/Output** tab as required.

The below field is auto-populated in input section of **Input/Output** tab:

Field	Description/Action
timestamp	Specify the time of insertion of the record.

Note:
The remaining fields in the input section are auto-populated based on the columns selected in **WHERE** tab and **If Clause** tab.

The below field is populated in the output section of **Input/Output** tab:

Field	Description/Action
success	Specifies the result of the service. It can either be *true* or *false*.

12. From the **File** menu, select **Save**.

Configuring Update Service

A Update service updates the existing record. You configure Update service using Designer. For more information about adapter services, see "Using Adapter Services" on page 14.

Read the section "Before Managing Adapter Services" on page 42, before you configure adapter services.

To configure an adapter service using the Update template

1. In Designer, right-click the folder in which the service is to be contained and select **New** > **Adapter Service**.

2. Select the parent namespace, type a name for the adapter service, and click **Next**.

3. Select **webMethods Adapter for Apache Cassandra** as the adapter type and click **Next**.

4. Select the appropriate **Adapter Connection Name** and click **Next**.

5. From the list of available templates, select the **Update** template and click **Finish**.

   The adapter service editor for the selected adapter service appears. You can select the **Adapter Settings** tab at any time to confirm Adapter Properties such as the **Adapter Name**, **Adapter Connection Name**, and **Adapter Service Template** as necessary.

6. Select the **Table** tab to configure the table and set the fields as follows:

Field	Field Description/Action
Keyspace Name	Specifies the keyspace which is auto-populated based on the created connection.
Table Name	Specify the name of the table.

7. Use the **Columns** tab to provide values for the following fields:

   a. Use the  icon to populate a single column in the table. You can use the  icon to display all columns created in an existing table.

Field	Description/Action
Column Name	The column name in the table.
Column Type	The column data type defined in the corresponding **Column Name** of the table.
Type Definition	Specifies the string representation of Column Type.
Input Field	The input field name.
Input Field Type	The data type of the input field.

8. Use the **Options** tab to provide the appropriate values for the following fields:

Field	Description/Action
Consistency Level	Defines to manage availability versus data accuracy. You can configure consistency for a session or per individual read or write operation. The write consistency levels in strongest-to-weakest order is given as follows: ■ `ALL`: Provides the highest consistency and the lowest availability of any other level. ■ `EACH QUORIUM`: Maintains a strict consistency level at the same level, when used in multiple datacenter clusters. ■ `QUORIUM`: Maintains a strong consistency across the cluster, when used in either single or multiple datacenter clusters. Use `QUORIUM`, if you can tolerate some level of failure. ■ `LOCAL QUORIUM`: Maintains consistency locally. `LOCAL QUORIUM` is used in multiple datacenter clusters with a rack-aware replica placement strategy, such as (NetworkTopologyStatergy), and a properly configured snitch. Can be used with SimpleStrategy. ■ `LOCAL ONE`: In a multiple datacenter clusters, the consistency level cross-DC traffic is not desirable when compared to the consistency level of ONE. `LOCAL ONE` accomplishes this. Use this consistency level in an offline datacenter to prevent automatic connection to online nodes in other datacenters if an offline node goes down. ■ `THREE`: Satisfies the needs of most users because consistency requirements are not stringent. ■ `ANY`: Provides low latency and a guarantee that a write never fails. Delivers the lowest consistency and highest availability. The read consistency levels in strongest-to-weakest order is given as follows: ■ `ALL`: Provides the highest consistency of all levels and the lowest availability of all levels. ■ `QUORIUM`: Maintains strong consistency across the cluster, when used in single or multiple datacenter clusters. Ensures strong consistency if you can tolerate some level of failure. ■ `LOCAL QUORIUM`: Used in multiple datacenter clusters with a rack-aware replica placement strategy (NetworkTopologyStrategy) and a properly configured snitch. Fails when used with SimpleStrategy. ■ `THREE`: Provides the highest availability of all the levels if you can tolerate a comparatively high probability of stale data being read. The

Field	Description/Action
	replicas contacted for reads may not always have the most recent write. ■ `LOCAL ONE`: In a multiple datacenter clusters, the consistency level cross-DC traffic is not desirable when compared to the consistency level of ONE. `LOCAL ONE` accomplishes this. Use this consistency level in an offline datacenter to prevent automatic connection to online nodes in other datacenters if an offline node goes down. ■ `SERIAL`: Use `SERIAL` to read the latest value of a column after a user has invoked a lightweight transaction to write to the column. Cassandra then checks the inflight lightweight transaction for updates and, if found, returns the latest data. ■ `LOCAL SERIAL`: Used to achieve linearizable consistency for lightweight transactions.
Enabling Tracing	Define to enable and disable the tracing for transactions on all nodes in the cluster. Tracing is enabled to troubleshoot performance problems.
If Exists	Checks if there are any duplicates with same column value.

9. Use the **WHERE** tab to specify the valid conditions for selecting information:

   a. Select the icon to define the new WHERE clause fields.

   b. Select the **AND** logical operator and specify the following fields:

Name	Name Description/Action
AND	The logical operator.
Column	The name of the column you want to use in the WHERE clause.
Operator	The following are the operators used: ■ = ■ IN
Input Field	The default value is ?, which acts as a placeholder for the variable so that you can set the input variable for that column at run time, or get input external to this adapter service. You can also type a fixed value in this field now or at run time. If you type a fixed value, be sure that it is valid, or an exception will be generated at run time.

   The adapter automatically generates values for the following fields:

Field	Description/Action
Parameter	The unique key for those rows in the above table which takes input at runtime.
Column	The name of the column you need to use in the WHERE clause.
CQL Type	The CQL data type defined in the corresponding **Column Name** of the table.
Input Field Type	The corresponding data type of the **Input Field**.
Input Field	The name of the input field. By default the name combines the values of **Parameter** and **Column** fields. However, you can also choose to specify any custom value.

10. Use the **If Clause** tab to specify the if condition for the following:

    a. Select the icon to define the new If Clause fields.

    b. Select the **AND** logical operator and specify the following fields:

Name	Name Description/Action
AND	The logical operator.
Column	The name of the column you want to use in the WHERE clause.
Operator	The following are the operators used: ■ = ■ < ■ > ■ <= ■ >= ■ IN ■ CONTAINS ■ CONTAINS KEY
Input Field	The default value is ?, which acts as a placeholder for the variable so that you can set the input variable for that column at run time, or get input external to this adapter service. You can also type a fixed value in this field now or at run time. If you type a fixed value, be sure that it is valid, or an exception will be generated at run time.

The adapter automatically generates values for the following fields:

Field	Description/Action
Parameter	The unique key for those rows in the above table which takes input at runtime.
Column	The name of the column you need to use in the WHERE clause.
CQL Type	The CQL data type defined in the corresponding **Column Name** of the table.
Input Field Type	The corresponding data type of the **Input Field**.
Input Field	The name of the input field. By default the name combines the values of **Parameter** and **Column** fields. However, you can also choose to specify any custom value.

11. To verify input or output information for the service, use the **Input/Output** tab as required.

    The below fields are auto-populated in the input section of **Input/Output** tab:

Field	Description/Action
timestamp	Specify the time of insertion of the record.
Time to live(ttl)	Specifies the setting time for data in a column to expire. The value for the field is accepted in milliseconds.

Note:
The remaining fields in the input section are auto-populated based on the columns selected in **Columns**, **WHERE**, and **If Clause** tabs.

12. From the **File** menu, select **Save**.

Configuring Query Service

A Query service retrieves the record from the Cassandra table. You configure Query service using Designer. For more information about adapter services, see "Using Adapter Services" on page 14.

Read the section "Before Managing Adapter Services" on page 42, before you configure adapter services.

To configure an adapter service using the Query template

1. In Designer, right-click the folder in which the service is to be contained and select **New** > **Adapter Service**.

2. Select the parent namespace, type a name for the adapter service, and click **Next**.

3. Select **webMethods Adapter for Apache Cassandra** as the adapter type and click **Next**.

4. Select the appropriate **Adapter Connection Name** and click **Next**.

5. From the list of available templates, select the **Query** template and click **Finish**.

   The adapter service editor for the selected adapter service appears. You can select the **Adapter Settings** tab at any time to confirm Adapter Properties such as the **Adapter Name**, **Adapter Connection Name**, and **Adapter Service Template** as necessary.

6. Select the **Table** tab to configure the table and set the fields as follows:

Field	Field Description/Action
Keyspace Name	Specifies the keyspace which is auto-populated based on the created connection.
Table Name	Specify the name of the table.

7. Use the **Columns** tab to provide values for the following fields:

   a. Use the icon to populate a single column in the table. You can use the icon to display all columns created in an existing table.

Field	Description/Action
Column Name	The column name in the table.
Column Type	The column data type defined in the corresponding **Column Name** of the table.
Type Definition	Specifies the string representation of Column Type.
Output Field	The output field name.
Output Field Type	The Java type that corresponds to the output field.

8. Use the **Options** tab to provide the appropriate values for the following fields:

Field	Description/Action
Fetch Size	Limits the number of records for a table, if the records exceeds by the given size. The default value is `-1`.

Field	Description/Action
Allow Filtering	Provides the capability to query the clustering columns using any condition.
Consistency Level	Defines to manage availability versus data accuracy. You can configure consistency for a session or per individual read or write operation. The write consistency levels in strongest-to-weakest order is given as follows: ■ `ALL`: Provides the highest consistency and the lowest availability of any other level. ■ `EACH QUORIUM`: Maintains a strict consistency level at the same level, when used in multiple datacenter clusters. ■ `QUORIUM`: Maintains a strong consistency across the cluster, when used in either single or multiple datacenter clusters. Use `QUORIUM`, if you can tolerate some level of failure. ■ `LOCAL QUORIUM`: Maintains consistency locally. `LOCAL QUORIUM` is used in multiple datacenter clusters with a rack-aware replica placement strategy, such as (NetworkTopologyStatergy), and a properly configured snitch. Can be used with SimpleStrategy. ■ `LOCAL ONE`: In a multiple datacenter clusters, the consistency level cross-DC traffic is not desirable when compared to the consistency level of ONE. `LOCAL ONE` accomplishes this. Use this consistency level in an offline datacenter to prevent automatic connection to online nodes in other datacenters if an offline node goes down. ■ `THREE`: Satisfies the needs of most users because consistency requirements are not stringent. ■ `ANY`: Provides low latency and a guarantee that a write never fails. Delivers the lowest consistency and highest availability. The read consistency levels in strongest-to-weakest order is given as follows: ■ `ALL`: Provides the highest consistency of all levels and the lowest availability of all levels. ■ `QUORIUM`: Maintains strong consistency across the cluster, when used in single or multiple datacenter clusters. Ensures strong consistency if you can tolerate some level of failure. ■ `LOCAL QUORIUM`: Used in multiple datacenter clusters with a rack-aware replica placement strategy (NetworkTopologyStrategy) and a properly configured snitch. Fails when used with SimpleStrategy.

Field	Description/Action
	■ THREE: Provides the highest availability of all the levels if you can tolerate a comparatively high probability of stale data being read. The replicas contacted for reads may not always have the most recent write. ■ LOCAL ONE: In a multiple datacenter clusters, the consistency level cross-DC traffic is not desirable when compared to the consistency level of ONE. LOCAL ONE accomplishes this. Use this consistency level in an offline datacenter to prevent automatic connection to online nodes in other datacenters if an offline node goes down. ■ SERIAL: Use SERIAL to read the latest value of a column after a user has invoked a lightweight transaction to write to the column. Cassandra then checks the inflight lightweight transaction for updates and, if found, returns the latest data. ■ LOCAL SERIAL: Used to achieve linearizable consistency for lightweight transactions.
Limit	Limits the number of records to be retrieved by providing a definite value.
Read Timeout in Millis	Specify the read time out value in milli seconds. This value is the amount of time Adapter for Apache Cassandra waits for the response from the Cassandra node to execute before stopping the Query service.
Enabling Tracing	Define to enable and disable the tracing for transactions on all nodes in the cluster. Tracing is enabled to troubleshoot performance problems.

9. Use the **WHERE** tab to specify the valid conditions for selecting information:

   a. Select the icon to define the new WHERE clause fields.

   b. Select the **AND** logical operator and specify the following fields:

Name	Name Description/Action
AND	The logical operator.
Column	The name of the column you want to use in the WHERE clause.
Operator	The following are the operators used: ■ = ■ < ■ > ■ <=

Name	Name Description/Action
	■ >= ■ IN ■ CONTAINS ■ CONTAINS KEY
Input Field	The default value is ?, which acts as a placeholder for the variable so that you can set the input variable for that column at run time, or get input external to this adapter service. You can also type a fixed value in this field now or at run time. If you type a fixed value, be sure that it is valid, or an exception will be generated at run time.

The adapter automatically generates values for the following fields:

Field	Description/Action
Parameter	The unique key for those rows in the above table which takes input at runtime.
Column	The name of the column you need to use in the WHERE clause.
CQL Type	The CQL data type defined in the corresponding **Column Name** of the table.
Input Field Type	The corresponding data type of the **Input Field**.
Input Field	The name of the input field. By default the name combines the values of **Parameter** and **Column** fields. However, you can also choose to specify any custom value.

10. Use the **OrderBy** tab to define the values for the following fields:

Field	Description/Action
Column Name	Displays the columns which contains only the cluster key type.
Sort Order	Displays the records in correct ascending or descending order. Select either **ASC** or **DES**. By default, **ASC** is selected.

11. To verify input or output information for the service, use the **Input/Output** tab as required.

    Here, the columns selected in WHERE tab is auto-populated.

12. From the **File** menu, select **Save**.

Configuring Query with Prepared Statement Service

A Query with Prepared Statement service retrieves the record from the Cassandra table using a prepared statement. You configure Query with Prepared Statement service using Designer. For more information about adapter services, see "Using Adapter Services" on page 14.

Read the section "Before Managing Adapter Services" on page 42, before you configure adapter services.

To configure an adapter service using the Delete template

1. In Designer, right-click the folder in which the service is to be contained and select **New** > **Adapter Service**.

2. Select the parent namespace, type a name for the adapter service, and click **Next**.

3. Select **webMethods Adapter for Apache Cassandra** as the adapter type and click **Next**.

4. Select the appropriate **Adapter Connection Name** and click **Next**.

5. From the list of available templates, select the **Query with Prepared Statement** template and click **Finish**.

   The adapter service editor for the selected adapter service appears. You can select the **Adapter Settings** tab at any time to confirm Adapter Properties such as the **Adapter Name**, **Adapter Connection Name**, and **Adapter Service Template** as necessary.

6. Select the **Table** tab to configure the table and set the fields as follows:

Field	Field Description/Action
Keyspace Name	Specifies the keyspace which is auto-populated based on the created connection.
Table Name	Specify the name of the table.

7. Use the **Columns** tab to provide values for the following fields:

   a. Use the  icon to populate a single column in the table. You can use the  icon to display all columns created in an existing table.

Field	Description/Action
Column Name	The column name in the table.
Column Type	The column data type defined in the corresponding **Column Name** of the table.

Field	Description/Action
Type Definition	Specifies the string representation of Column Type.
Output Field	The output field name.
Output Field Type	The Java type that corresponds to the output field.

8. Use the **Options** tab to provide the appropriate values for the following fields:

Field	Description/Action
Allow Filtering	Provides the capability to query the clustering columns using any condition.
Limit	Limits the number of records to be retrieved by providing a definite value.

9. Use the **WHERE** tab to specify the valid conditions for selecting information:

   a. Select the icon to define the new WHERE clause fields.

   b. Select the **AND** logical operator and specify the following fields:

Name	Name Description/Action
AND	The logical operator.
Column	The name of the column you want to use in the WHERE clause.
Operator	The following are the operators used: ■ = ■ < ■ > ■ <= ■ >= ■ IN ■ CONTAINS ■ CONTAINS KEY
Input Field	The default value is ?, which acts as a placeholder for the variable so that you can set the input variable for that column at run time, or get input external to this adapter service. You can also type a fixed value in this field

Name	Name Description/Action
	now or at run time. If you type a fixed value, be sure that it is valid, or an exception will be generated at run time.

The adapter automatically generates values for the following fields:

Field	Description/Action
Parameter	The unique key for those rows in the above table which takes input at runtime.
Column	The name of the column you need to use in the WHERE clause.
CQL Type	The CQL data type defined in the corresponding **Column Name** of the table.
Input Field Type	The corresponding data type of the **Input Field**.
Input Field	The name of the input field. By default the name combines the values of **Parameter** and **Column** fields. However, you can also choose to specify any custom value.

Note:
If Not Exists field cannot be used with **If Clause** field at the same time.

10. Use the **Orderby** tab to define the values for the following fields:

Field	Description/Action
Column Name	Displays the columns which contains only the cluster key type.
Sort Order	Displays the records in correct ascending or descending order. Select either **ASC** or **DES**. By default, **ASC** is selected.

11. To verify input or output information for the service, use the **Input/Output** tab as required.

    Here, the columns selected in WHERE tab is auto-populated.

12. From the **File** menu, select **Save**.

Configuring CQL Service

A CQL Statement service executes the configured CQL statements. You configure CQL service using Designer. For more information about adapter services, see , "Using Adapter Services" on page 14.

Read the section "Before Managing Adapter Services" on page 42, before you configure adapter services.

To configure an adapter service using the CQL template

1. In Designer, right-click the folder in which the service is to be contained and select **New** > **Adapter Service**.
2. Select the parent namespace, type a name for the adapter service, and click **Next**.
3. Select **webMethods Adapter for Apache Cassandra** as the adapter type and click **Next**.
4. Select the appropriate **Adapter Connection Name** and click **Next**.
5. From the list of available templates, select the **CQL** template and click **Finish**.

   The adapter service editor for the selected adapter service appears. You can select the **Adapter Settings** tab at any time to confirm Adapter Properties such as the **Adapter Name**, **Adapter Connection Name**, and **Adapter Service Template** as necessary.
6. Use the **CQL** tab to specify the If condition for the following fields:

   a. Select the icon to define the new If Clause fields.

Name	Name Description/Action
Input Field Name	Specify the input field name.
Input CQL Type	Select the Input CQL Type of the input field.
Input Field Type	The Java type that corresponds to the input CQL type.

   The adapter automatically generates values for the following fields:

Field	Description/Action
Output Column Name	The name of any output parameters of the CQL statement.
Output CQL Type	The output parameter of CQL type.
Output Column Type	The Java type that corresponds to the Output CQL Type.

7. To verify input or output information for the service, use the **Input/Output** tab as required.

   Here, the columns selected in the Input Field Name of CQL tab is auto-populated.
8. From the **File** menu, select **Save**.

6 Predefined Health Indicator

Predefined Health Indicator

Microservices Runtime includes predefined health indicators for some of its basic components. The health indicator captures the connection details for all the WmART based adapters at runtime. For more information, see *webMethods Adapter Runtime User's Guide*.

7 Administrator APIs

Administrator APIs

The Administrator APIs are available for Adapter for Apache Cassandra. For more information about Administrator APIs and samples, see *webMethods Adapter Runtime User's Guide*.

8 Configuration Variables Templates for Adapter Assets in Microservices Runtime

Configuration Variables Templates for Adapter Assets in Microservices Runtime

The webMethods Adapter Runtime (ART) asset properties that can be configured from Integration Server Administrator are available in the configuration variables template (`application.properties` file) generated by Microservices Runtime. For more information, see *webMethods Adapter Runtime User's Guide* and *Developing Microservices with webMethods Microservices Runtime*.

9 Data Type Mapping

SQL Data Type to JDBC Data Type Mappings

webMethods Adapter for Apache Cassandra's maps CQL or column type to respective java type as shown in the below table. Adapter for Apache Cassandra expects inputs in its respective java type while inserting or updating values for collection or tuple data types. Also, use the below data types for IN operator of WHERE clause.

Column Type/CQL Type	Java Data Type
ASCII	java.lang.String
BIGINT	java.lang.Long
BLOB	byte [ ]
BOOLEAN	java.lang.Boolean
COUNTER	java.lang.Long
DATE	java.util.Date
DECIMAL	java.math.BigDecimal
DOUBLE	java.lang.Double
FLOAT	java.lang.Float
INET	java.net.InetAddress
INT	java.lang.Integer
LIST	java.util.List
MAP	java.util.Map
SET	java.util.Set
SMALLINT	java.lang.Short
TEXT	java.lang.Strings
TIME	java.lang.Long
TIMESTAMP	java.lang.Long
TIMEUUID	java.util.UUID
TINYINT	java.lang.Byte
TUPLE	Object[]
UDT	com.wm.data.IData
UUID	java.util.UUID

Column Type/CQL Type	Java Data Type
VARCHAR	java.lang.String
VARINT	java.math.BigInteger

Note:
Be sure to pass java.nio.ByteBuffer type values in Object array, only for the WHERE clause with IN operator in BLOB data type.

User Defined Types

webMethods Adapter for Apache Cassandra handles user-defined data type as IData document types. Hence, in the Adapter service templates, input type or output type for UDT column type is considered as a document(com.wm.data.IData).