Using Digital Event Services to Communicate between Software AG Products

Version 10.1

October 2017

This document applies to Software AG Product Suite Version 10.1 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: DEV-AG-101-20171017

Table of Contents

About this Guide

This document gives you an overview of Digital Event Services, which is Software AG's tool for managing simple event-based interactions.

The goal of Digital Event Services is to facilitate the integration between Software AG products and applications by easily allowing them to communicate.

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
Narrowfont	Identifies storage locations for services on webMethods Integration Server, using the convention *folder.subfolder:service* .
UPPERCASE	Identifies keyboard keys. Keys you must press simultaneously are joined with a plus sign (+).
Italic	Identifies variables for which you must supply values specific to your own situation or environment. Identifies new terms the first time they occur in the text.
`Monospace font`	Identifies text you must type or messages displayed by the system.
{ }	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

Software AG Documentation Website

You can find documentation on the Software AG Documentation website at http://documentation.softwareag.com. The site requires Empower credentials. If you do not have Empower credentials, you must use the TECHcommunity website.

Software AG Empower Product Support Website

You can find product information on the Software AG Empower Product Support website at https://empower.softwareag.com.

To submit feature/enhancement requests, get information about product availability, and download products, go to Products.

To get information about fixes and to read early warnings, technical papers, and knowledge base articles, go to the Knowledge Center.

Software AG TECHcommunity

You can find documentation and other technical information on the Software AG TECHcommunity website at http://techcommunity.softwareag.com. You can:

- Access product documentation, if you have TECHcommunity credentials. If you do not, you will need to register and specify "Documentation" as an area of interest.
- Access articles, code samples, demos, and tutorials.
- Use the online discussion forums, moderated by Software AG professionals, to ask questions, discuss best practices, and learn how other customers are using Software AG technology.
- Link to external websites that discuss open standards and web technology.

1 Understanding Digital Event Services

Digital Event Services Overview

Digital Event Services (DES) enables other Software AG products to intercommunicate by exchanging *digital events*. Digital events are typed and serialized data structures that are used to convey or record information about the execution of a runtime. This information can be application information, such as the state of a business process step, including any associated business data, or it can be system information, for example, how much memory and how many threads an application is using.

Any digital event has a *digital event type*. Each digital event type has an associated *digital event type definition* that is created and stored in the digital event type repository for the installation. Digital event types are created by users in a development environment, and subsequently deployed to production installations. The format of the events is based on Google Protocol Buffers. For more information about the digital event type model, see "The Digital Event Type Model" on page 9.

Digital Event Services uses the *publish-subscribe* model. Applications can both *emit* and *subscribe to* streams of events of a given event type. When publishing a digital event, the originating application *emits* the event without considering whether other applications might receive the event, or whether the event might be archived to an event repository. On the other hand, applications that are consumers of events subscribe to digital events of a specific event type.

The destinations, also known as *destination services*, to which events are routed, are not defined at design-time. Instead, a system administrator manages the destination services at run-time. For more information about services, see " Messaging Services and Service Groups" on page 10.

Communication using Digital Event Services can be *persistent* or *non-persistent*. With persistent communication, the delivery of a digital event is guaranteed at least once. With non-persistent communication, the delivery of a digital event is not guaranteed.

A developer may set the *delivery preference* for a given event type, but the DES administrator controls the final *delivery mode*. For more information about delivery modes for Digital Event Services, see "Delivery Modes for Digital Event Services" on page 9.

Support of Digital Event Services

Digital Event Services (DES) is used for communication only between the following products:

- webMethods Integration Server
- Apama
- Software AG MashZone NextGen

- Software AG MashZone NextGen Explorer
- webMethods API Gateway
- webMethods Task Engine

You cannot use DES with other Software AG products. For more information about how to use DES from the perspective of the listed products, see the product documentation of the adopting products.

The Digital Event Type Model

Any digital event has a *digital event type*. A digital event type is a high-level definition of the event format that is independent of the underlying over-the-wire representation of an event. The event type model is designed to support an easy conversion of an event type into other Software AG type languages, such as Integration Server Document Type and Apama Event Processing Language.

For more information about the mapping of events from Integration Server and Apama to Digital Event Services, see *webMethods Integration Server Administrator's Guide* and *Connecting Apama Applications to External Components*, respectively.

Delivery Modes for Digital Event Services

Digital Event Services enables an administrator to set a delivery mode per event type. The delivery mode setting governs the quality of service if the DES runtime stops responding and becomes unavailable.

Digital events are stored in-memory or on-disk until their delivery is acknowledged. To ensure that at least one copy of an event is delivered, administrators can define the event type delivery mode as *persistent*. This means that the events of an event type are stored on-disk. If the runtime where DES is embedded becomes unavailable, the events are resent the next time the runtime starts.

If the delivery mode for an event type is *non-persistent*, events are stored in-memory. If the runtime where DES is embedded becomes unavailable events are not resent.

For information about how to configure delivery mode per event type, see "Configuring Event Type Associations" on page 26. Administrators can define on-disk and in-memory capacity globally and per event type. For more information about configuring in-memory and on-disk capacity, see "Configuring Storage Settings for Digital Event Services" on page 28.

Store-and-Forward Processing

Instead of directly delivering each event to the configured destination services and waiting for each service to acknowledge the event, Digital Event Services (DES) stores the event in an internal queue.

Depending on the delivery mode setting defined for each event type in Command Central, the queue can be held in-memory or on-disk. After an event is added to the queue, DES is ready to accept new events. In the meantime, the queued events are delivered to their destination services by a separate thread in the order in which they were added to the queue.

For more information about how to configure delivery mode per event type, see "Configuring Event Type Associations" on page 26.

Messaging Services and Service Groups

Messaging services are endpoints where events are published. Depending on your application's capabilities to emit or subscribe to events, you can tag the messaging service as either a destination, a source, or both. Digital Event Services (DES) supports the following service types:

- **Universal Messaging services** - use services of this type to send events to or receive events from a Universal Messaging server realm or cluster.
- **Digital Event Persistence services** - use services of this type to send events to a Digital Event Persistence destination.
- **In-Process service** - use this pre-configured service to send and receive events within the same server runtime.

Services are grouped together as a set of one or more services to which events can be sent. One of the services in a service group can be tagged as the source of events for all event types associated with the service group.

Each Digital Event Services runtime contains a default service group, which is sufficient for most use cases. However, you can also define a custom service group for a particular runtime.

For more information about how to configure services, see "Configuring Services" on page 14. For more information about how to group those services together, see "Configuring Service Groups" on page 23.

Considerations when Using Service Groups

- If a service group contains no source service and your application creates a subscription for this service group, no events are received. The subscription remains inactive until a source service is configured.

- If an application emits events to a service group that has no destination service configured, then the emitted events are not received anywhere.
- Administrators can create a service group that contains no services for troubleshooting purposes.

Storing Digital Events Using Digital Event Persistence

webMethods Digital Event Persistence is a unified model for storing digital events regardless of the underlying storage technology. With Digital Event Persistence, you can persist event instances sent to and from Digital Event Services to an event store. The event store is not a transactional or analytic database, but a system of record that supports long-term, high-volume storage.

You use webMethods Digital Event Persistence through the Digital Event Persistence service type for Digital Event Services. Digital Event Persistence supports the following storage technologies:

- Elasticsearch 2.3.2
- Apache Hadoop Distributed File System and Hive, Cloudera distribution (HDFS CDH) 5.3.0

You can configure Digital Event Persistence services to persist all instances of a digital event type to a storage destination. Digital Event Persistence also supports *dynamic service configuration* that enables you to persist events to different storage destinations, based on the content of each event.

For more information about how to configure Digital Event Persistence services, see "Configuring Digital Event Persistence Services for Elasticsearch" on page 19 and "Configuring Digital Event Persistence Services for HDFS" on page 16.

2 Administering Digital Event Services

About Administering Digital Event Services

You can administer Digital Event Services, using the Command Central web user interface, composite templates, or command line interface.

Note: Integration Server and Apama developers can also use the Digital Event Services command line tool to configure messaging connectivity without using Command Central. Developers can open a command line prompt and type `java -jar` followed by *Software AG_directory*`\common\lib\events-configuration-tool.jar` to run the utility.

To administer Digital Event Services within your Software AG installation, you need to have installed the Platform Manager plug-ins for:

- Digital Event Services
- Integration Server
- Apama
- Software AG MashZone NextGen

Using Command Central, you can perform the following operations:

- Configure messaging services and service groups.
- Configure event type associations.
- Configure storage settings for digital events.

Important: Do not edit configuration files manually.

For information about using composite templates to configure Digital Event Services, see "Configuring Digital Event Services Using Composite Templates" on page 33. For information about using Command Central CLI commands, see *Software AG Command Central Help*.

Configuring Services

Digital Event Services (DES) provides a default Universal Messaging service, which you can modify or delete, and a pre-defined In-Process service. In addition, you can create, modify, and delete custom Universal Messaging and Digital Event Persistence services, using Command Central.

Configuring Universal Messaging Services

You can create and configure services of Universal Messaging type, add them to groups, and associate event types to them.

To configure a Universal Messaging service

1. In Command Central, navigate to **Environments > Instances > All > *instance_name* > Digital Event Services > Configuration**.
2. Select **Messaging Services** from the drop-down menu, and then click +.
3. Specify values for the following fields:

Field	Description
Service Name	Required. A unique name for the service. The service name is not case-sensitive, and must start with a character. You can use the following characters as separators: . (dot) and - (dash).
Description	Required. A description of the purpose of the service.
Provider URL	Required. Supports Secure Sockets Layer (SSL). The host and port of the Universal Messaging server to which the service connects. The default value is `nsp://localhost:9000`. You can use an SSL-enabled Universal Messaging server, for example: `nsps://localhost:9000`.

Important: You cannot rename existing services. If you want to modify a service name, you must delete the existing service and create a new one with a different name.

Before deleting a service, you must first remove it from any service groups to which the service belongs.

4. Optionally, click **Test** to verify that your configuration is valid.

Note: The validation is done on a field level, and checks whether the specified configuration information complies with the field requirements. This action will not ping a server to verify that a connection is possible. It only checks if the port number is available.

5. Save your changes.

Using Universal Messaging Services with SSL

Digital Event Services (DES) supports the Universal Messaging NSPS protocol for secure communication.

To enable DES to communicate with an SSL-enabled Universal Messaging server, you must configure the following Java system properties for the runtime in which DES is running:

Property	Value
javax.net.ssl.keyStore	The path to the client keystore.
javax.net.ssl.keyStorePassword	The passowrd for the client keystore.
javax.net.ssl.trustStore	The path to the certificate authority (CA) keystore file.
javax.net.ssl.trustStorePassword	The password for the CA keystore.

Note: DES supports only .jks file based keystore and truststore types.

For more information about configuring Universal Messaging for SSL communication, see Universal Messaging documentation. For more information about configuring Java system properties, see *Software AG Command Central Help* or the documentation of the embedding runtime.

Configuring Digital Event Persistence Services for HDFS

With Digital Event Persistence services, you can store events to an Apache Hadoop Distributed File System and Hive, Cloudera distribution (HDFS CDH) 5.3.0 storage engine.

To use HDFS as the storage engine for Digital Event Persistence, you must first configure the Hadoop cluster by deploying the custom Hive SerDe and Joda Date/Time libraries from your Digital Event Persistence installation. For more information about how to configure HDFS for use with Digital Event Persistence services, see "Configuring HDFS for Digital Event Persistence" on page 64.

You can either specify static values in the configuration fields, or use dynamic service configuration to persist events to different storage destinations based on the content of the events. You can specify dynamic values in the **Name Node URI**, **Database**, **Hive Server URI**, and **User Id** fields. To specify a variable, start and end your expression with `$`.

For more information about adding dynamic service configuration to a digital event type, see "Adding Dynamic Service Information to Digital Event Types" on page 64.

To create Digital Event Services services of type Digital Event Persistence for HDFS

1. In Command Central, navigate to **Environments > Instances > All > *instance_name* > Digital Event Services > Configuration**.

2. Select **Event Persistence** from the drop-down menu.
3. Click +, and then select **HDFS CDH 5.3.0** for the service type.
4. Specify values for the following fields:

Parameter	Description
Service Name	Required. The name of the new service. Specify a unique service name that starts with a character. Valid separator characters are periods (.) and dashes (-). The service name is not case-sensitive. **Note:** You cannot rename an existing service. If you want to modify the service name, you must delete the existing service and create a new one with a different name.
Service Description	Optional. A description of the new service.
Name Node URI	Required. Supports dynamic service configuration. The URI of the Name Node in the HDFS cluster . Specify the Name Node URI as follows: `hdfs://host:port`, where *host* is the host name of the server, and *port* is the port on which the server listens for incoming requests. The default value is `hdfs://localhost:8020`. You can use dynamic service configuration to specify the host, for example: `hdfs://$host$:port`.
Maximum File Size(MB)	Required. The HDFS block size in megabytes. The default value is `65`.
Hive Server URI	Required. Supports dynamic service configuration. The URI of the Apache Hive Server. Specify the server URI as follows: `jdbc:hive2://host:port`, where *host* is the host name of the server, and *port* is the port on which the server listens for incoming connection requests. The default value is `jdbc:hive2://localhost:10000`. You can use dynamic service configuration to specify the Hive Server, for example: `jdbc:hive2://$host $:port`.

Parameter	Description
Database	Required. Supports dynamic service configuration. The name of the Hive database. You can use dynamic service configuration to specify all or part of the database name, for example: `$database_name$`.
Warehouse Location	Required. The location of the Hive warehouse. The default value is `/user/hive/warehouse`.
User Id	Required. Supports dynamic service configuration. The username for the Hive user account. You can use dynamic service configuration to specify the user ID, for example: `$userid$`.
Password	Required. The password for the Hive user account.
Batch Size	Required. The number of events that is written to HDFS with a single write operation. The default value is `10000`. **Note:** If the HDFS service queues a batch of events before the batch write timer expires, the service immediately persists all queued events to HDFS.
Batch Write Timer(sec)	Required. Batch write frequency in seconds. The default value is `15`. **Note:** If the batch write timer expires before the HDFS service queues a batch of events, all currently queued events are persisted to HDFS.

5. Optionally, click **Test** to verify that your configuration is valid.

Note: When using dynamic service configuration, it is not possible to successfully connect to HDFS using the **Test** button. However, field validation works as expected.

6. Save your changes.

Configuring Digital Event Persistence Services for Elasticsearch

With Digital Event Persistence services, you can store events to an Elasticsearch 2.3.2 storage engine.

Note: Digital Event Persistence does not support Elasticsearch clusters with Shield enabled.

When using Elasticsearch, you can persist events using Secure Sockets Layer (SSL). For more information about persisting events over SSL to an Elasticsearch instance, see "Configuring SSL for Digital Event Persistence" on page 65.

You can either specify static values in the configuration fields, or use dynamic service configuration to persist events to different storage destinations based on event data. You can specify dynamic service configuration variables in the **Cluster URI(s)**, **Cluster Name**, and **Index Name** fields.

To specify a variable, start and end your variable name with `$`. For example, to persist events in an Elasticsearch index depending on a customer's name, specify `$customer_name$` in the **Index Name** field . As a result, Digital Event Persistence stores events that are related to a customer with the name `john_smith` in an index with the same name.

For more information about adding dynamic service configuration to a digital event type, see "Adding Dynamic Service Information to Digital Event Types" on page 64.

To create Digital Event Services (DES) services of type Digital Event Persistence for Elasticsearch

1. In Command Central, navigate to **Environments > Instances > All > *instance_name* > Digital Event Services > Configuration**.
2. Select **Event Persistence** from the drop-down menu.
3. Click +, and then select **Elasticsearch 2.3.2** for the service type.
4. Specify values for the following fields:

Parameter	Description
Service Name	Required. The name of the new service. Specify a unique service name that starts with a character. Valid separator characters are periods (.) and dashes (-). The service name is not case-sensitive. **Note:** You cannot rename an existing service. If you want to modify the service name, you must delete the existing service and create a new one with a different name.

Parameter	Description
Service Description	Optional. A description of the new service.
Cluster URI(s)	Required. Supports dynamic service configuration. A comma-separated list of servers in an Elasticsearch cluster to which the Digital Event Persistence service can connect. If the initial host is unavailable, Digital Event Persistence attempts to connect to the other servers in the cluster. Specify the URI of the servers in the Elasticsearch cluster as follows: `elasticsearch://host:port`, where *host* is the host name of the server, and *port* is the port on which the server listens for incoming requests. The default value is `elasticsearch://localhost:9300`. You can use dynamic service configuration to specify all or part of the name of the cluster, for example: `elasticsearch://$host$:port`.
Cluster Name	Required. Supports dynamic service configuration. The name of the Elasticsearch cluster. You can use dynamic service configuration to specify all or part of the name of a cluster, for example: `$cluster_name$`.
Index Name	Required. Supports dynamic service configuration. The name of the index in which the events are stored. You can use dynamic service configuration to specify all or part of the name of the index, for example: `$index_name$`. **Note:** Elasticsearch supports only lowercase index names.
Core threads	Required. The number of threads to keep in the pool at all times. The default value is `5`.
Maximum Threads	Required. The maximum number of threads on which to run bulk indexing tasks. The default value is `10`.
Queue Size	Required. The size of the thread pools work queue. A value of 0 indicates an unlimited queue. A value other than 0 sets the number of pending bulk indexing tasks

Parameter	Description
	that can be queued before the processing thread blocks. The default value is `10`.
Keep Alive(sec)	Required. The time limit in seconds for which threads remain idle before being terminated. The default value is `5`.
Pre-Start Core Threads	Required. Whether the core threads start immediately rather than as work is pulled from the queue. Values are `true` or `false` (default).
Batch Size	Required. The number of events that are written to Elasticsearch with a single write operation. The default value is `1000`. **Note:** If the Elasticsearch service queues a batch of events before the batch write timer expires, the service immediately persists all queued events to Elasticsearch.
Batch Write Timer(sec)	Required. Batch write frequency in seconds. The default value is `15`. **Note:** If the batch write timer expires before the Elasticsearch service queues a batch of events, all currently queued events are persisted to Elasticsearch.
Enable SSL	Required. Enable or disable SSL for both REST and TCP ports. The default value is `false`. For more information about how to configure SSL for Digital Event Persistence, see "Configuring SSL for Digital Event Persistence" on page 65.
Key Store	Optional. Specify the absolute file path to the Java KeyStore (JKS) file as follows: *`folder/sub_folder/filename.`*
Trust Store	Optional. Specify the absolute file path to the JKS truststore file as follows: *`folder/sub_folder/filename`*.

5. Optionally, click **Test** to verify that your configuration is valid.

Note: When using dynamic service configuration, it is not possible to successfully connect to Elasticsearch using the **Test** button. However, field validation works as expected.

6. Save your changes.

Using the Default Event Data Store Service

Digital Event Persistence provides a pre-configured Event Data Store (Elasticsearch 2.3.2) service, named local-EventDataStore, that enables you to persist digital events to the Event Data Store. You can use the default Event Data Store service to easily persist events in less complex scenarios.

Note: You must install webMethods Event Data Store if you want to use the default Event Data Store service.

To persist events using the local-EventDataStore service, you add the service to the default Digital Event Services (DES) service group.

When you add the local-EventDataStore service to the default DES service group, all events that are of digital event types associated with the default service group are persisted to the Event Data Store.

For more information about configuring the DES default service group, see "Configuring the Default Service Group" on page 23. For more information about configuring Digital Event Persistence to persist events to Elasticsearch, see "Configuring Digital Event Persistence Services for Elasticsearch" on page 19. For more information about configuring the Event Data Store, see "Using webMethods Event Data Store" on page 67.

Using the In-Process Service

The In-Process (IP) service is a pre-defined service that enables different parts of an application runtime to exchange digital events without using an external messaging server. Only one In-Process service exists per runtime.

You cannot modify the configuration of the pre-defined In-Process service or create a custom In-Process service. You can only add the In-Process service to a service group. In the service group, you can set the **Usage** property for the In-Process service only to **Source and Destination**.

For more information about how to configure a custom service group, see "Configuring Custom Service Groups" on page 24.

Configuring Service Groups

A service group is a defined set of services, where each service has a specific usage. Digital Event Services allows administrators to associate event types with a custom service group during the creation of the group. You use Command Central to configure service groups.

Configuring the Default Service Group

Each product runtime where Digital Event Services is used has a default service group. All event types that are not explicitly associated with a custom service group are associated with the default service group. All events of these event types are delivered to the services within the default service group.

Note: You cannot rename or delete the default service group.

You can modify the default behavior of DES by adding services to or removing services from the default service group.

Note: You must create the services before adding them to the default service group. For more information about creating services, see "Configuring Services" on page 14.

To configure the default service group

1. In Command Central, navigate to **Environments > Instances > All > *instance_name* > Digital Event Services > Configuration**.
2. Select **Service Groups** from the drop-down menu, and then click **Default** in the **Service Group Name** column.
3. Click  to add services to the group.
4. Select the services from the drop-down menu in the **Service Name** field.
5. For each service that you add to the default service group, define the **Usage** property:

Usage	Description
Source Only	Specify this option if your application subscribes to digital events. **Note:** Your service group must contain at most one source service. You cannot include the same service twice in the same service group.

Usage	Description
Destination Only	Specify this option if your application emits events. This is the default value.
Source and Destination	Specify this option if your application both emits and subscribes to events.

6. Optionally, click **Test** to verify that your configuration is valid.

> **Note:** The validation is done on a field level, and checks whether the specified information complies with the field requirements.

7. Save your changes.

Configuring Custom Service Groups

By default, Digital Event Services provides a default service group that contains a pre-configured Universal Messaging service. You can create one or more custom service groups and associate a set of event types with them.

When events of those particular event types are sent or received, they go to all services within the service group. One of the services in the group can be defined as source and/or destination of events for all event types associated with the service group.

To configure a custom service group

1. In Command Central, navigate to **Environments > Instances > All > *instance_name* > Digital Event Services > Configuration**.
2. Select **Service Groups** from the drop-down menu.
3. Click + to add a new custom group.
4. Specify values for the following fields:

Field	Description
Group Name	Required. Specify a unique name for the service group. The name is not case-sensitive, and must start with a character. You can use the following characters as separators: .(dot) and - (dash). **Note:** You cannot rename a service group. If you want to modify a service group name, you must delete the existing service group and recreate the group with a different name.

Field	Description
Description	Required. Specify a description of the purpose of the service group.

Services	Optional. Click + to add services to the group and then do the following:
	■ In the **Service Name** column, select a service.
	■ In the **Usage** column, select on of the following options:
	  ■ **Source Only** - if your application subscribes to digital events.
	  ■ **Destination Only** - if your application emits events.
	  ■ **Source and Destination** - if your application both emits and subscribes to events. This is the default value.
	Note: Your service group must contain at most one source service. In addition, you cannot include the same service twice in the same service group.
	If you add the pre-defined In-Process service to a service group, you can set its usage only to **Source and Destination**. Any other services to the service group, you can only set them as **Destination Only**.

Event Types	Optional. Use the dual list box to manage the digital event types assigned to a service group. The box on the left shows event types that are not associated with any service group. The box on the right shows the event types currently assigned to this service group. Use the arrows to move event types between the two boxes. **Tip:** Use the search box to filter the list of available digital event types. You can also select multiple event types by holding down the CTRL button and clicking specific entries, or by holding down SHIFT and clicking the first and last entry of a range. You can also manage the association of event types with service groups on the Digital Event Services > Configuration > Event Types page in Command Central. For more information about event type management, see "Configuring Event Type Associations" on page 26.

Field	Description
	Important: An event type can only be associated with a single service group. If you want to change the service group with which an event type is associated, you must first dis-associate the event type from its original service group, so that the event type shows in the dual list box.

Note: You cannot delete a custom service group that has event types associated with it.

5. Optionally, click **Test** to verify that your configuration is consistent.
6. Save your changes.

Configuring Event Type Associations

When events of a particular event type are sent or received, Digital Event Services delivers them to the services within a service group. To configure to where events are sent you need to associate event types with their respective service groups.

Applications cannot subscribe for digital event types that are not present in the DES event type repository. For more information about adding event types to the event type repository, see " Considerations for Deploying Digital Event Types" on page 46.

To configure digital event type associations

1. In Command Central, navigate to **Environments > Instances > All > *instance_name* > Digital Event Services > Configuration**.
2. Select **Event Types** from the drop-down menu.
3. In the **Name** column, click **Event Types**, and then click **Edit**.
4. For each digital event type in the **Event Types** table, specify the following information:

Field	Description
Delivery Mode	The quality of service per event type, if the runtime in which DES is embedded stops responding and becomes unavailable. Values are: ■ **Follow Preference** - events are delivered according to the delivery mode preference set in the event type definition. This is the default value. ■ **Persistent** - events are stored on-disk and resent after the runtime becomes available again.

Field	Description
	■ **Non-persistent** - events are stored in-memory and are not resent after the runtime becomes available. **Important:** Changing the delivery mode preference for an event type in the digital event type definition requires a restart of the runtime in which DES is embedded for the changes to take effect. To avoid restarting the runtime, instead of **Follow Preference**, you can select either **Persistent** or **Non-Persistent** in the **Delivery Mode** field to reflect the change in the event type. For more information about the quality of service setting, see "Delivery Modes for Digital Event Services" on page 9.
In-Memory Capacity	The maximum number of events that are kept in-memory for an event type. You can specify any positive integer, or you can specify 1K (1024), 1M (1024K), or 1G (1024M). Alternatively, you can leave an empty string or specify `Default` to use the global default setting of the **In-Memory Capacity** property. For more information about configuring the global default value for **In-Memory Capacity**, see "Configuring Storage Settings for Digital Event Services" on page 28.
On-Disk Capacity	The maximum number of events that are kept on-disk for an event type. You can specify any positive integer, or you can specify 1K (1024), 1M (1024K), or 1G (1024M). Alternatively, you can leave an empty string or specify `Default` to use the global default setting of the **On-Disk Capacity** property. For more information about configuring the global default value for **On-Disk Capacity**, see "Configuring Storage Settings for Digital Event Services" on page 28.
Service Group	Select a service group from the drop-down menu with which to associate the event type.

Field	Description
	Note: Digital event types that are not explicitly associated with a custom service group are associated with the default service group. You can also associate event types with service groups during the configuration of a service group. For more information, see "Configuring Custom Service Groups" on page 24.

5. Optionally, click **Test** to verify that your configuration is consistent.
6. Save your changes.

Configuring Storage Settings for Digital Event Services

Depending on the delivery mode configured for an event type, all events sent to Digital Event Services are stored on-disk or in-memory. For more information about how the delivery preference you set affects DES, see "Delivery Modes for Digital Event Services" on page 9.

To configure the storage settings for Digital Event Services

1. In Command Central, navigate to **Environments > Instances > All > *instance_name* > Digital Event Services > Configuration**.
2. Select **Runtime Configuration** from the drop-down menu.
3. In the **Name** column, select **Runtime Configuration**, and then click **Edit**.
4. Specify values for the following fields:

Field	Description
Storage Location	Optional. Defines the location where events are stored on-disk. Specify an existing folder in your file system. **Important:** Before you specify a new storage location, if the existing storage location is in use, you must copy any existing files manually to the new storage location. If you do not specify a value, events are stored in the same directory as the configuration.

Field	Description
Storage Batch Size	Required. Defines the number of events that must be added to the on-disk store-and-forward queue before the queue is persisted to disk. The default value is `100` events.
Sync Timeout	Required. Defines the timeout in milliseconds that the on-disk store-and-forward queue waits before the events in the queue are persisted, in case the batch size has not been reached. The default value is `500` milliseconds.
Default On-Disk Capacity	Required. Defines the maximum number of events of each event type that can be stored on disk. The default value is `1000000`. **Note:** You can override the setting by specifying **On-Disk Capacity** per event type. For more information about configuring the **On-Disk Capacity** property, see "Configuring Event Type Associations" on page 26.
Default In-Memory Capacity	Required. Defines the maximum number of events of each event type that can be stored in the memory. The default value is `1000`. **Note:** You can override the setting by specifying **In-Memory Capacity** per event type. For more information about configuring the **In-Memory Capacity** property, see "Configuring Event Type Associations" on page 26.

5. Save your changes.

Digital Event Services detects that the configuration has been updated, and starts to use the new settings for **Default On-Disk Capacity** and **Default In-Memory Capacity** automatically.

Important: If you modify the **Storage Batch Size**, **Sync Timeout**, and **Storage Location** properties, you must restart the runtime where the DES component is embedded for the changes to take effect.

For 10.1, this means restarting the Integration Server or Apama runtime. For more information about shutting down and starting Integration Server and Apama, see *webMethods Integration Server Administrator's Guide* and *Deploying and Managing Apama Applications*, respectively.

Monitoring Digital Event Services

You can monitor the status of Digital Event Services (DES) in real time by using:

- **Optimize for Infrastructure**. Optimize for Infrastructure provides you with a set of default rules for notification of object failure and also enables you to define your own custom rules. For more information about monitoring DES by using Optimize for Infrastructure, see *Administering webMethods Optimize*.
- **The DES Monitoring API**. The monitoring API is based on Java Management Extensions (JMX) and represents services and queues by using managed beans (MBeans). For more information about the MBeans that DES exposes, see" Digital Event Services MBeans" on page 49.

To enable monitoring by MBeans, set the com.softwareag.events.routing.enable.jmx system property to `true`. For more information about setting the property, see the product documentation of the DES runtime.

In addition, you can use the JMX Remote API to monitor remotely a server runtime where DES is running. To enable monitoring from remote systems, when you start the server runtime set com.sun.management.jmxremote.port system property. For more information about setting the property, see the product documentation of the DES runtime.

Important: When you enable remote monitoring of DES, ensure that the exposed information can only be accessed after authorization. For more information about securing the DES runtime, see the product documentation of the server runtime and the Java documentation.

Secure Password Handling

You can use Digital Event Services (DES) services to connect to password-protected systems. For example, you might configure DES to archive events in a database by using Digital Event Persistence services.

DES service configuration files need to contain a password to successfully establish a secure connection to systems that require password authentication. To protect these passwords, DES encrypts and decrypts passwords using a master password that is also encrypted. In a Software AG installation, all runtimes where DES is embedded use a single master password. The master password is stored in *Software AG_directory*/common/DigitalEventServices/security/passman.

Important: When you first install DES, the default master password is "manage". Software AG recommends that you change the default master password shortly after installing DES. For more information about specifying a new master password, see "Changing the Master Password" on page 31.

Changing the Master Password

Digital Event Services (DES) provides you with a command line tool for changing the master password. The command line tool is installed in *Software AG_directory*/common/lib/events-passman-tool.jar.

Note: You can find usage information for the command line utility by opening a command line prompt in *Software AG_directory*/common/lib and by executing the following command:

```
java -jar events-passman-tool.jar
```

To change the master password:

1. Open command line prompt in the *Software AG_directory* /common/lib.
2. Specify a new master password by typing the following command:

   ```
   java -jar events-passman-tool.jar change-mpw old_mpw new_mpw
   passman_config_location
   ```

   where:

   - *old_mpw* is the old master password that is needed for authentication.
   - *new_mpw* is the new master password.
   - *passman_config_location* is the storage location of the master password: *Software AG_directory*/common/DigitalEventServices/security/passman.

   Note: If the passwords and/or the configuration location contain characters that affect the parameter resolution of the command shell, you need to apply appropriate escaping.

The command line tool changes the master password of the given configuration to the value you specify for *new_mpw* , and reports the number of updated passwords in service definitions.

Lifecycle Actions for Digital Event Services

Digital Event Services does not support any lifecycle actions, such as start, stop, or restart. Although the **Lifecycle Actions** setting for DES is active in the Command Central web user interface, administrators should not use this functionality.

For more information about shutting down and starting Integration Server and Apama, see *webMethods Integration Server Administrator's Guide* and *Deploying and Managing Apama Applications*, respectively.

Digital Event Services Licensing

If you do not have a valid license for Digital Event Services, you cannot create events, and your applications cannot emit and subscribe to digital events.

By default, Digital Event Services comes with a 30-day trial license. The original DES license can be overwritten by another license at any time. The DES license is located in the following file: *Software AG_directory*/common/DigitalEventServices/license/license.xml.

Important: A missing license file is regarded as an invalid license.

For more information about licensing, see *Installing Software AG Products*.

Performance Considerations on Linux

For deployments on a Linux operating system where performance is a concern, use an XFS file system for the Digital Event Services store-and-forward queue. If the throughput is high, XFS file systems perform better compared to the default ext4 file systems.

3 Configuring Digital Event Services Using Composite Templates

About Configuring Digital Event Services with Command Central Composite Templates

You can use a composite template and the Command Central Command Line Interface to configure Digital Event Services for one or more run-time components in new or existing installations.

For more information about composite templates and scripting, see *Software AG Command Central Help*.

The following sections contain examples of how composite templates can be used with Digital Event Services.

Configuring DES for a Single Runtime Instance

The template snippet in this example creates and configures a Digital Event Services messaging service of type Universal Messaging for the default Integration Server instance.

The template defines a single layer, with alias `is1`. The `is1` layer includes the `is1-des-messaging-configuration` inline template, which is defined in the templates section. The `is1` layer maps to the local node in the provision section.

In the configuration section of the Integration Server instance template, the run-time component `OSGI-IS_default-DigitalEventServices` refers to `DES-MESSAGING-CONFIG`, which defines the parameters for the `COMMON-WMMESSAGING-DES` configuration type.

`OSGI-IS_default-DigitalEventServices` is the run-time component ID, which you can obtain by using the `sagcc list inventory components` CLI command.

`COMMON-WMMESSAGING-DES` is the configuration type ID, and `MyUM` is the configuration instance ID. The value of the `"@alias"` parameter should be the same as the configuration instance ID, in this case: `"@alias": MyUM`.

Command Central applies `DES-MESSAGING-CONFIG` to the `OSGI-IS_default-DigitalEventServices` run-time component, thus creating and setting up a Universal Messaging service for Digital Event Services.

For more information about working with composite templates and the `sagcc list inventory components` command, see *Software AG Command Central Help*.

```
alias: single-des-config-single-install
description: Configure DES for a single instance
version: 0.1

layers:
  is1:
    templates: [is1-des-messaging-configuration]

DES-MESSAGING-CONFIG: &DES-MESSAGING-CONFIG
```

```
  COMMON-WMMESSAGING-DES:
    MyUM:
      Messaging:
        "@alias": MyUM
        Description: NewUM
        Enabled: true
        Provider:
          "@type": UM
          URL: nsp://localhost:1234

templates:
  is1-des-messaging-configuration:
    products:
      integrationServer:
        default:
          configuration:
            OSGI-IS_default-DigitalEventServices: *DES-MESSAGING-CONFIG

provision:
  default:
    is1: local
```

Configuring DES for Multiple Runtime Instances within a Single Installation

The composite template in this example creates and configures a Digital Event Services messaging service of type Universal Messaging for two Integration Server instances named `IS_default` and `IS_default2`.

The template defines two layers, with aliases `is1` and `is2`. Both layers map to the local node in the provision section.

The inline template applied to the `is1` layer is `is1-des-messaging-configuration`. The template configures the run-time component `OSGI-IS_default-DigitalEventServices` by referring to `DES-MESSAGING-CONFIG`, which defines the parameters for the `COMMON-WMMESSAGING-DES` configuration type.

The inline template applied to the `is2` layer is `is2-des-messaging-configuration`. The template configures the run-time component `OSGI-IS_default2-DigitalEventServices` by referring to `DES-MESSAGING-CONFIG`, which defines the parameters for the `COMMON-WMMESSAGING-DES` configuration type.

`OSGI-IS_default-DigitalEventServices` and `OSGI-IS_default2-DigitalEventServices` are the IDs of the run-time components to be configured. You can obtain the component ID of a runtime by using the `sagcc list inventory components` CLI command.

`COMMON-WMMESSAGING-DES` is the configuration type ID, and `MyUM` is the configuration instance ID. The alias parameter should be the same as the configuration instance ID, in this case: `"@alias": MyUM`.

Command Central applies the `DES-MESSAGING-CONFIG` configuration to the run-time components `OSGI-IS_default-DigitalEventServices` and `OSGI-IS_default2-`

`DigitalEventServices`. Applying `DES-MESSAGING-CONFIG` creates and configures a Universal Messaging service for both components.

For more information about working with composite templates and the `sagcc list inventory components` command, see *Software AG Command Central Help*.

```
alias: multiple-des-config-single-install
description: Configure DES for multiple instances in a single installation
version: 0.1

layers:
  is1:
    templates:
      - is1-des-messaging-configuration

  is2:
    templates:
      - is2-des-messaging-configuration

DES-MESSAGING-CONFIG: &DES-MESSAGING-CONFIG
  COMMON-WMMESSAGING-DES:
    MyUM:
      Messaging:
        "@alias": MyUM
        Description: NewUM
        Enabled: true
        Provider:
          "@type": UM
          URL: nsp://localhost:1234

templates:
  is1-des-messaging-configuration:
    products:
      integrationServer:
        default:
          configuration:
            OSGI-IS_default-DigitalEventServices: *DES-MESSAGING-CONFIG

  is2-des-messaging-configuration:
    products:
      integrationServer:
        default2:
          configuration:
            OSGI-IS_default2-DigitalEventServices: *DES-MESSAGING-CONFIG

provision:
  default:
    is1: local
    is2: local
```

Configuring DES for a Set of Installations

The composite template in this example configures Digital Event Services for the Integration Server instance `IS_Default` in two different installations.

This template defines two layers, with aliases `is1` and `is2`, which in the provision section map to the nodes `local` and `local2`, respectively.

The templates applied to the `is1` and `is2` layers are `is1-des-messaging-configuration` and `is2-des-messaging-configuration`, respectively. The templates configure the run-time component `OSGI-IS_default-DigitalEventServices` by referring to `DES-MESSAGING-CONFIG`, which defines the parameters for the `COMMON-WMMESSAGING-DES` configuration type.

`OSGI-IS_default-DigitalEventServices` is the ID of the run-time component to be configured. You can obtain the component ID of a runtime by using the `sagcc list inventory components` CLI command.

`COMMON-WMMESSAGING-DES` is the configuration type ID, and `MyUM` is the configuration instance ID. The alias parameter should be the same as the configuration instance ID, in this case: `"@alias": MyUM`.

Command Central applies the `DES-MESSAGING-CONFIG` configuration to the run-time component `OSGI-IS_default-DigitalEventServices`, thus creating and configuring Universal Messaging for multiple instances in multiple installations.

For more information about working with composite templates and the `sagcc list inventory components` command, see *Software AG Command Central Help*.

```
alias: multiple-des-config-multi-install
description: Configure DES for multiple instances in multiple installations
version: 0.1

layers:
  is1:
    templates:
      - is1-des-messaging-configuration

  is2:
    templates:
      - is2-des-messaging-configuration

DES-MESSAGING-CONFIG: &DES-MESSAGING-CONFIG
  COMMON-WMMESSAGING-DES:
    MyUM:
      Messaging:
        "@alias": MyUM
        Description: NewUM
        Enabled: true
        Provider:
          "@type": UM
          URL: nsp://localhost:1234

templates:
  is1-des-messaging-configuration:
    products:
      integrationServer:
        default:
          configuration:
            OSGI-IS_default-DigitalEventServices: *DES-MESSAGING-CONFIG

  is2-des-messaging-configuration:
    products:
      integrationServer:
        default:
          configuration:
            OSGI-IS_default-DigitalEventServices: *DES-MESSAGING-CONFIG
```

```
nodes:
  default:
    default:
      secure: true
      credentials:
        username: Administrator
        password: manage
    local:
      host: localhost
      port: 8093
    local2:
      host: localhost
      port: 9815

provision:
  default:
    is1: local
    is2: local2
```

Creating and Configuring a DES Runtime Instance

The composite template in this example sets up a new installation on the node with alias `local2`, with an Integration Server instance called `IS_default2`, and also configures Digital Event Services.

The template defines the layer `new-is`, which is mapped to the node `local2`, as per the details in specified under `provision`. Integration Server will be installed from the repository called `webMethods-10.1` to the directory `C:/SoftwareAG2`.

The template applied to the `new-is` layer is `[create-and-configure-is]` and configures Digital Event Services and the Integration Server instance `IS_default2`. The template configures the run-time component `OSGI-IS_default-DigitalEventServices` by referring to `DES-MESSAGING-CONFIG`, which defines the parameters for the `COMMON-WMMESSAGING-DES` configuration type.

`COMMON-WMMESSAGING-DES` is the configuration type ID, and `MyUM` is the configuration instance ID. The alias parameter should be the same as the configuration instance ID, in this case: `"@alias": MyUM`.

When applying the template, Command Central applies the `new-is` layer on the `local2` node and installs Digital Event Services and Integration Server with the instance defined in the `[create-and-configure-is]` template. When `IS_default2` is up and running, Command Central applies the `DES-MESSAGING-CONFIG` configuration on the `OSGI-IS_default2-DigitalEventServices` run-time component.

For more information about working with composite templates, see *Software AG Command Central Help*.

```
alias: setup-installation-configure-instance
description: Create a new installation with an IS instance and configure DES
version: 0.1

layers:
  new-is:
    productRepo: webMethods-10.1
```

```
    templates: [create-and-configure-is]

DES-MESSAGING-CONFIG: &DES-MESSAGING-CONFIG
  COMMON-WMMESSAGING-DES:
    MyUM:
      Messaging:
        "@alias": MyUM
        Description: NewUM
        Enabled: true
        Provider:
          "@type": UM
          URL: nsp://localhost:1234

templates:
  create-and-configure-is:
    products:
         DEV:
         integrationServer:
           default2:
             primary.port: 5805
             diagnostic.port: 5806
             jmx.port: 5807
             configuration:
               OSGI-IS_default2-DigitalEventServices: *DES-MESSAGING-CONFIG

nodes:
  default:
    local2:
      host: localhost
      port: 9815
      secure: true
      bootstrapInfo:
        installDir: C:/SoftwareAG2

provision:
  default:
    new-is: local2
```

Creating a Digital Event Persistence Service Using Composite Templates

This section includes an example of a composite template that you can use to create a service of type Digital Event Persistence for Digital Event Services (DES), which is embedded in an Integration Server instance.

Before You Begin

Before you can create the composite template for creating a Digital Event Persistence service, do the following:

- Install a default Integration Server with DES and Digital Event Persistence. When using the sample template in this section, install Integration Server on the local host with default ports.
- See the "Creating Custom Composite Templates" section in *Software AG Command Central Help*.

Creating the Template Definition File

The following example configures a Digital Event Persistence service with Apache Hadoop 5.3.0 as the storage technology. To configure a Digital Event Persistence service with Elasticsearch 2.3.2 as the storage technology, use the same template but include parameters specific to Elasticsearch.

The template creates a single layer, with alias `local-is` and applies the `create-an-evp-service` inline template on the layer. The `local-is` layer is created in the local installation.

In the configuration section of the template, the configuration details specified in `DES-EVP-CONFIGURATION` are applied on the `OSGI-IS_default-DigitalEventServices` run-time component. `DES-EVP-CONFIGURATION` contains the parameters for the `EVENT-PERSISTENCE-DES` configuration type, and `LocalPersistence` is the instance ID.

`OSGI-IS_default-DigitalEventServices` is the run-time component ID, which you can obtain by using the `sagcc list inventory components` CLI command.

Important: To connect to an HDFS server you need to authenticate with a password. To successfully provision a Digital Event Persistence service when you apply the template you must provide the value for `hiveServerPassword` in one of the following ways:

- As a value in the composite template itself.
- As an argument.
- In a properties file.

You must also provide a valid password when updating a Digital Event Persistence service with composite templates. For more information about updating a Digital Event Persistence service with composite templates, see "Updating a Digital Event Persistence Service Using Composite Templates" on page 42.

Example

```
alias: evp-configuration
description: Create and configure a EVP service in DES
version: 0.1

layers:
  local-is:
    templates: [create-an-evp-service]

DES-EVP-CONFIGURATION: &DES-EVP-CONFIGURATION
  EVENT-PERSISTENCE-DES:
    LocalPersistence:
      serviceName: LocalPersistence
      serviceType: HDFS CDH.5.3.0
      serviceDescription: MyDescription
      eventTypeStoreLocation: default
      nameNodeURI: hdfs://localhost:8020
      hdfsMaxFileSize: 65
      hiveServerURI: jdbc:hive2://localhost:10000
```

```
      hiveDatabaseName: Database
      hiveWarehouseLocation: /user/hive/warehouse
      hiveServerUser: Admin
      hiveServerPassword: my.secret
      hdfsBatchSize: 10000
      hdfsBatchWriteTimerSec: 15
      elasticSearchBatchSize: 1000
      elasticSearchBatchWriteTimerSec: 15
      elasticSearchCoreThreadPoolSize: 5
      elasticSearchMaximumThreadPoolSize: 10
      elasticSearchThreadPoolKeepAliveTime: 5
      elasticSearchThreadPoolQueueSize: 10
      elasticSearchPreStartCoreThreads: false
      elasticSearchSSLEnabled: false

templates:
  create-an-evp-service:
    products:
      integrationServer:
        default:
          configuration:
            OSGI-IS_default-DigitalEventServices: *DES-EVP-CONFIGURATION

provision:
  default:
    local-is: local
```

Importing the Composite Template

Before you can apply the template, you must import the template into Command Central.

Import the template from the example with the following command:

- If you only have a template definition file:

  ```
  sagcc exec templates composite import -i evp-configuration.yaml
  ```

- If you have a template archive with a template definition and an environment properties file:

  ```
  sagcc exec templates composite import -i evp-configuration.zip
  ```

Applying the Composite Template

Importing the template registers it in Command Central. You can apply the template with the following command:

```
sagcc exec templates composite apply evp-configuration
```

Apply the template from the example with the following command:

- With an input argument for `hiveServerPassword`:

  ```
  sagcc exec templates composite apply evp-configuration my.password=my.secret
  ```

- With an input properties file:

  ```
  sagcc exec templates composite apply -i evp-configuration sample.properties
  ```

  where `sample.properties` is the name of your custom properties file that contains the value for the `hiveServerPassword` parameter.

Verifying the Template Execution

Verify that the template execution is successful in the following ways:

- Check the **Jobs** view in the Command Central web user interface while applying the template.
- After applying the template, check in Command Central if the service is listed with the correct configuration settings in **Environments > Instances > All > local-is > Digital Event Services > Configuration > Event Persistence**.

Updating a Digital Event Persistence Service Using Composite Templates

You can update a Digital Event Persistence service with the same template you use to create the service. For more information about how to create a Digital Event Persistence service with composite templates, see "Creating a Digital Event Persistence Service Using Composite Templates" on page 39.

Before updating a Digital Event Persistence service, see the "Updating a Provisioned Environment Using the Same Composite Template" section in *Software AG Command Central Help*.

Update the Digital Event Persistence Service Details

Make the necessary changes to the configuration section of the same template you use to create Digital Event Persistence services. Use a local copy instead of modifying the template in *Software AG_directory* \profiles\CCE\data\templates\composite.

Import the Updated Template

When you import an updated template you should use the `overwrite={true | false}` argument to specify whether you want to create a new version of the original template or whether you want to replace it. Use the following command to update the composite template from the example:

```
sagcc exec templates composite import -i evp-configuration.yaml overwrite=true
```

Retrieve Encrypted Password

Before applying the template again, you can obtain the encrypted password for the Apache Hadoop 5.3.0 server, and replace the clear text password in the template definition. This step is optional.

To retrieve the password you use in the composite template example, use the following command:

```
cc get configuration data local OSGI-IS_default-DigitalEventServices LocalPersistence -f json
```

When you are using a custom properties file to store the value of the `hiveServerPassword` parameter, you must replace the original value with the updated encrypted password.

Applying the Updated Composite Template

You can apply the updated composite template with the following command:

```
sagcc exec templates composite apply evp-configuration
```

If you have not specified a value for `hiveServerPassword` in the template definition run the command as follows:

- With an input argument:

  ```
  sagcc exec templates composite apply evp-configuration my.password=my.secret
  ```

  where *`my.secret`* is the password of the Apache Hadoop Server user.

- With an input properties file:

  ```
  sagcc exec templates composite apply -i evp-configuration sample.properties
  ```

  where *`sample`*`.properties` is the name of the custom properties file that contains the value for `hiveServerPassword`.

Verifying the Template Execution

Verify that the template execution is successful in the following ways:

- Check the **Jobs** view in the Command Central web user interface while applying the template.
- After applying the template, check in Command Central if the service is updated with the correct configuration settings in **Environments > Instances > All > local-is > Digital Event Services > Configuration > Event Persistence**.

4 Deployment of Digital Event Types

Deployment of Digital Event Types

Deployment, in the context of Digital Event Services (DES), is the process of provisioning a run-time environment with digital event types.

You use repository-based deployment in webMethods Deployer to deploy digital event type composites to one or more target runtimes. You must install the Asset Build Environment (ABE) to use repository-based deployment.

The digital event type composites that you create prior to deployment must have a specific structure in order to be deployable using Deployer.

You create digital event types from the native document type definitions of Apama and Integration Server by synchronizing them to a local event type repository for the installation. The local event type repository is the source for the digital event type composites. When you run the ABE build script, the script searches the specified event type repositories and creates a composite for all digital event types in it.

Digital event type definitions are individual assets that the Asset Build Environment packs into zip archives. A single zip file can contain multiple event type definitions.

For more information about installing the Asset Build Environment, see *Installing Software AG Products*. For more information about building composites for repository-based deployment, see *webMethods Deployer User's Guide*.

Considerations for Deploying Digital Event Types

Consider the following information when deploying digital event types from your local environment to other instances of Integration Server or Apama:

- Applications cannot subscribe to digital event types that are not present in the DES event type repository.
- Do not modify the internal structure of the event type repository or the names of digital event types.

5 Design-Time Considerations

Design-Time Considerations for Digital Event Services

In Software AG Designer, you develop Integration Server and Apama applications that communicate with one another by exchanging digital events. Each digital event type has an associated digital event type definition that is created and stored in the digital event type repository for the installation. You can use both Integration Server and Apama tooling to generate a digital event type from their native type definitions.

Important: The ownership of a digital event type is non-transferable and belongs to a single application.

Consider the following information when developing applications that use Digital Event Services (DES) to communicate:

- Do not create the same digital event type definition from both an Apama event type and an Integration Server document type. This constraint applies to both digital event types and the digital event types that they require implicitly in their event type definitions.
- Start by generating a digital event type from an Integration Server document type when an Integration Server application processes digital events. Integration Server does not support the creation of Integration Server document types from digital event types.
- Do not modify Apama event types that you generate from a digital event type. Instead, modify the source Integration Server document type and re-generate the digital event type from it. Use Apama tooling to re-generate the Apama event type definition from the digital event type.

A Digital Event Services MBeans

Event Type Information MBean

Retrieves information about an event type.

MBean Name

```
com.softwareag.events.routing:type=EventTypes,eventType=event_type_name,
category=Information
```

where *event_type_name* is the name of the event type.

MBean Attributes

Attribute	Description
deliveryMode	String. The delivery mode of the event type. Values are: ■ `Persistent` ■ `Non-Persistent` For more information about delivery modes, see "Delivery Modes for Digital Event Services" on page 9.

Universal Messaging Service MBean

Retrieves information about the Universal Messaging (UM) services configured in a service group that is associated with an event type.

MBean Name

```
com.softwareag.events.routing:type=EventTypes,eventType=event_type_name,
category=Services,serviceName=service_name
```

where *event_type_name* is the name of the event type and *service_name* is the name of the Universal Messaging service.

MBean Attributes

Attribute	Description
name	String. The name of the service.

Attribute	Description
usage	String. The usage of the service. Values are: - `SourceOnly` - you can subscribe to the service, but you cannot use the service to send events. - `DestinationOnly` - you can use the service to only send events. - `SourceAndDestination` - you can both subscribe to the service and use the service to send events. For information about the possible usage of a service, see "Configuring Service Groups" on page 23.
providerUrl	String. The URL of the Universal Messaging server. The providerUrl attribute has the following format: *protocol* ://*host* : *port* . The default value is `nsp://localhost:9000`.
connected	Boolean. Indicates whether the service is connected to the configured Universal Messaging server.
connectTime	Instant. The time when the service connected to the Universal Messaging server, or reconnected to the server after losing connection.
disconnectTime	Instant. The time when the service disconnected from the Universal Messaging server.
activeExclusiveDurable-SubscribersCount	Integer. The number of active exclusive durable subscribers to the service. If the service usage is `DestinationOnly`, the value of this attribute is empty.
activeExclusiveDurable-SubscribersIds	List<String>. The identifiers of the active exclusive durable subscribers to the service. If the service usage is `DestinationOnly`, the value of this attribute is empty.
activeSharedDurable-SubscribersCount	Integer. The number of active shared durable subscribers to the service. If the service usage is `DestinationOnly`, the value of this attribute is empty.

Attribute	Description
activeSharedDurable-SubscribersIds	List<String>. The identifiers of the active shared durable subscribers to the service. If the service usage is `DestinationOnly`, the value of this attribute is empty.
activePriorityDurable-SubscribersCount	Integer. The number of active priority durable subscribers to the service. If the service usage is `DestinationOnly`, the value of this attribute is empty.
activePriorityDurable-SubscribersIds	List<String>. The identifiers of the active priority durable subscribers to the service. If the service usage is `DestinationOnly`, the value of this attribute is empty.
inactiveExclusiveDurable-SubscribersCount	Integer. The number of exclusive durable subscribers to the service that are inactive but still subscribed. If the service usage is `DestinationOnly`, the value of this attribute is empty. **Important:** The system keeps inactive subscribers only while DES is running. If an exclusive durable subscriber is closed, but remains subscribed, the subscriber ID is added to the inactiveExclusiveDurableSubscribersIds list, and inactiveExclusiveDurableSubscribersCount increases. However, when you restart the runtime where DES is embedded, DES does not examine the Universal Messaging channel for available named objects. As a result, the inactiveExclusiveDurableSubscribersIds list is initially empty and does not contain the subscriber ID that was previously added.
inactiveExclusiveDurable-SubscribersIds	List<String>. The identifiers of the exclusive durable subscribers to the service that are inactive but still subscribed. If the service usage is `DestinationOnly`, the value of this attribute is empty. **Important:** The system keeps inactive subscribers only while DES is running. If an exclusive durable subscriber is closed, but remains subscribed, the subscriber ID is added to the inactiveExclusiveDurableSubscribersIds list, and inactiveExclusiveDurableSubscribersCount

Attribute	Description
	increases. However, when you restart the runtime where DES is embedded, DES does not examine the Universal Messaging channel for available named objects. As a result, the inactiveExclusiveDurableSubscribersIds list is initially empty and does not contain the subscriber ID that was previously added.
inactiveSharedDurable-SubscribersCount	Integer. The number of shared durable subscribers to this service that are inactive but still subscribed. If the service usage is `DestinationOnly`, the value of this attribute is empty. **Important:** The system keeps inactive subscribers only while DES is running. If a shared durable subscriber is closed, but remains subscribed, the subscriber ID is added to the inactiveSharedDurableSubscribersIds list, and inactiveSharedDurableSubscribersCount increases. However, when you restart the runtime where DES is embedded, DES does not examine the Universal Messaging channel for available shared named objects. As a result, the inactiveSharedDurableSubscribersIds list is initially empty and does not contain the subscriber ID that was previously added.
inactiveSharedDurable-SubscribersIds	List<String>. The identifiers of the shared durable subscribers to the service that are inactive, but still subscribed. If the service usage is `DestinationOnly`, the value of this attribute is empty. **Important:** The system keeps inactive subscribers only while DES is running. If a shared durable subscriber is closed, but remains subscribed, the subscriber ID is added to the inactiveSharedDurableSubscribersIds list, and inactiveSharedDurableSubscribersCount increases. However, when you restart the runtime where DES is embedded, DES does not examine the Universal Messaging channel for available shared named objects. As a result, the inactiveSharedDurableSubscribersIds list is initially empty and does not contain the subscriber ID that was previously added.

Attribute	Description
inactivePriorityDurable-SubscribersCount	Integer. The number of priority durable subscribers to the service that are inactive but still subscribed. If the service usage is `DestinationOnly`, the value of this attribute is empty. **Important:** The system keeps inactive subscribers only while DES is running. If a priority durable subscriber is closed, but remains subscribed, the subscriber ID is added to the inactivePriorityDurableSubscribersIds list, and inactivePriorityDurableSubscribersCount increases. However, when you restart the runtime where DES is embedded, DES does not examine the Universal Messaging channel for available priority named objects. As a result, the inactivePriorityDurableSubscribersIds list is initially empty and does not contain the subscriber ID that was previously added.
inactivePriorityDurable-SubscribersIds	List<String>. The identifiers of the priority durable subscribers to the service that are inactive but still subscribed. If the service usage is `DestinationOnly`, the value of this attribute is empty. **Important:** The system keeps inactive subscribers only while DES is running. If a priority durable subscriber is closed, but remains subscribed, the subscriber ID is added to the inactivePriorityDurableSubscribersIds list, and inactivePriorityDurableSubscribersCount increases. However, when you restart the runtime where DES is embedded, DES does not examine the Universal Messaging channel for available priority named objects. As a result, the inactivePriorityDurableSubscribersIds list is initially empty and does not contain the subscriber ID that was previously added.
nonDurable-SubscribersCount	Integer. The number of non-durable subscribers to the service. If the service usage is `DestinationOnly`, the value of this attribute is empty.
receivedEvents	Long. The number of events received by all subscribers to the service since the service became

Attribute	Description
	active. If the service usage is `DestinationOnly`, the value of this attribute is empty.

lastSendingStatus	String. The status of the last sending operation performed by the service. Values are:
	■ `Green` - when the last sending operation to the Universal Messaging server is successful.
	■ `Red` - when the last sending operation to the Universal Messaging server failed. Possible reasons for a `Red` sending status are:
	  ■ The Universal Messaging server is unavailable.
	  ■ The Universal Messaging server is overloaded and responds with a delay.
	  ■ The events that DES sends exceed the capacity of the Universal Messaging channel in use.
	  ■ There is no network connectivity or it is slow.
	Important: If the Universal Messaging server becomes unavailable, but no events are sent for long time, the lastSendingStatus might still be `Green` because the status is updated only after a transaction to the server.
	To determine whether the server is available without sending events, check the connected, connectTime, and disconnectTime attributes.

lastSendingStatusDetails	String. Information about the last sending status. When the status is `Red`, the explanation contains the exception that caused the sending operation to fail.
sentEvents	Long. The number of events sent to the service. If the service usage is `SourceOnly`, the value of this attribute is empty.
acknowledgedSentEvents	Long. The number of events sent to the service and successfully acknowledged by the Universal Messaging server. If the service usage is `SourceOnly`, the value of this attribute is empty.
lastSendingTime	Instant. The time when the last event was sent to the Universal Messaging server.

Attribute	Description
	Note: The lastSendingTime attribute reflects the time when an event was last sent to the destination service. The lastSendingTime attribute does not carry information about whether an event was acknowledged by the destination service. To see whether an event was acknowledged by the destination service, check the lastCompletedAcknowledgementTime attribute.
lastCompletedAcknowledgementTime	Instant. The last time when an event sent to the service was successfully acknowledged by the Universal Messaging server. If the service usage is `SourceOnly`, the value of this attribute is empty.
regularDeliveryInterruptionsCount	Integer. The number of times when the service switched to redelivery mode because of a failure to deliver an event to the Universal Messaging server. If the service usage is `SourceOnly`, the value of this attribute is empty.

In-Process Service MBean

Retrieves information about the In-Process (IP) service configured in a service group that is associated with an event type.

MBean Name

```
com.softwareag.events.routing:type=EventTypes,eventType=event_type_name,
category=Services,serviceName=In-Process
```

where *event_type_name* is the name of the event type.

MBean Attributes

Attribute	Description
name	String. The name of the service. The value of name is always `In-Process`.
usage	String. The usage of the service. The value of usage is always `SourceAndDestination`.

Attribute	Description
	For more information about the usage of an IP service, see "Using the In-Process Service" on page 22.
connected	Boolean. Indicates whether the service is connected. The value of connected is always `true`.
connectTime	Instant. The time when the IP service connects on first usage.
disconnectTime	Instant. The time when the IP service disconnects. The value of disconnectTime is always empty.
activeExclusiveDurable-SubscribersCount	Integer. The number of active exclusive durable subscribers to the IP service.
activeExclusiveDurable-SubscribersIds	List<String>. The identifiers of the active exclusive durable subscribers to the service.
activeSharedDurable-SubscribersCount	Integer. The IP service does not support shared durable subscribers. The value of activeSharedDurableSubscribersCount is always `0`.
activeSharedDurable-SubscribersIds	List<String>. The IP service does not support shared durable subscribers. The value of activeSharedDurableSubscribersIds is always an empty list.
activePriorityDurable-SubscribersCount	Integer. The IP service does not support priority durable subscribers. The value of activePriorityDurableSubscribersCount is always `0`.
activePriorityDurable-SubscribersIds	List<String>. The IP service does not support priority durable subscribers. The value of activePriorityDurableSubscribersIds is always an empty list.
inactiveExclusiveDurable-SubscribersCount	Integer. The number of exclusive durable subscribers to the service that are inactive but still subscribed.

Attribute	Description
inactiveExclusiveDurable-SubscribersIds	List<String>. The identifiers of the exclusive durable subscribers to the service that are inactive but still subscribed.
inactiveSharedDurable-SubscribersCount	Integer. The IP service does not support shared durable subscribers. The value of inactiveSharedDurableSubscribersCount is always `0`.
inactiveSharedDurable-SubscribersIds	List<String>. The IP service does not support shared durable subscribers. The value of inactiveSharedDurableSubscribersIds is always an empty list.
inactivePriorityDurable-SubscribersCount	Integer. The IP service does not support priority durable subscribers. The value of inactivePriorityDurableSubscribersCount is always `0`.
inactivePriorityDurable-SubscribersIds	List<String>. The IP service does not support priority durable subscribers. The value of inactivePriorityDurableSubscribersIds is always an empty list.
nonDurable-SubscribersCount	Integer. The number of non-durable subscribers to the service.
receivedEvents	Long. The number of events received by all subscribers to the IP service since the service became active.
lastSendingStatusDetailslastSendingStatus	String. The status of the last sending operation performed by the IP service. Values are: ■ `Green` - when the last sending operation is successful. ■ `Red` - when the last sending operation has failed.
	String. Information about the last sending status. If the status is `Red`, the explanation contains the exception that caused the sending operation to fail.
sentEvents	Long. The number of events sent to the IP service.

Attribute	Description
acknowledgedSentEvents	Long. The number of events sent to the IP service and successfully acknowledged by the client event processor.
lastSendingTime	Instant. The time when the last event was sent to the client event processor. **Note:** The lastSendingTime attribute reflects the time when an event was last sent to the destination service. The lastSendingTime attribute does not carry information about whether an event was acknowledged by the client event processor. To see whether an event was acknowledged by the destination service, check the lastCompletedAcknowledgementTime attribute.
lastCompletedAcknowled-gementTime	Instant. The last time when an event sent to the IP service was successfully acknowledged by the client event processor.
regularDelivery-InterruptionsCount	Integer. The number of times when the service switched to redelivery mode. The IP service does not support redelivery and the value of regularDeliveryInterruptionsCount is always `0`.

Queue MBean

Retrieves on-disk and in-memory storage details per event type.

MBean Name

```
com.softwareag.events.routing:type=EventTypes,eventType=event_type_name,
category=Queues,queueType=queue_type
```

where *event_type_name* is the name of the event type and *queue_type* is type of storage used for the queue. Values for *queue_type* are `On-Disk` and `In-Memory`.

MBean Attributes

Attribute	Description
status	String. The status of the queue. Values are: ■ `Green`

Attribute	Description
	■ `Yellow` ■ `Red`
statusDetails	String. An explanation of the current status. Values are: ■ `Green` - the utilization is below 90%. ■ `Yellow` - the queue is functioning properly, but the utilization is equal to or greater than 90% but still less than 100%. ■ `Red` - the queue is full.
capacity	Long. The maximum number of events that the queue can store.
currentSize	Long. The number of events currently stored in the queue.
averageUtilization	Float. The average *size* :*capacity* ratio since the first send operation for the event type that uses the queue.
queueBufferFullCount	Integer. The number of times that the queue becomes full, that is when currentSize is equal to capacity.

Subscriber MBean

Monitors the subscribers to Digital Event Services (DES) event types.

MBean Name

```
com.softwareag.events.routing:type=EventTypes,
eventType=event_type_name,category=Subscribers,subscriber=subscriber_name
```

where *event_type_name* is the name of the event type and *subscriber_name* is the unique name that DES generated for the subscriber.

Note: You can retrieve the subscriber ID that you specify when creating a durable subscriber from the subscriberId attribute.

MBean Attributes

Attribute	Description
subscriberId	String. The identifier of a subscriber.
type	String. The type of the subscriber. Values are: ■ `Exclusive durable` ■ `Shared durable` ■ `Priority durable` ■ `Exclusive non-durable` For more information about the different types of subscribers, see *webMethods Service Development Help*.
sourceService	String. The name of the source service from which the subscriber receives events.
status	String. The status of the subscriber. Values are: ■ `Active` - the subscriber is working as expected. ■ `Inactive` - the subscriber is closed but still subscribed. ■ `Not configured` - no source service is configured for the event type, or the source service is removed from the configuration after the subscriber was created. If the source service in a service group is an In-Process (IP) service, the status of shared durable subscribers and priority durable subscribers is `Not configured`. The IP service does not support priority durable and shared durable subscribers. **Note:** When a subscriber is *closed and unsubscribed*, its MBean is unregistered. In this case, the subscriber has no status.
statusDetails	String. An explanation of the current subscriber status.
receivedEvents	Long. The number of events received by the subscriber, including redelivered events.

Attribute	Description
lastReceivedEventTime	Instant. The time when the last event was received by the subscriber.
receivedRedeliveredEvents	Long. The number of redelivered events received by the subscriber.
lastReceivedRedelivered-EventTime	Instant. The time when the last redelivered event was received by the subscriber.
acknowledgedEvents	Long. The number of events acknowledged by the client event processor associated with the subscriber.
lastAcknowledgementTime	Instant. The time of the last event acknowledgement by the client event processor associated with the subscriber.

B Using Digital Event Persistence

Configuring HDFS for Digital Event Persistence

Before you can store events with Digital Event Persistence using HDFS as the storage engine, you must configure the Hadoop cluster. You must copy the custom Hive SerDe and Joda Date/Time libraries from your Digital Event Persistence installation to your HDFS CDH 5.3.0 distribution.

To configure HDFS CDH 5.3.0 as the storage engine for Digital Event Persistence:

1. In your Software AG installation, locate the Joda Date/Time and the Digital Event Persistence Hive SerDe .jar files:
   - joda-time_2.9.3.jar - available in the *Software AG_directory*\common\runtime\bundles\platform\eclipse\plugins directory.
   - com.softwareag.evp.hive.serde_10.1.0.0000-*nnnn* .jar - available in the *Software AG_directory*\common\runtime\bundles\evs\eclipse\plugins directory, where *nnnn* is the build number of your Digital Event Persistence installation.
2. Copy both files to the Hive library directory on all nodes in the Hadoop cluster where Hive is running, for example *CDH5.3.0_directory* /var/lib/hive/lib.
3. Copy both files to the Yarn library directory on all data nodes in the Hadoop cluster, for example *CDH5.3.0_directory* /var/lib/hadoop-yarn/lib or *CDH5.3.0_directory* /var/lib/hadoop-mapreduce/lib, if you are using MapReduce MRv1.
4. Restart Hive.

Adding Dynamic Service Information to Digital Event Types

By default, Digital Event Persistence provides the following digital event types in which you include values for any variables defined in a dynamic service configuration:

- `des.evs.DynamicServiceConfiguration` - contains a configuration field that contains an array of `des.evs.KeyValuePair` event types:

```
des.evs.DynamicServiceConfiguration : event
{
     configuration : des.evs.KeyValuePair[]
}
```

- `des.evs.KeyValuePair` - provides the variable and variable values defined within the dynamic service configuration:

```
des.evs.KeyValuePair : event
{
    key: string
    value : string
}
```

When the Digital Event Persistence service encounters a dynamic service configuration field within a published event, the information provided in the key/value pairs is used to perform variable substitution for the fields in which variables are declared.

You can find the two event types in *Software AG_directory*\common\DigitalEventServices\TypeRepository\eventtypes\des\evs.

Example

The following example shows a digital event type that uses dynamic service configuration by defining a header field of type `des.evs.DynamicServiceConfiguration`.

```
WebM.Des.Sample.OrderManagement.DynamicPurchaseOrderCreated : event
    @EventId(path = "EventId")
    @EventTime(path = "EventStart")
{
    Header            : des.evs.DynamicServiceConfiguration
    EventId           : string
    EventStart        : int64                                          @Timestamp
    OrderNumber       : int32
    OrderDateTime     : int64                                          @Timestamp
    CustomerName      : string
    Attachment        : bytes
    BillingAddress    : WebM.Des.Sample.OrderManagement.PurchaseOrderAddress
    ShippingAddress   : WebM.Des.Sample.OrderManagement.PurchaseOrderAddress
    RequestedShipDate : string
    RequestedShipTime : string
    SalesRepName      : string
    OrderTotal        : float64
    LineItems         : WebM.Des.Sample.OrderManagement.PurchaseOrderItem[]
}
```

Configuring SSL for Digital Event Persistence

You can persist digital events over Secure Sockets Layer (SSL) when you use Elasticsearch as the storage technology. To persist events over SSL, you must perform the following steps at a high level:

1. Enable your Elasticsearch or Event Data Store instance for SSL. For more information, see "Enabling Elasticsearch for SSL" on page 65 and "Securing Communication with Event Data Store" on page 70, respectively.
2. Configure your Digital Event Persistence service of type Elasticsearch to use SSL. For more information about how to configure the service to persist events over SSL, see "Configuring Digital Event Persistence Services for Elasticsearch" on page 19.

Enabling Elasticsearch for SSL

Before you can persist digital events to Elasticsearch over SSL, you must enable SSL for your Elasticsearch storage engine and have the Search Guard plugin installed for Elasticsearch. For more information about how to install and configure SSL for

Elasticsearch 2.3.2, see the Elasticsearch documentation. For information about using the Search Guard plugin, see the Search Guard plugin documentation.

C Using webMethods Event Data Store

About Event Data Store

webMethods Event Data Store is an Elasticsearch 2.3.2 packaging that Software AG provides. Using Command Central, administrators can configure Digital Event Persistence to persist Digital Event Services events to an Event Data Store server instance or cluster.

Starting the Event Data Store Server on Windows

Start an instance of Event Data Store from the Windows Start menu.

To start an Event Data Store Server on Windows

1. Click **Start**.
2. In the **All Programs** menu, click the **Software AG** folder.
3. Click the **Start Servers** folder.
4. Click **Start Event Data Store 10.1**.

Stopping the Event Data Store Server on Windows

Stop an instance of Event Data Store from the Windows Start menu.

To stop an Event Data Store Server on Windows

1. Click **Start**.
2. In the **All Programs** menu, click the **Software AG** folder.
3. Click the **Stop Servers** folder.
4. Click **Stop Event Data Store 10.1**.

Starting, Stopping, and Restarting the Event Data Store Server on Linux

You can start, stop, and restart Event Data Store on Linux by running the following scripts:

- Start Event Data Store - *Software AG_directory*/EventDataStore/bin/startup.sh.
- Stop Event Data Store - *Software AG_directory*/EventDataStore/bin/shutdown.sh.
- Restart Event Data Store - *Software AG_directory*/EventDataStore/bin/restart.sh.

Changing the Event Data Store HTTP Port

The default HTTP port that clients use to make calls to Event Data Store is `9240`. Use the following procedure to change the HTTP port number.

To change the Event Data Store HTTP port

1. In Command Central, navigate to **Environments > Instances > All > *instance_name* > Configuration**.
2. Select **Ports** from the drop-down menu.
3. Click **http port** and specify values for the following fields:

Field	Description
Port Number	Required. The HTTP port number. The default value is `9240`.
Use SSL	Optional. Enable Secure Sockets Layer (SSL) to secure communication with Event Data Store. **Note:** ■ When you enable SSL for the HTTP port, you automatically enable SSL for the TCP port as well. ■ The Event Data Store uses the Search Guard SSL plugin for Elasticsearch. For more information about the Search Guard plugin, see the Search Guard documentation.

4. Optionally, click **Test** to verify your configuration.
5. Save your changes.
6. Restart the Event Data Store instance.

Changing the Event Data Store TCP Port

Besides the HTTP port, clients can use the TCP port to make calls to Event Data Store. In addition, the nodes in an Event Store cluster use the TCP port to communicate with one another. The default TCP port is `9340`.

To change the Event Data Store TCP port

1. In Command Central, navigate to **Environments > Instances > All > *instance_name* > Configuration**.

2. Select **Ports** from the drop-down menu.
3. Click **tcp port** and specify values for the following fields:

Field	Description
Port Number	Required. The TCP port number. The default value is `9340`.
Use SSL	Optional. Enable Secure Sockets Layer (SSL) for the TCP port. **Note:** ■ When you enable SSL for the TCP port, you automatically enable SSL for the HTTP port as well. ■ The Event Data Store uses the Search Guard SSL plugin for Elasticsearch. For more information about the Search Guard plugin, see the Search Guard documentation.

4. Optionally, click **Test** to verify your configuration.
5. Save your changes.
6. Restart the Event Data Store instance.

Important: If you change the default TCP port, you must change the respective value in the **Clustering** configuration.

Securing Communication with Event Data Store

When you install Event Data Store it comes with a pre-configured SSL certificate, and default *keystore* and *trustore* files. The keystore and truststore function as repositories for the storage of keys and certificates necessary for SSL authentication, encryption/decryption, and digital signing/verification services. You can find the default truststore and keystore files in the following locations:

- *Software AG_directory*/EventDataStore/plugins/search-guard-2/sgconfig/kirk-keystore.jks
- *Software AG_directory*/EventDataStore/plugins/search-guard-2/sgconfig/truststore.jks

For more information about how to enable or disable SSL for Event Data Store, see "Enabling SSL for Event Data Store" on page 71 and "Disabling SSL for Event Data Store" on page 71, respectively.

The Event Data Store is enabled for SSL through the Elasticsearch Search Guard plugin. The Search Guard plugin provides an sgadmin command line tool that you can use to

customize your Search Guard configuration. To modify the Search Guard configuration of an SSL-enabled Event Data Store, you must authenticate the sgadmin tool with a .jks-based keystore and truststore. Run one of the following scripts to access the sgadmin tool:

- For Linux - *Software AG_directory*/EventDataStore/repo/search-guard-2/tools/sgadmin.sh.
- For Windows - *Software AG_directory*\EventDataStore\repo\search-guard-2\tools\sgadmin.bat.

If you use Event Data Store in a production environment, you should replace the Event Data Store default certificates, keystore and truststore files with custom files. For more information about creating keystores and truststores, importing keys and certificates into keystores and truststores, and other operations with these files, see the documentation for your certificate management tool.

Enabling SSL for Event Data Store

You automatically enable SSL for Event Data Store by running one of the following scripts:

- For Linux - *Software AG_directory*/EventDataStore/bin/enable_ssl.sh.
- For Windows - *Software AG_directory*\EventDataStore\bin\enable_ssl.bat.

You can also enable SSL for Event Data Store when you configure the Event Data Store ports in Command Central. For more information about configuring the Event Data Store ports, see "Changing the Event Data Store HTTP Port" on page 69 and "Changing the Event Data Store TCP Port" on page 69.

Disabling SSL for Event Data Store

Note: Before you run the script for disabling SSL, you must disconnect all clients connected to the Event Data Store server.

You automatically disable SSL for Event Data Store by running one of the following scripts:

- For Linux - *Software AG_directory*/EventDataStore/bin/disable_ssl.sh.
- For Windows - *Software AG_directory*\EventDataStore\bin\disable_ssl.bat.

The Event Data Store Keystores

By default, Event Data Store has the following pre-configured keystores:

- **HTTP Keystore** - A keystore for HTTP clients.
- **TCP Keystore** - A keystore for TCP clients.

- **sgadmin Keystore** - A keystore that authenticates the sgadmin tool.

You cannot add or remove the pre-configured keystores. However, you can use custom keystore files instead. For more information about creating keystores, see the documentation of your certificate management tool.

Configuring the Event Data Store HTTP Keystore

Use the following procedure to modify the keystore for the HTTP port of the Search Guard plugin.

To modify the keystore for the HTTP port of the Search Guard plugin

1. In Command Central, navigate to **Environments > Instances > All > *instance_name* > Configuration**.
2. Select **Keystores** from the drop-down menu.
3. In the **Alias** column, click **HTTP_KEYSTORE** and then click **Edit**.
4. Specify values for the following fields:

Field	Description
Description	Optional. Specify a description for the keystore for the HTTP port of the Search Guard plugin.
Location	Required. Specify the absolute filepath to the Java keystore file as follows: *folder/sub_folder/filename* . The default value is: `../plugins/search-guard-2/sgconfig/node-0-keystore.jks`
Password	Optional. Specify the password for the keystore.

5. Optionally, click **Test** to verify that your configuration is valid.
6. Save your changes.
7. Restart the Event Data Store instance.

Configuring the Event Data Store TCP Keystore

Use the following procedure to modify the keystore for the TCP port of the Search Guard plugin.

To modify the keystore for the TCP port of the SearchGuard plugin

1. In Command Central, navigate to **Environments > Instances > All > *instance_name* > Configuration**.
2. Select **Keystores** from the drop-down menu.

3. In the **Alias** column, click **TCP_KEYSTORE** and then click **Edit**.
4. Specify values for the following fields:

Field	Description
Description	Optional. Specify a description for the keystore for the TCP port of the Search Guard plugin.
Location	Required. Specify the absolute filepath to the Java keystore file as follows: *folder/sub_folder/filename* . The default value is: `../plugins/search-guard-2/sgconfig/node-0-keystore.jks`
Password	Optional. Specify the password for the keystore.

5. Optionally, click **Test** to verify that your configuration is valid.
6. Save your changes.
7. Restart the Event Data Store instance.

Configuring the Event Data Store sgadmin Keystore

The sgadmin tool authenticates itself agains the SSL-enabled Event Data Store with a keystore.

To modify the keystore for the sgadmin tool

1. In Command Central, navigate to **Environments > Instances > All > *instance_name* > Configuration**.
2. Select **Keystores** from the drop-down menu.
3. In the **Alias** column, click **SGADMIN_KEYSTORE** and then click **Edit**.
4. Specify values for the following fields:

Field	Description
Description	Optional. Specify a description for the keystore for the sgadmin tool.
Location	Required. Specify the absolute filepath to the Java keystore file as follows: *folder/sub_folder/filename* . The default value is: `../plugins/search-guard-2/sgconfig/sgadmin-keystore.jks`

Field	Description
Password	Optional. Specify the password for the keystore.

5. Optionally, click **Test** to verify that your configuration is valid.
6. Save your changes.
7. Restart the Event Data Store instance.

Configuring the Event Data Store Truststore

By default, Event Data Store has a single pre-configured truststore for both the TCP and the HTTP ports.

If you use Event Data Store in a production environment, replace the Event Data Store default trustsore file with a custom file. For more information about creating truststore files, see the documentation of your certificate management tool.

To modify the default Event Data Store truststore

1. In Command Central, navigate to **Environments > Instances > All > *instance_name* > Configuration**.
2. Select **Truststores** from the drop-down menu and click **Edit**.
3. Specify values for the following fields:

Field	Description
Description	Optional. Specify a description for the truststore for the Search Guard plugin.
Location	Required. Specify the absolute filepath to the truststore file as follows: *folder/sub_folder/filename* . The default value is: `../plugins/search-guard-2/sgconfig/truststore.jks`
Password	Optional. Specify the password for the truststore.

4. Optionally, click **Test** to verify that your configuration is valid.
5. Save your changes.
6. Restart the Event Data Store instance.

Configuring an Event Data Store Cluster

You can persist digital events to a cluster of Event Data Store servers. You must specify at least one host and port pair for your configuration in Command Central to be valid. The Event Data Store comes with a default host and port pair.

To configure an Event Data Store cluster

1. In Command Central, navigate to **Environments > Instances > All > *instance_name* > Configuration**.
2. Select **Clustering** from the drop-down menu, and then click **Edit**.
3. Specify values for the following fields:

Field	Description
Cluster Name	Required. The name of the cluster.
Cluster Discovery Nodes	Required. Click + and then do the following to add host and port information for each cluster endpoint: ■ In the **Host** column, specify the host information for a cluster endpoint. The default host is `localhost`. ■ In the **Port** column, specify the port for a cluster endpoint. The default port is `9340`.

4. Optionally, click **Test** to verify that your configuration is valid.
5. Save your changes.
6. Restart the Event Data Store instance.

Configuring Custom Event Data Store Properties

You can specify custom properties for your Event Data Store configuration.

To specify custom properties for Event Data Store

1. In Command Central, navigate to **Environments > Instances > All > *instance_name* > Configuration**.
2. Select **Properties** from the drop-down menu and click **Edit**.
3. In the **Content** field, specify custom parameters. Use YAML syntax and the *property_name*:*value* format.
4. Restart the Event Data Store instance.

Commands that Event Data Store Supports

Event Data Store supports the Platform Manager commands listed in the following table. The table lists where you can find information about each command.

Commands	For more information, see...
`sagcc get configuration data`	For general information about the command, see *Software AG Command Central Help*. For information about the configuration types that Event Data Store supports, see "Configuration Types that Event Data Store Supports" on page 77.
`sagcc update configuration data`	For general information about the command, see *Software AG Command Central Help*. For information about the configuration types that Event Data Store supports, see "Configuration Types that Event Data Store Supports" on page 77.
`sagcc get configuration instances`	For general information about the command, see *Software AG Command Central Help*. For information about the configuration types that Event Data Store supports, see "Configuration Types that Event Data Store Supports" on page 77.
`sagcc list configuration instances`	For general information about the command, see *Software AG Command Central Help*. For information about the configuration types that Event Data Store supports, see "Configuration Types that Event Data Store Supports" on page 77.
`sagcc get configuration types`	For general information about the command, see *Software AG Command Central Help*.

Commands	For more information, see...
	For information about the configuration types that Event Data Store supports, see "Configuration Types that Event Data Store Supports" on page 77.
`sagcc list configuration types`	For general information about the command, see *Software AG Command Central Help*. For information about the configuration types that Event Data Store supports, see "Configuration Types that Event Data Store Supports" on page 77.
`sagcc exec configuration validation update`	For general information about the command, see *Software AG Command Central Help*. For information about the configuration types that Event Data Store supports, see "Configuration Types that Event Data Store Supports" on page 77.
`sagcc exec lifecycle`	For general information about the command, see*Software AG Command Central Help*. For Event Data Store-specific information about the command, see " Run-Time Monitoring Statuses for Event Data Store" on page 78.
`sagcc get monitoring`	For general information about the command, see *Software AG Command Central Help*. For Event Data Store-specific information about the command, see " Run-Time Monitoring Statuses for Event Data Store" on page 78.

Configuration Types that Event Data Store Supports

The Event Data Store run-time component supports the following configuration types:

Configuration Type	Use to configure...
COMMON-CLUSTER	Settings for an Event Data Store cluster. You can configure the name of the cluster and the host and port pairs of the server endpoints of the cluster. **Note:** The changes that you make to a cluster configuration take effect after you restart Event Data Store.
COMMON-KEYSTORES	Configuration instance for a keystore alias that identifies a keystore file.
COMMON-PORTS	Configuration instances for HTTP and TCP ports.
COMMON-TRUSTSTORES	Configuration instance for a truststore alias that identifies a truststore file.
CUSTOM-PROPERTIES	Additional properties for the configuration of an Event Data Store server.

Run-Time Monitoring Statuses for Event Data Store

The following table lists the run-time statuses that the Event Data Store run-time component can return in response to the `sagcc get monitoring state` command, along with the meaning of each run-time status.

Run-time Status	Meaning
ONLINE	The Event Data Store instance is running.
STOPPED	The Event Data Store instance is stopped.

Lifecycle Actions for Event Data Store

The following table lists the actions that Event Data Store supports with the `sagcc exec lifecycle` command. You can also perform these actions in the Command Central web user interface.

Action	Description
`start`	Starts the Event Data Store instance.
`stop`	Stops the Event Data Store instance.
`restart`	Restarts the Event Data Store instance.