software AG

Using Digital Event Services to Communicate between Software AG Products

Version 10.3

October 2018

Table of Contents

About this Guide

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

Convention	Description
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Software AG Documentation Website

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

Software AG Empower Product Support Website

If you do not yet have an account for Empower, send an email to "empower@softwareag.com" with your name, company, and company email address and request an account.

Once you have an account, you can open Support Incidents online via the eService section of Empower at "https://empower.softwareag.com/".

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".

If you have any questions, you can find a local or toll-free number for your country in our Global Support Contact Directory at "https://empower.softwareag.com/public_directory.asp" and give us a call.

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.

Data Protection

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

1   Understanding Digital Event Services

Digital Event Services Overview

Software AG 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.

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.

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*.

Support of Digital Event Services

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

- webMethods Integration Server

- Apama

- Software AG MashZone NextGen

- webMethods Business Rules

- 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 (DES) 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. Administrators can define on-disk and in-memory capacity globally and per event type.

The Store-and-Forward Queue

By default, Digital Event Services (DES) stores events in a local queue until DES receives an acknowledgement that the events have reached the configured destination services. When you configure a service group, you can specify destination services that listen for events of the event types that you associate with the service group. Before these events reach a destination service, DES stores the events in a local *store-and-forward queue*. The store-and-forward queue ensures that no events are lost when a destination service becomes unavailable.

In the Command Central web user interface, you can modify the delivery mode for each event type. Depending on the delivery mode you specify, the store-and-forward queue for the event type can be held on-disk or in-memory. After an event is added to

the queue, DES is ready to accept new events. In the meantime, DES delivers the queued events to the destination services in the order in which they were added to the queue.

Consideration for Using the Store-and-Forward Queue

■ When a destination service becomes unavailable, DES tries to redeliver the events until the destination service becomes available. Meanwhile, events accumulate in the store-and-forward queue until the capacity of the queue is reached. When the capacity of the queue is reached, depending on the runtime in which DES is embedded, DES either throws an exception or blocks the sending of further events until some of the accumulated events reach the destination service.

■ When you stop DES while it is in the process of sending events, if the destination service is active, DES allows for a grace period of 30 seconds during which DES tries to route the events that remain on the store-and-forward queue to the destination service. After the grace period is over, depending on the delivery mode of the events that you send, DES does the following:

■ If the events are from an event type that has a persistent delivery mode, after the grace period is over, any events that do not reach a destination service remain in storage until you start DES again and the events can be delivered.

■ If the events are from an event type that has non-persistent delivery mode, DES removes any events that remain waiting for the destination in the store-and-forward queue after the grace period is over.

■ When you remove a destination service from a service group:

■ If a destination service is available and you remove it from a service group, DES gives this service a grace period of 30 seconds to process and acknowledge events in the store-and-forward queue that wait for this destination service. DES removes any events that remain waiting for the destination in the store-and-forward queue after the grace period is over.

■ If a destination service is currently unavailable and you remove it from a service group, DES removes any events in the store-and-forward queue that wait for this service to become available.

Related Topics

Non-Queued Delivery of Digital Events

If you configure a service group to have a single destination service, you can choose to bypass the local store-and-forward queue and instead use *non-queued delivery*. You can

choose this option when you do not want events to accumulate in the store-and-forward queue.

You can use non-queued delivery for a service group that has a destination service of type Universal Messaging or Digital Event Persistence.

When you configure event type associations in the Command Central web user interface, you can specify a persistent or a non-persistent delivery mode. Bypassing the store-and-forward queue and using non-queued delivery does not affect the delivery mode preference you configure.

Note: When you use non-queued delivery, DES ignores the storage settings you specify for the store-and-forward queue.

Considerations for Using Non-Queued Delivery

- When you use non-queued delivery and an event fails to reach the destination service, depending on the runtime where DES is embedded, DES either throws an exception that notifies the application that the destination service is unavailable or blocks the sending of further events until the destination service becomes available.

- When you stop DES while it is in the process of sending events, DES does not guarantee that all events reach the configured destination service before it shuts down. After you restart DES, it does not deliver any events that you sent after DES stopped.

Related Topics

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.

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 and 5.6.4

■ 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.

2   Administering Digital Event Services

About Administering Digital Event Services

You can administer Software AG Digital Event Services (DES) using the Command Central web user interface, templates, or command-line interface.

Using the Command Central web user interface, 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.

Developers can also provision and configure DES using the Command Central templates available in Software AG's GitHub project "sagdevops-templates.git". You can use the following templates as samples that you adapt to fit your own use case:

■ sag-des - this template shows how to provision DES and how to configure a default messaging service.

■ sag-des-config - this templates shows how to provision DES, so that it uses secure communication with Java keystores and truststores. For this purpose, the template also shows how to configure a custom service group and associate one digital event type with it. You can also use secure communication with the default service group.

Note: Integration Server and Apama developers can also use the DES command-line tool to configure messaging connectivity without using Command Central. You 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.

For information about using Command Central CLI commands and Command Central templates, 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 each **Field** in the table as outlined in the **Description** column:

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. The host and port of the Universal Messaging server to which the service connects. Supported transport protocols are NSP (TCP connection) and NSPS (SSL connection). The default value is `nsp://localhost:9000`. **Note:** When you configure an SSL connection to Universal Messaging, you must also specify values for **Keystore Alias**, **Key Alias**, and **Truststore Alias**.
Keystore Alias	Optional. The alias that identifies the keystore for establishing an SSL connection to Universal Messaging. You define a client keystore as described in "Defining a DES Keystore" on page 19.
Key Alias	Optional. The label that identifies a specific key within the keystore for establishing an SSL connection to Universal Messaging.
Truststore Alias	Optional. The alias that identifies the truststore for establishing an SSL connection to Universal Messaging.

Field	Description
	You define a client truststore as described in "Defining a DES Truststore" on page 19.
User Name	Optional. The name of the user authorized to connect to a Universal Messaging server with server-side authentication enabled.
Password	Optional. The password of the user authorized to connect to a Universal Messaging server with server-side authentication enabled.
	If you only update the **Password** field, you must restart the runtime where the DES component is embedded for the new password to take effect.

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: When performing validation, Command Central checks whether the specified configuration information complies with the field requirements. This action does not ping a server to verify that a connection is possible.

5. Save your changes.

Using Universal Messaging Services with SSL

Digital Event Services (DES) supports the Universal Messaging NSPS protocol for secure communication over Secure Sockets Layer (SSL).

To enable DES to communicate with an SSL-enabled Universal Messaging server over NSPS, you must first define a keystore and a truststore file for DES. 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 must define at least one keystore and one truststore for DES. If you cannot create new keystore and truststore files, you can refer to existing keystore and truststore files that are used for Universal Messaging communication and contain the appropriate private keys and certificates.

For more information about configuring Universal Messaging for SSL communication, see the Universal Messaging documentation. For more information about operations

with keystore and truststore files, see the documentation for your certificate management tool.

Defining a DES Keystore

To use Digital Event Services (DES) with an SSL-enabled Universal Messaging server, you must specify the keystore that contains the private keys and certificates that authenticate DES for communication over NSPS.

To define keystores for DES

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

2. Select **Keystores** from the drop-down menu, and then click ➕.

3. Specify values for each **Field** in the table as outlined in the **Description** column:

Field	Description
Keystore Name	Required. A unique name for the keystore. The keystore name is not case-sensitive and must start with a character. You can use the following characters as separators: . (dot) and – (dash).
Description	Optional. A description for the keystore.
Location	Optional. The absolute file path to the location of the keystore.
Password	Optional. The password for the keystore.

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

> **Note:** When performing validation, Command Central checks whether the specified configuration information complies with the field requirements. Command Central does not check whether the information you provide is accurate.

5. Save your changes.

6. Restart the runtime in which DES is embedded.

Defining a DES Truststore

To use Digital Event Services (DES) with an SSL-enabled Universal Messaging server, you must specify the DES truststore that contains the trusted Universal Messaging certificates for establishing an SSL connection.

To define truststores for DES

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

2. Select **Truststores** from the drop-down menu, and then click ➕.

3. Specify values for each **Field** in the table as outlined in the **Description** column:

Field	Description
Truststore Name	Required. A unique name for the truststore. The truststore name is not case-sensitive and must start with a character. You can use the following characters as separators: . (dot) and – (dash).
Description	Optional. A description for the truststore.
Location	Required. The absolute file path to the location of the truststore.
Password	Optional. The password for the truststore.

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

> **Note:** When performing validation, Command Central checks whether the specified configuration information complies with the field requirements. Command Central does not check whether the information you provide is accurate.

5. Save your changes.

6. Restart the runtime in which DES is embedded.

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.

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 $.

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 each **Field** in the table as outlined in the **Description** column:

Field	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 Uniform Resource Identifier (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`.

Field	Description
	You can use dynamic service configuration to specify the Hive Server, for example: `jdbc:hive2://$host$:port`.
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 user name 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.

Related Topics

Configuring Digital Event Persistence Services for Elasticsearch

With Digital Event Persistence services, you can store events to an Elasticsearch 2.3.2 or 5.6.4 storage engine. When using Elasticsearch, you can persist events using Secure Sockets Layer (SSL).

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

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.

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** or **Elasticsearch 5.6.4** for the service type.

4. Specify values for each **Field** in the table as outlined in the **Description** column:

Field	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.

Field	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 Uniform Resource Identifier (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

Field	Description
	than 0 sets the number of pending bulk indexing tasks 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`.
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.

Related Topics

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**.

Related Topics

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 (DES) 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.

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 as outlined in the **Description** column of this table:

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.
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, in the **Use Local Store&Forward Queue** field, select one of the following values:

   - **Yes** -Default. DES stores events in a local store-and-forward queue until DES receives an acknowledgement that the events have reached the configured destination services.

   - **No** - The events bypass the local store-and-forward queue and reach the destination service directly.

     Note: You can use this option only when a single destination service is configured in the service group.

   For more information about using the local store-and-forward queue and how it differs from non-queued delivery, see "The Store-and-Forward Queue" on page 11 and "Non-Queued Delivery of Digital Events" on page 12.

7. 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.

8. Save your changes.

Related Topics

Configuring Custom Service Groups

By default, Digital Event Services (DES) 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.

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

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 each **Field** in the table as outlined in the **Description** column:

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

Field	Description
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 the SHIFT button 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.
	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

Field	Description
	original service group, so that the event type shows in the dual list box.
Use Local Store&Forward Queue	Optional. Specify whether DES stores the events of the event types associated with the service group in a local store-and-forward queue. Select one of the following options:

■ **Yes** - Default. DES stores events in a local store-and-forward queue until DES receives an acknowledgement that the events have reached the configured destination services.

■ **No** - The events bypass the local store-and-forward queue and reach the destination service directly.

> **Note:** You can use this option only when a single destination service is configured in the service group.

For more information about using the local store-and-forward queue and how it differs from non-queued delivery, see "The Store-and-Forward Queue" on page 11 and "Non-Queued Delivery of Digital Events" on page 12.

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

6. Save your changes.

Related Topics

Considerations When Modifying a Service Group Configuration

When you change the Universal Messaging source service of a service group, make sure that the Universal Messaging server specified in the original source service is running when Digital Event Services (DES) is applying the configuration change. DES applies the new configuration shortly after you make the changes in Command Central when the runtime in which DES is embedded is started, or the next time when the runtime is started.

If the Universal Messaging server for the original source service is unavailable when DES is processing the configuration change, the durable subscription objects on the channel of the source Universal Messaging server are not deleted. As a result, any events sent to this Universal Messaging channel are kept on it and consume memory. Even if

you configure the new source service to use the original Universal Messaging server again, the events sent prior to this change do not reach the subscribers for the event type.

Configuring Event Type Associations

When events of a particular event type are sent or received, Digital Event Services (DES) 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.

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 values for each **Field** in the following table as outlined in the **Description** column:

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.
	■ **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.

Field	Description
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.
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.
Service Group	Select a service group from the drop-down menu with which to associate the event type.
	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.

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

6. Save your changes.

Related Topics

Configuring Storage Settings

Depending on the delivery mode that you configure for an event type, events sent to Digital Event Services (DES) are stored on-disk or in-memory. Use the following procedure to configure storage settings for DES.

> **Note:** Depending on the runtime in which DES is embedded, events are emitted synchronously or asynchronously. When events are emitted synchronously, DES ignores the **Sync Batch Size** and **Sync Timeout** settings.

To configure the storage settings for DES

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 each **Field** in the table as outlined in the **Description** column:

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.
	You must restart the runtime where the DES component is embedded for the changes to the **Storage Location** property to take effect.
Sync Batch Size	Required. Defines the number of events that are added to the store-and-forward queue before the queue is persisted to disk. DES ignores the **Sync Batch Size** setting if events are sent synchronously.
	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. DES ignores the **Sync Timeout** setting if events are sent synchronously.
	The default value is `500` milliseconds.

Field	Description
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.
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.

5. Save your changes.

Related Topics

Using DES Command Line Tools

Digital Event Services (DES) provides several command line (CLI) tools that you use along with the user interfaces in Command Central and Software AG Designer to administer DES. You can find the following command line tools in the *Software AG_directory*/common/ lib directory:

- `events-console.jar` - Use this tool to monitor the publishing of digital events.

- `events-generator.jar` - Use this tool to generate streams of digital events.

- `events-configuration-tool.jar` - Use this tool to configure message connectivity.

- `events-passman-tool.jar` - Use this tool to change the DES master password.

To access the usage information and run any of the available CLI tools for administering DES, open a command prompt and type `java -jar` followed by *Software_AG_directory*\common\lib\ and the file name of the tool, for example:

```
java -jar Software_AG_directory\common\lib\events-console.jar
```

Related Topics

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).

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.

Related Topics

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.

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 (DES) 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.

Managing Digital Event Services Licenses

You must have a valid license file to use Digital Event Services (DES). When you install DES, Software AG Installer asks you to provide your license file for DES. If you do not provide a license file, DES is installed with a 30-day trial license. Installer copies the license file, named license.xml, to the *Software AG_directory*/common/ DigitalEventServices/license directory.

DES requires a single license file per Software AG installation. This license applies to all runtimes that use digital events to communicate.

To overwrite the trial license or change a license that has expired, you must:

1. Register your license, as described in "Registering a DES License in Command Central" on page 37.

2. Update the license, as described in "Updating a Digital Event Services License in Command Central" on page 38.

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

If you do not use Command Central to administer DES, you can manually update the DES license by copying the new license to the *Software AG_directory*/common/ DigitalEventServices/license directory. The new file must be named license.xml.

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

Related Topics

"Viewing Digital Event Services License Details"

Registering a DES License in Command Central

Use the following procedure to add a Digital Event Services (DES) license file in the Command Central web user interface.

To add a Digital Event Services license

1. In Command Central, navigate to **Licensing > Keys**.

2. Click ⊞, and then select **Add License Key**.

3. Specify values for each **Field** in the table as outlined in the **Description** column:

Field	Description
License key	Required. The location of the license file.

Field	Description
Alias	Required. The alias name of the license key.

4. Click **Add**.

Updating a Digital Event Services License in Command Central

Before updating a Digital Event Services (DES) license, you must register the license file in Command Central.

To update a Digital Event Services license

1. In Command Central, navigate to **Environments > Instances > All > SPM > DES Common > Configuration**.

2. Select **Licenses** from the drop-down menu, and then click **DigitalEventServices**.

3. Click **Edit** and select a license key alias in the **License Alias** field.

4. Save your changes.

Viewing Digital Event Services License Details

You can view license details for Digital Event Services (DES), including the status and expiration date of the license, the actual license key, and product information in the Command Central web user interface.

To view license details

1. In Command Central, navigate to **Environments > Instances > All > SPM > DES Common > Configuration**.

2. Select **Licenses** from the drop-down menu, and then click **DigitalEventServices**.

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   Deployment of Digital Event Types

Deployment of Digital Event Types

Deployment, in the context of Software AG 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.

4 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 Software AG 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

The following table lists the attributes of the Event Type Information MBean and their descriptions:

Attribute	Description
deliveryMode	String. The delivery mode of the event type. Values are: ■ `Persistent` ■ `Non-Persistent`

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

The following table lists the attributes of the Universal Messaging Service MBean and their descriptions:

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.
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.
activeSharedDurable-SubscribersIds	List<String>. The identifiers of the active shared durable subscribers to the service. If the service

Attribute	Description
	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.
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.
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.
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.
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.
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.

Attribute	Description
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 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.

Attribute	Description
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. **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.
regularDelivery-InterruptionsCount	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

The following table lists the attributes of the In-Process Service MBean and their descriptions:

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`.
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.

Attribute	Description
inactiveExclusiveDurable-SubscribersCount	Integer. The number of exclusive durable subscribers to the service that are inactive but still subscribed.
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.

Attribute	Description
	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.
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

The following table lists the attributes of the Queue MBean and their descriptions:

Attribute	Description
status	String. The status of the queue. Values are: ■ Green ■ 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.
usedMemorySize	Long. The memory space that events currently occupy in the store. If an error occurs while reading the value from the store, the value you retrieve is -1.
usedDiskSpace	Long. The bytes on disk that events currently occupy in the store.

Attribute	Description
	If the event type is configured with non-persistent delivery mode, or if an error occurs while reading the value from the store, the value you retrieve is -1.
freeFileSpacePercentage	Long. The percentage of free space in the store.
	If the event type is configured with non-persistent delivery mode, or if an error occurred while reading the value from the store, the value you retrieve is -1.

Subscriber MBean

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

MBean 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

The following table lists the attributes of the Subscriber MBean and their descriptions:

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.

Attribute	Description
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.
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_10.3.0.0000-*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         : time
    OrderNumber        : int32
    OrderDateTime      : time
    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 for SSL.

2. Configure your Digital Event Persistence service of type Elasticsearch to use SSL.

Related Topics

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