10
DBMS_SAGA_ADM

The `DBMS_SAGA_ADM` package provides a collection of saga administration functions and procedures to define and manage saga participants, coordinators and brokers.

This chapter contains the following topics:

- DBMS_SAGA_ADM Overview
- DBMS_SAGA_ADM Security Model
- Summary of DBMS_SAGA_ADM Subprograms

DBMS_SAGA_ADM Overview

The `SYS.DBMS_SAGA_ADM` package provides the saga administration APIs to manage sagas.

The saga administration APIs implemented using the `DBMS_SAGA_ADM` package enables the administrators to define and manage saga participants, coordinators and brokers. The package APIs are described here for your reference.

DBMS_SAGA_ADM Security Model

A participant must invoke the procedures in `DBMS_SAGA_ADM` package from inside its schema and PDB. Invocation of APIs in the `DBMS_SAGA_ADM` package requires the `SAGA_ADM_ROLE` role.

Summary of DBMS_SAGA_ADM Subprograms

This table lists and briefly describes the `DBMS_SAGA_ADM` package subprograms.

Table 10-1 DBMS_SAGA_ADM Package Subprograms

Subprogram	Description
ADD_COORDINATOR Procedure	Adds a saga coordinator to the saga framework
DROP_COORDINATOR Procedure	Drops the given coordinator from the saga framework.
ADD_PARTICIPANT Procedure	Adds a participant to the local PDB and the broker
DROP_PARTICIPANT Procedure	Drops the given participant on the local PDB and the broker if there are no pending sagas and pending saga messages for the participant.
ADD_BROKER Procedure	Adds a broker to the saga framework and creates a JMS topic for the broker.
DROP_BROKER Procedure	Drops a broker and the associated JMS topic.
REGISTER_SAGA_CALLBACK Procedure	Allows users to create or modify a callback package if created using `add_participant` procedure.

ADD_BROKER Procedure

This procedure adds a broker to the saga framework.

As a part of adding a broker, a JMS topic is created for the broker. Both incoming and outgoing message propagation channels from participants to the broker use the broker's JMS topic.

Syntax

```
add_broker(
 broker_name                IN varchar2,
 broker_schema              IN varchar2 default sys_context('userenv', 'CURRENT_USER'),
 storage_clause             IN varchar2 default NULL,
 queue_partitions           IN number DEFAULT 1,
 version                    IN number default 1
 );
```

Parameters

Table 10-2 ADD_BROKER Procedure Parameters

Parameter	Description
broker_name	A unique broker name.
broker_schema	The schema for the microservice participant being added. If no value is provided, CURRENT_USER is used.
storage_clause	The storage parameter is included in the CREATE_TRANSACTIONAL_EVENT_QUEUE statement when the inbound and outbound JMS topics are created. The storage_clause argument can take any text that you can use in a standard DBMS_AQADM.CREATE_TRANSACTIONAL_EVENT_QUEUE storage_clause argument.
queue_partitions	The number of partitions for the Saga entity IN/OUT queues. This allows for higher parallelism and throughput for the Saga entity. DEFAULT is 1.
version	Saga version

DROP_BROKER Procedure

This procedure drops a broker and the associated JMS topic.

A broker can only be dropped if there are no registered participants and pending messages for the broker.

Syntax

```
drop_broker(broker_name      IN varchar2)
```

Parameters

Table 10-3 DROP_BROKER Procedure Parameters

Parameter	Description
broker_name	The broker name to be dropped.

ADD_COORDINATOR Procedure

This procedure adds a Saga coordinator to the Saga framework. A Saga coordinator acts as a transaction manager for Sagas. The Saga coordinator maintains the state of a Saga across various participants.

Syntax

```
add_coordinator(
 coordinator_name       IN varchar2,
 coordinator_schema     IN varchar2 DEFAULT sys_context('USERENV', 'CURRENT_USER'),
 storage_clause         IN varchar2 DEFAULT NULL,
 dblink_to_broker       IN varchar2 DEFAULT NULL,
 mailbox_schema         IN varchar2,
 broker_name            IN varchar2,
 dblink_to_coordinator  IN varchar2 DEFAULT NULL,
 queue_partitions       IN number DEFAULT 1,
 listener_count         IN number DEFAULT -1,
 version                IN number default 1);
```

Parameters

Table 10-4 ADD_COORDINATOR Procedure Parameters

Parameter	Description
coordinator_name	A unique name for the Saga coordinator.
coordinator_schema	The schema for the Saga coordinator being added. The parameter uses CURRENT_USER if no value is specified.
storage_clause	The storage parameter is included in the CREATE_TRANSACTIONAL_EVENT_QUEUE statement when the inbound and outbound JMS topics are created. The storage_clause argument can take any text that you can use in a standard DBMS_AQADM.CREATE_TRANSACTIONAL_EVENT_QUEUE storage_clause argument.
dblink_to_broker	Pre-created database link to connect from the coordinator PDB to the broker PDB.
mailbox_schema	The schema for the mailbox owner at the broker.
broker_name	The name of the broker.
dblink_to_coordinator	Pre-created database link to connect from the broker PDB to the coordinator PDB.
queue_partitions	The number of partitions for the Saga entity IN/OUT queues. This allows for higher parallelism and throughput for the Saga entity. DEFAULT is 1.

> **Note:**
>
> The number of queue partitions should be equal to the partition count specified while performing the add_broker() operation.

Table 10-4 (Cont.) ADD_COORDINATOR Procedure Parameters

Parameter	Description
listener_count	The number of concurrent Saga listener processes consuming inbound messages for the Saga coordinator. listener_count can have one of the following values: • DBMS_SAGA_ADM.AQ_NTFN, which is also the DEFAULT. In this case, the Saga coordinator processes messages using the AQ notification mechanism. • DBMS_SAGA_ADM.AUTO_NTFN- In this case, the Saga coordinator processes messages using job queue processes that dequeue from the coordinator queue. The job automatically adds and removes jobs depending on the queue depth. • If any positive number (N) is specified, then N number of fixed jobs are created to process the messages for the Saga coordinator.
version	Saga version

DROP_COORDINATOR Procedure

This procedure drops the given coordinator and disables message propagation between the coordinator and the broker. A coordinator can only be dropped if there are no participants associated with the coordinator.

Syntax

```
drop_coordinator(coordinator_name  IN varchar2)
```

Parameters

Table 10-5 DROP_COORDINATOR Procedure Parameters

Parameter	Description
coordinator_name	The name of the saga coordinator that is being dropped.

ADD_PARTICIPANT Procedure

This procedure adds a Saga participant to the database and the broker.

A participant name is unique to the broker where the participant is added. The ADD_PARTICIPANT procedure creates system-defined inbound and outbound JMS topics for the participant. A JMS topic is also created at the broker's side if the topic does not exist. Message propagation relationships are set up from:

- The outbound JMS topic of the participant to the JMS topic of the broker.
- The JMS topic of the broker to the inbound JMS topic of the participant.

Syntax

```
add_participant(
 participant_name           IN varchar2,
 participant_schema         IN varchar2 DEFAULT sys_context('USERENV', 'CURRENT_USER'),
 storage_clause             IN varchar2 DEFAULT NULL,
 coordinator_name           IN varchar2 DEFAULT NULL,
```

```
dblink_to_broker           IN varchar2,
mailbox_schema             IN varchar2,
broker_name                IN varchar2,
callback_schema            IN varchar2 DEFAULT sys_context('USERENV','CURRENT_USER'),
callback_package           IN varchar2,
dblink_to_participant      IN varchar2,
queue_partitions           IN number DEFAULT 1,
version                    IN number DEFAULT 1
);
```

Parameters

Table 10-6 ADD_PARTICIPANT Procedure Parameters

Parameter	Description
participant_name	A unique name for the saga participant.
participant_schema	The schema for the saga participant being added. The parameter uses CURRENT_USER if no value is specified.
storage_clause	The storage parameter is included in the CREATE_TRANSACTIONAL_EVENT_QUEUE statement when the inbound and outbound JMS topics are created. The storage_clause argument can take any text that you can use in a standard DBMS_AQADM.CREATE_TRANSACTIONAL_EVENT_QUEUE storage_clause argument.
coordinator_name	The name of the saga coordinator. If no value is provided and create_coordinator is TRUE, a system generated name is assumed.
dblink_to_broker	The pre-created database link to connect from the participant PDB to broker PDB.
mailbox_schema	The schema for the mailbox owner at the broker's side.
broker_name	The name of the broker. If no value is provided, the mailbox_schema name is assumed.
callback_schema	The schema of the user-defined callback package. If no value is provided, CURRENT_USER is used.
callback_package	The name of the user-defined callback package. The callback package isolates application developers from the internal details of the saga infrastructure, and enables the developers to focus on business logic.
dblink_to_participant	The pre-created database link to connect from broker PDB to the participant PDB.
queue_partitions	The number of partitions for the Saga entity IN/OUT queues. This allows for higher parallelism and throughput for the Saga entity. DEFAULT is 1. **Note:** The number of queue partitions should be equal to the partition count specified while performing the add_broker() operation.
version	Saga version

Usage Notes

The `ADD_PARTICIPANT` procedure needs coordinator name as an argument, hence the `ADD_COORDINATOR` procedure must be used before the `ADD_PARTICIPANT` procedure.

DROP_PARTICIPANT Procedure

This procedure drops the given participant on the local PDB and the broker.

The `DROP_PARTICIPANT` procedure drops the given participant on the local PDB and the broker if there are no pending sagas and pending saga messages for the participant. The `DROP_PARTICIPANT` procedure also disables message propagation between the participant and the broker.

Syntax

```
drop_participant(participant_name  IN varchar2)
```

Parameters

Table 10-7 DROP_PARTICIPANT Procedure Parameters

Parameter	Description
`participant_name`	The name of the participant microservice that is being dropped.

REGISTER_SAGA_CALLBACK Procedure

This procedure enables users to create or modify a callback package if the package is created using ADD_PARTICIPANT procedure.

If the callback package was not added before, the REGISTER_SAGA_CALLBACK procedure sets up the required subscribers and the notification callback. While doing an add participant the user has the option of specifying the callback package. If they want to defer specifying the callback package or if they want to change the callback package they can use the REGISTER_SAGA_CALLBACK procedure. If the callback already exists for a participant, then it is simply replaced. The REGISTER_SAGA_CALLBACK procedure is useful in cases where the user is not sure about whether to implement PL/SQL or Java, and wants to decide about it later, or the user wants to switch implementations (for example, from a Java client to a PL/SQL based client).

Syntax

```
register_saga_callback(participant_name IN varchar2,
 callback_schema        IN varchar2 default sys_context('USERENV','CURRENT_USER'),
 callback_package       IN varchar2);
```

Parameters

Table 10-8 REGISTER_SAGA_CALLBACK Procedure Parameters

Parameter	Description
`participant_name`	The name of the participant already created.

Table 10-8 (Cont.) REGISTER_SAGA_CALLBACK Procedure Parameters

Parameter	Description
`callback_schema`	The schema for the user-defined callback package. If no value is provided, `CURRENT_USER` is used.
`callback_package`	The name of the user-defined callback package. The callback package isolates application developers from the internal details of the saga infrastructure, and enables them to focus on business logic.