Quest

SharePlex® 11.4

Administrator Guide

SharePlexAdministrator Guide
Updated - 5/8/2024
Version - 11.4

Contents

About this Guide

This manual provides instructions for:

- Operating SharePlex
- Planning your replication strategy
- Preparing the environment for replication
- Configuring replication
- Starting replication
- Monitoring, tuning, and troubleshooting replication
- Failover/failback in a high-availability environment
- Performing administrative operations on replication systems

Other SharePlex Documentation

For the complete SharePlex documentation set, go to https://support.quest.com/shareplex/technical-documents.

Conventions Used in this Guide

Conventions used in this manual

The following typographic conventions are used in this guide:

- **Bold** represents required components of a command or option that must be typed as shown.
- *Italics* represent variables defined, named or entered by the user.
- {Braces} enclose required arguments.
- [Brackets] represent optional command components and may also be used in example command strings to emphasize required user defined variables in long strings.

  Example:

  reconcile queue {*queuename*} for {*datasource-datadest*} [**on** *host*]

- A vertical bar, or “pipe” character, (|) within brackets or braces indicates that you can use only one of the enclosed components.

  Example:

  abort service {*service* | **all**}

Names of commands, programs, directories and files are expressed in **Bold**.

Other names are expressed in capital letters using the default font.

Examples:

The **sp_ctrl** program is located in the **bin** directory.

Open the **oramsglst** file.

Find the value for ORACLE_HOME.

Click **Apply**.

System displays, such as prompts and command output, are expressed in a `monofaced` (fixed-space) font.

Examples:

```
sp_ctrl(sysA)>
```

```
User is a viewer (level=3)
```

Windows menu items, dialog boxes, and options within dialog boxes are expressed in **Bold**.

Example:

From the **File** menu, select **Print**.

System names are expressed generically or fictitiously. When necessary, the source system (or primary system) is referred to as *SysA*. Target systems (or secondary systems) are referred to as *SysB*, *SysC*, *SysD*, and so forth.

Revision History

Document Version	Date	Change History
2	19th April 2024	Added the Activate replication with PostgreSQL hot backup on an active database section.
3	8th May 2024	Added the Initial Database Synchronization for PostgreSQL to PostgreSQL Replication section.

About us

We are More than Just a Name

We are on a quest to make your information technology work harder for you. That is why we build community-driven software solutions that help you spend less time on IT administration and more time on business innovation. We help you modernize your data center, get you to the cloud quicker and provide the expertise, security and accessibility you need to grow your data-driven business. Combined with Quest's invitation to the global community to be a part of its innovation, and our firm commitment to ensuring customer satisfaction, we continue to deliver solutions that have a real impact on our customers today and leave a legacy we are proud of. We are challenging the status quo by transforming into a new software company. And as your partner, we work tirelessly to make sure your information technology is designed for you and by you. This is our mission, and we are in this together. Welcome to a new Quest. You are invited to Join the Innovation™.

Our Brand, our Vision. Together.

Our logo reflects our story: innovation, community and support. An important part of this story begins with the letter Q. It is a perfect circle, representing our commitment to technological precision and strength. The space in the Q itself symbolizes our need to add the missing piece — you — to the community, to the new Quest.

Contacting Quest

For sales or other inquiries, visit www.quest.com/contact.

Technical Support Resources

Technical support is available to Quest customers with a valid maintenance contract and customers who have trial versions. You can access the Quest Support Portal at https://support.quest.com.

The Support Portal provides self-help tools you can use to solve problems quickly and independently, 24 hours a day, 365 days a year. The Support Portal enables you to:

- Submit and manage a Service Request
- View Knowledge Base articles
- Sign up for product notifications
- Download software and technical documentation
- View how-to-videos
- Engage in community discussions
- Chat with support engineers online
- View services to assist you with your product

1

Overview of SharePlex

SharePlex provides high-speed replication that supports a variety of topology configurations in heterogeneous database environments. This chapter provides an overview of how SharePlex replication works. It explains concepts surrounding SharePlex replication and provides an overview of SharePlex capabilities.

For more information about the platforms and databases that SharePlex supports, see the SharePlex Release Notes.

Contents

The Advantages of SharePlex

SharePlex provides high-speed replication from Oracle and PostgreSQL (on-premises and cloud) databases to different target databases and messaging containers on major Unix and Linux platforms, both on-premises and in the cloud. SharePlex supports a wide variety of configurations to meet different and complex data availability needs. What's more, SharePlex includes — without extra charge — the compare and repair tools that you need to verify that replication is accurate and reliable.

Meet today's high-demand data availability requirements

SharePlex is designed for the nonstop replication of enterprise volumes of data. It is capable of replicating millions of transactions a day for thousands of tables and other objects. It supports business varieties of data, including large object types and National Language Character Set types, as well as Oracle XML and user-defined types.

You have full control over which data is replicated, and to where. Through column partitioning, you can replicate a subset of the columns of a table beyond a firewall, while protecting other, more sensitive data. Through row partitioning, you can replicate different records to different locations, or prevent the replication of certain records altogether. You can configure SharePlex to interact with PL/SQL procedures that transform data before, or instead of, posting it to a target database.

With SharePlex, your enterprise can ensure high availability, migrate data from one platform to another, and integrate data among many different datastores at once — whether locally, remote, or in the cloud. SharePlex not only supports standard query-driven replication targets, such as those for reporting, analytics and data warehousing, but it can also deliver the data to messaging systems and provide data in file or XML format for input to other enterprise solutions.

Support for a variety of replication sources and targets

SharePlex supports capture from and replication to many of today's popular datastores:

- Capture from **Oracle** (including Exadata) databases, replicating to Oracle target databases, including those hosted by Amazon, Microsoft, and Oracle Cloud and PaaS cloud environments.
- Replication from Oracle sources to many popular ODBC-compliant databases, such as Microsoft **SQL Server**, SAP **HANA**, other **PostgreSQL** implementations, and Oracle **MySQL**. SharePlex supports replication to several of these databases in Amazon EC2 and RDS cloud services, Google Cloud SQL for PostgreSQL, and Microsoft Azure Marketplace.
- Oracle databases to targets other than relational database systems, such as flat files (SQL and XML format), JMS, and Apache Kafka (XML and JSON).
- Oracle to an Oracle change-history target, where each change to the source data is replicated as a new row in the target, leaving the previous state of the target intact and providing a history of every change that was made to the source data.

SharePlex replicates to many different targets at the same time, requiring only one configuration file to provide the routing instructions for them all.

Deploy quickly and easily, without frameworks or add-ons

Everything required for data replication is provided "out of the box" with SharePlex, without the need to buy any add-ons or management packs. This includes the SharePlex Manager monitoring GUI software, and a compare-repair utility for detecting and repairing out-of-sync data.

The installation of SharePlex is fast and straightforward, and it includes utilities that help you configure connections to a database. Complex replication scenarios, such as active-active or a multi-step cascading scenario, may require more time, but overall SharePlex is driven primarily from a single configuration file on each source system. This file supplies most of the needed replication instructions: table lists, special handling such as column mapping or partitioning, and data routing. A relatively small set of commands and files supplies the rest of the input for setup and control.

SharePlex makes it easy to synchronize the data and start replication. In the case of Oracle data, you can even allow transactions on the source data to continue while you copy the source data and populate the target. SharePlex keeps track of the ongoing changes during the copy and then reconciles those changes with the results of the copy, so that it only applies transactions that occurred after the copy. Database patches and upgrades can be accomplished with a similar technique.

Although SharePlex is a reliable, relatively low-maintenance solution, our top-rated support team is ready around the clock to help with any trouble you may have. To get you started with your deployment, our professional services team is highly experienced and readily available.

One all-inclusive solution for both replication and repair

When you have SharePlex, you have both replication and data compare-repair software all in the same package. You pay no extra. You can run the SharePlex Compare and Repair features on a regular basis to ensure the consistency of source and target data. Run Compare to detect hidden out-of-sync conditions, and run Repair to repair the target rows to restore synchronization. SharePlex detects extra or missing rows and rows where the values do not match. By repairing mismatches at the row level on a regular basis, you can avoid larger problems that may require full data resynchronization. You can customize your comparisons, for example to filter the rows that are compared. These features work without stopping user activity or replication processing.

Maintain an Oracle high availability environment

In an Oracle environment, SharePlex supports reliable high availability configurations where replication maintains a duplicate database in a different location that is ready for fast, seamless failover and failback in planned or unplanned mode. If the primary system fails, transaction activity moves to the secondary system and continues while the secondary instance is copied to the primary system during recovery. SharePlex reconciles the copy with the replicated transactions from the secondary system, then discards operations that were already applied by the copy. After SharePlex restores synchronization of the data, transaction activity can move back to the primary system.

SharePlex also supports reliable replication recovery in deployments where the source and target are mirrored, such as with disk mirroring or Oracle Data Guard. SharePlex quickly recovers replication whether the source fails, the target fails, or both fail.

Conserve system resources

SharePlex performs replication without significantly impacting the source database, the source system, or the network. SharePlex reads the Oracle redo logs changes as they occur, rather than on a refresh schedule, this reduces the impact of replication on the network and does not cause spikes in network performance. This design also minimizes latency between source and target systems. Removing non-transactional data use from the production server improves the performance of the production database while enabling target databases to be optimized for the needs of their users.

Replicate with both speed and accuracy

SharePlex is fast, minimizing the latency between source and target databases by capturing changes to configured objects continuously. SharePlex maintains read consistency, maintaining operation order and session context all the way to the target. SharePlex uses standard SQL to apply replicated changes to the target database.

SharePlex continuously reads the transaction stream and sends the appropriate data to the target as quickly as possible, even before it receives a commit record. In the case of Oracle, if a transaction is canceled, SharePlex simply replicates the rollback so that the target remains an accurate representation of the source.

SharePlex provides tools to help you maximize replication throughput. Named queues enable you to split large transaction volumes into parallel processing streams. Hash partitioning enables you to split the rows of large tables across parallel Post processes.

Maintain fault tolerance and control

SharePlex tolerates outages regardless of where they occur. If the target system is down, or if there are network problems, SharePlex stores the data on the source system until operations and connections are restored. If the target system is running but the target database or receiving software itself is down, SharePlex queues the captured data on the target system until the target is available again.

You have control over when SharePlex sends the data to the target. By default, SharePlex sends a steady stream of data to the target systems, but you can delay transmission by stopping the Export process. You can delay the posting of data to a target by stopping or delaying the Post process.

Reduce downtime and risk from migrations

Hardware migrations usually require a significant amount of downtime, whether you need to change hardware platforms, move a data center, or consolidate servers to reduce costs. By maintaining a near-realtime copy of the database, SharePlex can help you minimize migration downtime by enabling the original system to function normally until the migration is complete.

About Source and Target Data

SharePlex replication uses the concepts of *source* and *target*.

- The source data is the primary data that is to be replicated. This data resides on the *source system*.
- The target data is a full or subset copy of the primary data. This data resides on the *target system*.

The object of replication is to keep the source and target data synchronized, or *in-sync*, which means that the state of the source data is reflected accurately by the target data, adjusting for any transformation that is performed and for any time lag in the replication stream.

The target data can take the form of any of the SharePlex-supported target types: tables in a database, messages in a messaging queue or topic, or XML or SQL records in a file that can be consumed by other software programs.

About the SharePlex Architecture

This topic explains the default configuration of SharePlex. You can customize the SharePlex configuration to add additional queues and processes for the purpose of isolating data streams or improving performance.

SharePlex directories

SharePlex uses two main directories:

The product directory: This is the SharePlex installation directory, where the SharePlex programs and libraries are stored.

The variable-data directory: This is the SharePlex working directory, where the queue files, log files and other components that comprise the current replication environment are stored.

> **NOTE**: These directories are often referred to as *productdir* and *vardir*, respectively.

Do not remove, rename or edit any files or directories installed by SharePlex. Some directories contain hidden files that are essential for replication. Some files appear empty but must exist under their original names because they are referenced by one or more SharePlex processes. Some items in the directories are for use only under the supervision of Quest Technical Support.

Programs meant for general use in a production environment are documented in the published SharePlex documentation. If you do not find documentation for a program in a SharePlex directory, do not attempt to run it. Contact Quest Technical Support first.

Files and directories can vary from version to version of SharePlex, but the basic structure appears as follows.

SharePlex product directory

Sub-directory	Contents
BACKUP	Uninstall information
bin	SharePlex executable files
config	Internally used content.

Sub-directory	Contents
data	Default parameter settings
doc	Catalog of exception messages
install	(Unix and Linux only) Scripts related to installation, licensing and upgrades
lib	SharePlex shared libraries
log	SharePlex log files
mks_oe	Runtime installation files for third-party software used by SharePlex.
util	SharePlex utilities
.app-modules	(Unix and Linux only) Hidden internal directory that contains raw executables. Do not use the contents of this directory to launch processes.
.meta-inf	(Unix and Linux only) Hidden internal directory that contains meta information used during the installation process.

SharePlex variable-data directory

Sub-directory	Contents
config	Configuration files for this installation of SharePlex.
data	Status Database, configuration activation information, user-defined parameter settings, and other user-defined files that direct replication activities.
db	Configuration internal database for each activation of a configuration file.
downgrd	Information about SharePlex targets that are a lower version than the source.
dump	Core files (if a process fails)
log	SharePlex log files
rim	Queue files (working data files)
save	Information about active and inactive configurations.
state	Information about the current state of SharePlex when a configuration is active, such as the object and sequence caches.
temp	Used by the copy and append features and other SharePlex sync-related processes.
oos	Stores the transactions that contain out-of-sync operations when the SP_OPO_SAVE_OOS_ TRANSACTION parameter is enabled.

The sp_cop process

The **sp_cop** program coordinates the SharePlex replication processes: (Capture, Read, Export, Import, Post) and the SharePlex queues, and it initiates all of the other background processes that perform specific tasks. It also maintains communication with other systems in the replication network. In general, most SharePlex users have little interaction with **sp_cop** other than to start and stop it. Once started, **sp_cop** runs in the background.

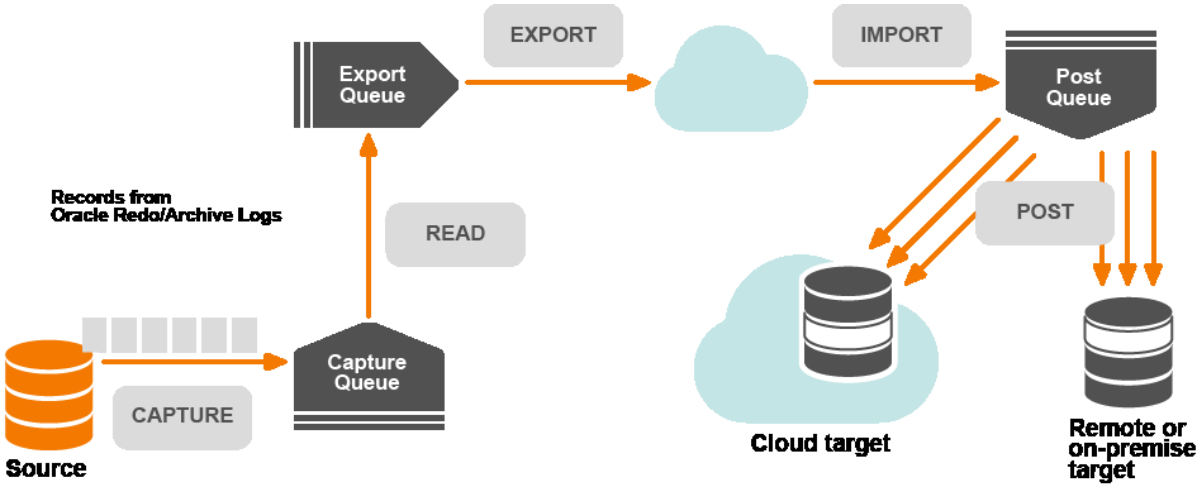
- Only a SharePlex Administrator (member of the SharePlex **admin** group) can start or stop **sp_cop**.
- **sp_cop** must be started on all source and target systems involved in replication.
- Start **sp_cop** as soon as (or before) applications access the data on the source system, so that all of the SharePlex processes are ready to start processing transactions. That way, Capture can keep pace with the changes that are made to the source data.

The sp_ctrl process

Use **sp_ctrl** to issue the commands that start, stop, configure, direct and monitor SharePlex activities. The **sp_ctrl** program interacts internally with the **sp_cnc** (command and control) process, which is the child process of **sp_cop** that executes the commands. Users do not interact with **sp_cnc** itself.

SharePlex replication processes

SharePlex replicates data through a series of replication processes that are started by the main SharePlex process, **sp_cop**.



- **The Capture process**: The Capture process reads the transaction records on the source system for changes to objects that are configured for replication by SharePlex. The Capture process writes the data to the capture queue, where it accumulates until the Read process is ready for it. When data is being replicated from more than one datasource, there is a separate Capture process for each one, each functioning concurrently and independently. The Capture process is named **sp_ocap** (Oracle Capture).

- **The Read process**: The Read process operates on the source system to read data from the capture queue and add routing information to it. After processing the data, the Read process sends it to the Export queue. The Read process is named **sp_ordr**.
- **The Export process**: The Export process operates on the source system to read data from the export queue and send it across the network to a target system. By default, there is one Export process for each target system. For example, if there are two target systems, there are two Export processes. The Export process is the first part of the Export/Import transport pair, which moves data between systems over a TCP/IP network. The Export process is named **sp_xport**.
- **The Import process**: The Import process is the second half of the Export/Import transport pair. The Import process operates on a target system to receive data and build a post queue. There is one Import process on a target system for each Export process that sends data to that target. For example, if there are two source systems (each with an Export process) replicating data to a single target system, there are two Import processes on that target. The Import process is named **sp_mport**.

  NOTE: It is possible to replicate data between databases on the same system. In this case the Export and Import processes are not created. The Read process places data directly into a post queue on that system.
- **The Post process**: The Post process operates on a target system to read the post queue and apply the replicated operations to the target database, file, message queue or topic. There is a Post process for each post queue on a target system. Multiple Post processes can operate simultaneously on a system. The Post process is **sp_opst_mt** (Oracle Post) or **sp_xpst** (Open Target Post).

All communication and movement of data by SharePlex is handled by an internal messaging and transport system, using an asynchronous stream protocol with TCP/IP connections that is very efficient for large data transfers. This method ensures optimal performance, reliability and restart capabilities, while conserving communication bandwidth. SharePlex can replicate over any TCP/IP network.

SharePlex queues

Queues store the replicated data as it is transported from the source system to the target system. Queues are part of a checkpoint recovery system that facilitates safe, asynchronous transport of data. Data travels through the queues in the sequence in which it was generated.

Data is not read-released (deleted) from one queue until it is written to the next one. Data accumulates in the queues on the source and target systems if the network, system, or database slows down or fails, or when a replication process stops. When the problem or outage is resolved, SharePlex resumes processing from the point where it stopped.

SharePlex replication uses the following queues:

- *Capture Queue*: The capture queue resides on the source system and stores captured data for further processing by SharePlex. There is one capture queue for each datasource that is being replicated. A capture queue is identified by the datasource, for example **o.fin1**.
- *Export Queue*: The export queue resides on the source system. It holds data that has been processed by SharePlex and is ready for transport to the target system. By default, there is one export queue on a source system regardless of the number of active configurations or target systems. A default export queue is identified by the name of the source system on which it resides, for example, **SysA**. You can instruct SharePlex to create additional *namedexport queues* for more complex replication strategies.

- *Post Queue*: The post queue resides on the target system. It holds data that is ready for Post to write to the target database, file, or message queue or topic. On each target system, there is one post queue for the replication stream between a datasource and its target. For example, if DatabaseA and DatabaseB are both replicating to DatabaseC, there are two post queues. A default post queue is identified by the name of the source system plus the datasource and the target, for example **SysA (o.DatabaseA-o.DatabaseB)**. You can instruct SharePlex to create additional named post queues for more complex replication strategies.

NOTE: All SharePlex queue files are created and maintained in the **rim** sub-directory of the SharePlex variable-data directory.

SharePlex installed objects

Much of the replication process is controlled and tracked through a series of internal objects that are installed into the source or target database during the installation of SharePlex. They are essential for SharePlex to operate, so do not alter them in any way.

NOTE: Not all objects are used for all databases. Most are used for Oracle databases. If you do not see an object in your database, it is not relevant to the database, or the information is stored internally within the SharePlex configuration. If you see an object that is in your database but not in this list, it is not being used in the current release.

Table	Object type	Description
DEMO_SRC	Table	Used as the source table for the SharePlex demonstrations.
DEMO_DEST	Table	Used as the target table for the SharePlex demonstrations.
SHAREPLEX_ACTID	Table	Used by Capture to checkpoint its state.
SHAREPLEX_ANALYZE	Table	Used by the **analyze** command.
SHAREPLEX_CHANGE_ OBJECT	Table	Used by users to stop and resume replication for an object.
SHAREPLEX_COMMAND	Table	Used for the **flush**, **abort** and **purge** commands.
SHAREPLEX_CONFIG	Table	Used by the activation and Capture processes to mark the start of a new activation.
SHAREPLEX_DATA	Table	Used by the SharePlex wallet for Oracle TDE replication.
SHAREPLEX_ DATAEQUATOR	Table	Used by the **compare** and **repair** commands and the Post process to synchronize their operations.
SHAREPLEX_ DATAEQUATOR_ INSERT_TEMP	Table	Used as a temporary table by the **compare** and **repair** commands.
SHAREPLEX_ DATAEQUATOR_ UPDATE_TEMP	Table	Used as a temporary table by the **compare** and **repair** commands.
SHAREPLEX_	Table	Used as a temporary table by the **compare** and **repair** commands.

Table	Object type	Description
DATAEQUATOR_ DELETE_TEMP		
SHAREPLEX_DDL_ CONTROL	Table	Used to refine control of DDL that is enabled for replication by the SP_OCT_REPLICATE_ALL_DDL parameter.
SHAREPLEX_JOBID	Sequence	Used by the **sp_cnc** process and the **compare**, **repair**, and **copy** commands to provide a unique job ID.
SHAREPLEX_JOBS	Table	Used by the **sp_cnc** process and the **compare**, **repair**, and **copy** commands to store information about a job.
SHAREPLEX_JOB_ STATS	Table	Used by the **sp_cnc** process and the **compare**, **repair**, and **copy** commands to store information about a job.
SHAREPLEX_JOBS_ CONFIG	Table	Used by the **disable jobs** and **enable jobs** commands.
SHAREPLEX_LOB_ CACHE	Table	Used by the Capture process when processing VARRAYs stored as LOB.
SHAREPLEX_LOBMAP	Table	Used by the Capture process to map LOBIDs and rows when a table with LOB columns does not have PK/UK logging enabled.
SHAREPLEX_LOGLIST	Table	Used by the Capture process to track inactive RAC instances.
SHAREPLEX_MARKER	Table	Used by the Read process when PK/UK logging is not enabled.
SHAREPLEX_OBJMAP	Table	Used by the activation and Capture processes to define the objects in replication.
SHAREPLEX_ PARTITION_CACHE	Table	Used by the Capture process to map Oracle partition IDs to tables in replication.
SHAREPLEX_SYNC_ MARKER	Table	Used by the **copy** command and the Read and Post processes to sync their operations.
SHAREPLEX_TRANS or SHAREPLEX_OPEN_ TRANS	Table	Used by the Post process to store checkpoints and to mark transactions that were applied in a primary-to-primary configuration.

How SharePlex Replication Works

To replicate data, SharePlex reads the stream of transaction data on the source system and captures changes that are made to objects that are specified in a configuration file. In the configuration file, you specify which data to replicate and the target to which it is applied.

You activate a configuration file to start replication. This is done by means of the **activate config** command in **sp_ctrl** within a sequence of steps that also includes synchronizing the source and target data for the first time. When a configuration is active, SharePlex replicates only the changes that are made to the objects specified in the configuration file, not entire data records, which provides a fast and reliable replication solution.

For more information see:

- Configure SharePlex to Replicate Data on page 60
- Start Replication on your Production Systems on page 254

From the information that it has about a transaction operation, SharePlex creates one or more messages that are sent from the source system to the target system. A message can reflect a SQL operation or an internal SharePlex operation, but most of the time it is an INSERT, UPDATE, DELETE, COMMIT, TRUNCATE or a supported DDL operation.

NOTE: Large operations like those on LONG or LOB columns can require more than one message because a message has a size limitation. Other operations, such as array inserts of small records, have the inverse effect: There could be one record for numerous operations. For example, an array insert of 70,000 rows might be recorded in the transaction stream as only 700 messages, depending on the data. In general, unless you are replicating numerous changes to those kinds of data types, you can assume that the number of messages shown in the status output for a process or queue approximately corresponds to the same number of SQL operations.

The Post process reads messages from the post queue and applies the replicated data changes to the target. In the case of a database target, Post constructs SQL statements to apply the data. In the case of non-database targets, Post outputs data records in the format required by the target, for example a file or messaging queue or topic.

The following explains the default ways that SharePlex builds SQL statements on the target system:

- If the change is an INSERT, SharePlex uses all of the columns in the row to build an INSERT statement.
- If the change is a DELETE, SharePlex uses only the key to build a WHERE clause to locate the correct row. In the case of Oracle, if a table lacks a key, SharePlex simulates one by using the values of all of the columns, except LONG and LOB columns. You can specify columns to use as a key when you create the configuration file. In the case of SQL Server, all configured objects must have a primary key.
- If the change is an UPDATE, SharePlex uses the key plus the values of the changed columns to build a WHERE clause to locate the correct row. Before applying changes to the database, the Post process compares a pre-image of the values of the source columns to the existing values of the target columns. The pre-image (also known as the *before image*) is the value of each changed column before the UPDATE. If the pre-image and the existing target values match, confirming a synchronized state, Post applies the changes. If not, then Post logs the operation to an error file and SharePlex returns an “out-of-sync” error.
- If the change is an UPDATE or DELETE statement that affects multiple rows on the source machine, SharePlex issues multiple statements on the target to complete the task. For example, an **UPDATE tableA set name = ‘Lisa’ where rownum < 101** statement actually sends 100 UPDATE statements to the target, even though only one statement was issued on the source.

Understand the Concept of Synchronization

The concept of synchronization applies mainly to table-to-table replication, where Post performs integrity checks to make certain that only one row in the target matches the row change that is being replicated. It does not apply to file, messaging targets, and change-history targets, which contain a record of every operation replicated by Post, some of which may be identical over time. The Post process does not perform integrity checks on those targets.

Characteristics of synchronized tables

The basic characteristics of synchronized source and target tables are as follows (unless the transformation feature is used).

- If a row exists in the source database, it exists in the target.
- Corresponding columns in source and target tables have the same structure and data types.
- Data values in corresponding rows are identical, including the values of the key.

Ensuring data integrity is the responsibility of the Post process. Post applies a WHERE clause to compare the key values and the before values of the SQL operations that it processes. Post uses the following logic to validate synchronization between source and target tables:

- Post applies a replicated INSERT but a row with the same key already exists in the target. Post applies the following logic:
    - If all of the current values in the target row are the same as the INSERT values, Post considers the rows to be in-sync and discards the operation.
    - If any of the values are different from those of the INSERT, Post considers this an out-of-sync condition.

    NOTE: You can configure Post so that it does not consider non-key values when posting an INSERT (applicable only when replicating data from Oracle to Oracle). See the `SP_OPO_SUPPRESSED_OOS` parameter in the SharePlex Reference Guide.

- Post applies a replicated UPDATE but either cannot find a row in the target with the same key value as the one in the UPDATE or Post finds the correct row but the row values do not match the before values in the UPDATE. Post applies the following logic:
    - If the current values in the target row match the after values of the UPDATE, Post considers the rows to be in-sync and discards the operation.
    - If the values in the target row do not match the before or after values of the UPDATE, Post considers this an out-of-sync condition.

    NOTE: You can configure Post so that it returns an out-of-sync message if the current values in the target row match the after values of the UPDATE (applicable only when replicating data from Oracle to Oracle). See the `SP_OPO_SUPPRESSED_OOS` parameter in the SharePlex Reference Guide.

- A DELETE is performed on the source data, but Post cannot locate the target row by using the key. When Post constructs its DELETE statement, it includes only the key value in its WHERE clause. If the row does not exist in the target, Post discards the operation.

Hidden out-of-sync conditions

Post only verifies the integrity of the rows that are being changed by its current SQL operation. It does not verify whether other rows in that table, or in other tables, are out of synchronization in the target database. A hidden out-of-sync condition may not show up until much later, when a change to the affected row is eventually replicated by SharePlex or a discrepancy is detected in the course of using that data.

Example of a detectable out-of-sync condition

Someone logs into the target and updates the COLOR column in the *target* table from “blue” to “red” in Row1. Then, an application user on the *source* system makes the same change to the source table, and SharePlex replicates it to the target. In the WHERE clause used by Post, the pre-image for the target table is “blue,” but the current value in the target row is “red.” Post generates an out-of-sync error alerting you to the out-of-sync condition.

Example of a hidden out-of-sync condition

Someone logs into the target and updates the COLOR column in the target table from “blue” to “red” in Row2, but the change is not made to the source table and is not replicated. The two tables are now out-of-sync, but Post does not return an error message, because there is no replication performed on that row. No matter how many subsequent updates are made to other columns in the row (SIZE, WEIGHT), the hidden out-of-sync condition for the COLOR column persists (and users on the target have inaccurate information) until someone updates the COLOR column in the source table. When that change is replicated, only then does Post compare the pre-images and return an error message.

The majority of time, the cause of out-of-sync data is not anything done wrong by replication, but rather DML applied on the target, an incomplete backup restore, or some other hidden out-of-sync condition, which goes undetected until replication affects the row. Solving out-of-sync conditions can be time-consuming and disruptive to user activity. Once replication is started, it is recommended that you:

- Prevent write access to the target tables, so that DML and DDL cannot be applied to them.
- Use the **compare** command to compare source and target data regularly to verify synchronization and detect hidden out-of-sync conditions. You can use the **repair** command to repair any out-of-sync rows. For more information about these commands, see the SharePlex Reference Guide.

How SharePlex responds to an out-of-sync condition

You can decide how you want SharePlex to respond to transactions that generate an out-of-sync error:

- The default Post behavior when a transaction contains an out-of-sync operation is to continue processing other valid operations in the transaction to minimize latency and keep targets as current as possible. Latency is the amount of time between when a source transaction occurs and when it is applied to the target. Different factors affect the amount of latency in replication, such as unusually high transaction volumes or interruptions to network traffic.

  Post logs the SQL statement and data for the out-of-sync operation to the `ID_errlog.sql` log file, where *ID* is the database identifier. This file is in the **log** sub-directory of the variable-data directory on the target system.

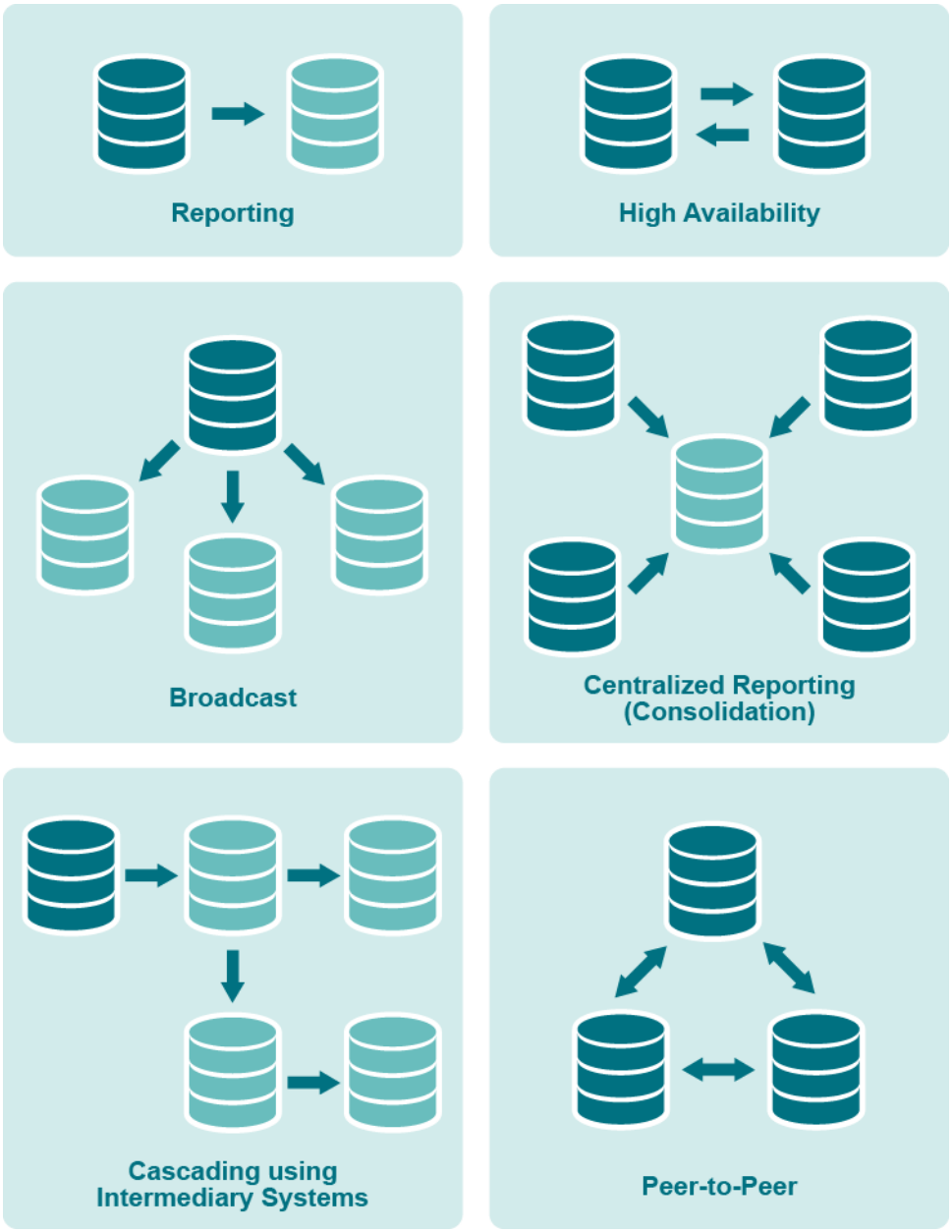
- You can configure Post to stop when it encounters an out-of-sync condition by setting the following parameter to 1:
  - **Oracle targets:** `SP_OPO_OUT_OF_SYNC_SUSPEND`
  - **Open Target targets:** `SP_OPX_OUT_OF_SYNC_SUSPEND`
- You can configure Post to roll back and discard a transaction if any operation in that transaction generates an out-of-sync error. The entire transaction is logged to a SQL file, but not applied to the target.You can edit the SQL file to fix the invalid DML and then run the SQL file to apply the transaction. This feature is enabled by setting the `SP_OPO_SAVE_OOS_TRANSACTION` to 1.

Strategies for Information Availability

With SharePlex, you can put a replica database to work as a reliable, continuously updated alternate database that can be used in many different ways. The following strategies enable you to get the right data to the people who need it, when they need it.

NOTE: Support for these topologies may vary depending on the type of database involved.

Figure 1: SharePlex replication strategies at a glance



Reporting instances

Targets maintained by SharePlex are ideal for offloading report and query processing because they are accessible while being kept up-to-date, and they can be optimized with keys and indexes designed for optimal query performance. You can run reports all day long, without complaints about performance from your OLTP users. Even during busy reporting times such as the end of the month or quarter, application response time will be unaffected by heavy reporting. And, your organization's decision-makers will appreciate the accuracy of the data reflected in the reports. For more information, see Configure Replication to Share or Distribute Data on page 155.

Broadcast and cascade

When many remote users access or use data stored in a primary database, you can move their processing to one or more secondary databases that are kept current through SharePlex replication. That way, you can keep the primary database and system optimized for transactions. SharePlex also can cascade data through an intermediary system to remote systems, providing access for remote users who have no direct network connection to the primary database. For more information, see Configure Replication through an Intermediary System on page 207

Data warehousing

SharePlex can replicate from numerous source systems to one target system. This configuration is ideal for consolidating data in a data warehouse or a data mart so that information is available enterprise-wide for queries and reports. You have control over the data that is replicated and the option to transform any data to conform to a different target structure. These capabilities enable you to populate your data warehouse with the specific, timely information that users need to make good decisions. For more information, see Configure Replication to Maintain a Central Datastore on page 160.

High availability and disaster recovery

SharePlex can be used to maintain duplicate databases over local or wide-area networks. Production can move to the alternate sites in an emergency or in a planned manner when routine maintenance is performed on the primary server. SharePlex replication enables the secondary database to be used for queries and reporting. For more information, see Configure Replication to Maintain High Availability on page 212.

Peer-to peer

SharePlex supports replication among multiple source databases where applications on each system can make changes to the same data while SharePlex maintains synchronization. In this strategy, the databases are usually mirror images of each other, with all objects existing in their entirety on all systems. Although similar in benefit to a high-availability strategy, the difference between the two is that peer-to-peer allows concurrent changes to the same data, while high availability permits changes to the secondary database only in the event that the primary database goes offline. A few ways to use peer-to-peer replication are to maintain the availability and flexibility of a database by enabling access from different locations or to distribute heavy online transaction processing volumes among multiple access points. For more information, see Configure Peer-to-Peer Replication on page 165.

Test before you deploy

Before you implement SharePlex on production systems, make certain to perform tests in a mirror of the production environment to ensure that you configured SharePlex properly to support your requirements. Testing can uncover issues such as configuration errors and unexpected environmental issues, for example network or resource issues that affect replication performance or availability.

Additionally, it is assumed that your organization has in place an infrastructure that supports the use of enterprise applications such as SharePlex. These include, but are not limited to, the following:

- Availability and use of the database and SharePlex documentation
- Training programs for users

- Rollout and upgrade plans that ensure minimal interruption to business. When SharePlex is implemented as part of an application's infrastructure, it is strongly recommended to test new application functionality in conjunction with SharePlex in a non-production environment.
- Database or system maintenance procedures that consider SharePlex dependencies, such as the proper shutdown of SharePlex processes and the preservation of unprocessed transaction records and replication queues to accommodate system or database maintenance.
- Sufficient security that prevents unauthorized persons from accessing SharePlex data records or making configuration changes.

The SharePlex Professional Services team can help you prepare for, install, and deploy SharePlex in your environment.

2

Run SharePlex

This chapter contains instructions for running SharePlex on UNIX and Linux.

Contents

Run SharePlex on Unix and Linux

On Unix and Linux systems, you start SharePlex by running the **sp_cop** program. After you activate a configuration, **sp_cop** spawns the necessary child replication processes on the same system. Each instance of **sp_cop** that you start is a parent to its own set of child replication processes. The **sp_cop** process must be started on each system that is part of the replication configuration.

You can start sp_cop in one of two ways:

- From the operating system command line.
- At system startup as part of the startup file.

IMPORTANT: Run SharePlex from either the korn (**ksh**) or C shell (**csh**) shell.

- Do not use the Bourne shell (sh), because the way it handles background processes is not compatible with SharePlex. If you must use the Bourne shell, switch shells to **ksh** or **csh** to run SharePlex, then exit the shell and return to the Bourne shell.
- If using an Exceed X window emulator, switch from the default shell of POSIX to the **ksh** shell, then run **sp_cop** from the **ksh** shell only.

Startup sequence on Unix and Linux

When you start systems that are involved in replication, start the components in this order:

1. Start the system.
2. Start the source and target databases.
3. Start SharePlex.
4. Start **sp_ctrl**.
5. Verify that the SharePlex processes are started by issuing the **lstatus** command in **sp_ctrl**.

   ```
   sp_ctrl> lstatus
   ```

6. Allow users on the system.

Start SharePlex on Unix and Linux

To start SharePlex, you must log onto the system as a SharePlex Administrator. Your user name must be assigned to the SharePlex **admin** group in the **/etc/group** file. For more information, see Assign SharePlex Users to Security Groups on page 251.

Table 1: SharePlex startup syntax

Method	Startup syntax
From root, with full path	$ /*productdir*/**bin**/**sp_cop** [**-u***identifier*] **&**
CD to the product directory	$ **cd** /*productdir*/**bin** $**./sp_cop** [**-u***identifier*] **&**
From a startup script	**#!/bin/ksh** **cd** *productdir***bin** **nohup sp_cop** [**-u***identifier*] **&**

Table 2: Description of SharePlex startup syntax

Argument	Description
&	Causes SharePlex to run in the background.
nohup	Directs the startup of SharePlex to continue in the background after the current user logs out.
-u*identifier*	Starts **sp_cop** with a unique identifier. Use this option when there are multiple instances of **sp_cop** running on a system, which is required for some SharePlex configurations. For more information, see Run Multiple Instances of SharePlex on page 48. Some suggestions for *identifier* are: • the SharePlex port number (such as **-u2100**) • the identifier of the database for which replication is running (such as **-uora12c**) • any descriptive identifier (such as **-utest**)

Identify SharePlex processes on Unix and Linux

Every session of **sp_cop** has a process ID number. The ID is returned after startup and then the command prompt reappears. If a configuration was activated during a former session of **sp_cop**, replication begins immediately. Without an active configuration, **sp_cop** runs passively in the background.

On Unix and Linux systems, you can use the ps -ef | grep sp_ command to view the SharePlex processes that are running.

- The **sp_cop** process is the root process.
- **The following child processes are spawned by sp_cop on a source system:**
  - Command and Control process (**sp_cnc**)
  - Capture (**sp_ocap**)
  - Read (**sp_ordr**)
  - Export (**sp_xport**)
- **The following child processes are spawned by sp_cop on a target system:**
  1. Command and Control process (**sp_cnc**)
  2. Import (**sp_mport**)
  3. Post (**sp_opst_mt** if the database is Oracle or **sp_xpst** if the database is Open Target)

Each child process has the same **-u***identifier* as its parent **sp_cop** process. This makes it easier to identify related processes when multiple session of **sp_cop** are running.

Stop SharePlex on Unix and Linux

To stop SharePlex, issue the **shutdown** command in **sp_ctrl**. This is a graceful shutdown that saves the state of each process, performs a checkpoint to disk, read/releases buffered data, and removes child processes. Data in the queues remains safely in place, ready for processing when **sp_cop** starts again. The shutdown process can take some time if SharePlex is processing large operations.

You can use the **force** option with the **shutdown** command to forcefully shut down replication if necessary. It terminates **sp_cop** immediately, bypassing normal shutdown procedures. See the SharePlex Reference Guide for more information about this command.

Shutdown considerations on Unix and Linux

You can safely shut down SharePlex for a short time while there is still transactional activity. The next time you start SharePlex, replication resumes at the correct place in the redo logs or the archive logs, if needed. However, the best practice is to leave SharePlex running while there is transactional activity. Otherwise, SharePlex may need to process a large volume of redo backlog when you start it again, and there will be latency between the source and target data.

If the redo logs wrap and the archive logs cannot be accessed, resynchronization of the source and target data may be the only option. Take this possibility into account whenever you stop SharePlex while redo is still being generated.

NOTE: If you want to shut down both SharePlex and the database, shut down SharePlex first. Otherwise, SharePlex will interpret that the database is failing and generate a warning message.

As an alternative to stopping SharePlex, you can use the **stop** command in **sp_ctrl** to stop individual SharePlex replication processes as needed. See the SharePlex Reference Guide for more information about this command.

Run SharePlex on Linux for PostgreSQL

On Linux systems, start SharePlex by running the **sp_cop** program. After you activate a configuration, **sp_cop** spawns the necessary child replication processes on the same system. Each instance of **sp_cop** that you start is a parent to its own set of child replication processes. The **sp_cop** process must be started on each system that is part of the replication configuration.

You can start sp_cop in one of two ways:

- From the operating system command line.
- At system startup as part of the startup file.

IMPORTANT: Run SharePlex from either the korn (**ksh**) or C shell (**csh**) shell.

WARNING! Do not use the Bourne shell (sh), because the way it handles background processes is not compatible with SharePlex. If you must use the Bourne shell, switch shells to **ksh** or **csh** to run SharePlex, then exit the shell and return to the Bourne shell.

Start SharePlex on Linux

To start SharePlex, you must log onto the system as a SharePlex Administrator. Your user name must be assigned to the SharePlex **admin** group in the **/etc/group** file. For more information, see Assign SharePlex users to security groups in the SharePlex Administrator's guide.

Table 3: SharePlex startup syntax

Method	Startup syntax
From root, with full path	$ /*productdir*/**bin**/**sp_cop** [**-u***identifier*] **&**
CD to the product directory	$**cd** /*productdir*/**bin** $**./sp_cop** [**-u***identifier*] **&**
From a startup script	**#!/bin/ksh** **cd** *productdir***bin** **nohup sp_cop** [**-u***identifier*] **&**

Table 4: Description of SharePlex startup syntax

Argument	Description
&	Causes SharePlex to run in the background.
nohup	Directs the startup of SharePlex to continue in the background after the current user logs out.
-u*identifier*	Starts **sp_cop** with a unique identifier. Use this option when there are multiple instances of **sp_cop** running on a system, which is required for some SharePlex configurations. For more information, see Run multiple instances of SharePlex in the SharePlex Administrator Guide. **Some suggestions for *identifier* are:** • the SharePlex port number (such as **-u2100**) • the identifier of the database for which replication is running (such as **-pg13c**) • any descriptive identifier (such as **-utest**)

Identify SharePlex processes on Linux

Every session of **sp_cop** has a process ID number. The ID is returned after startup and then the command prompt reappears. If a configuration was activated during a former session of **sp_cop**, replication begins immediately. Without an active configuration, **sp_cop** runs passively in the background.

On Linux systems, you can use the **ps -ef | grep sp_** command to view the SharePlex processes that are running.

- The **sp_cop** process is the root process.
- **The following child processes are spawned by sp_cop on a source system:**
  - Command and Control process (**sp_cnc**)
  - Capture (**sp_ocap**)
  - Read (**sp_ordr**)
  - Export (**sp_xport**)
- **The following child processes are spawned by sp_cop on a target system:**
  - Command and Control process (**sp_cnc**)
  - Import (**sp_mport**)
  - Post (**sp_xpst**)

Each child process has the same **-u***identifier* as its parent **sp_cop** process. This makes it easier to identify related processes when multiple session of **sp_cop** are running.

Stop SharePlex on Linux

Before shutting down the SharePlex process user should deactivate the active configuration and check the following processes:

- Capture/read/export on source
- Import/post on target are not running

For more information on this, see the Deactivate Config in the SharePlex Reference Guide.

To stop SharePlex, issue the **shutdown** command in **sp_ctrl**. This is a graceful shutdown that saves the state of each process, performs a checkpoint to disk, read/releases buffered data, and removes child processes. Data in the queues remains safely in place, ready for processing when **sp_cop** starts again. The shutdown process can take some time if SharePlex is processing large operations.

You can use the **force** option with the **shutdown** command to forcefully shut down replication if necessary. It terminates **sp_cop** immediately, bypassing normal shutdown procedures. See the SharePlex Reference Guide for more information about this command.

Shutdown considerations on Linux

You can safely shut down SharePlex for a short time while there is still transactional activity. The next time you start SharePlex, replication resumes at the correct place in the WAL files, if needed. However, the best practice is to leave SharePlex running while there is transactional activity. Otherwise, SharePlex may need to process a large volume of WAL files backlog when you start it again, and there will be latency between the source and target data.

NOTE: If you want to shut down both SharePlex and the database, shut down SharePlex first. Otherwise, SharePlex will interpret that the database is failing and generate a warning message.

As an alternative to stopping SharePlex, you can use the **stop** command in **sp_ctrl** to stop individual SharePlex replication processes as needed. See the SharePlex Reference Guide for more information about this command.

3

Run Multiple Instances of SharePlex

This chapter shows you how to configure and run multiple instances of SharePlex on one machine. For example, when replication is configured from multiple source systems to a central target system, you can run multiple instances of **sp_cop** to isolate the replication streams.

Contents

Run Multiple Instances of SharePlex from Separate Installations

IMPORTANT! This topic assumes that there is no active configuration. It assumes you are configuring multiple instances of **sp_cop** as part of an initial setup of the replication environment.

This method provides a one-to-one relationship between the binaries and the variable-data directory. This procedure creates autonomous SharePlex instances, with nothing in common between them. You start, control and maintain each SharePlex instance separately, and there are no special setup requirements.

This method has the following benefits:

- Processes are easily isolated. You do not have to set environment variables to point to the correct port and variable-data directory.
- You can upgrade or perform other maintenance one product directory at a time, or choose not to perform those tasks.
- You can run the same or different versions of SharePlex on the same system.

The disadvantages are:

- You must install and upgrade each installation separately.
- More disk space is required to store the product files.
- In startup and shutdown scripts, and other places where you must map environment variables to SharePlex components, you must map them for each installation.

To set up multiple instances of SharePlex in this configuration:

- Install each one separately. There should be one product directory and one variable-data directory per installation.
- Install each one on a different TCP/IP port number.

IMPORTANT! Make certain to create a different database account for each installation.

To install SharePlex, see the SharePlex Installation Guide.

Run Multiple Instances of SharePlex from One Installation

IMPORTANT! This topic assumes that there is no active configuration. It assumes you are configuring multiple instances of **sp_cop** as part of an initial setup of the replication environment.

This method provides a one-to-many relationship between a set of binaries and two or more variable-data directories. In this configuration, you create multiple variable-data directories and link each one to a unique port number, each running a separate instance of **sp_cop**.

This method has the following benefits:

- You install and upgrade only one installation of SharePlex. Maintenance procedures are performed for only one installation.
- You conserve disk space, because you only store one set of SharePlex binaries and installed files.
- The customization of the SharePlex monitoring scripts only need to be done once, in one place. For more information, see Run Monitor Scripts on Unix or Linux on page 291.
- Startup and shutdown scripts only need to be created and run for one set of binaries.

The disadvantages are:

- Processes must be directed to each instance. You must set environment variables for each instance, start **sp_cop** with the correct identifier for each instance, and set a port connection in **sp_ctrl** to ensure that commands are directed to the correct instance.
- Upgrades apply to all instances of SharePlex.
- All **sp_cop** instances are the same version of SharePlex.

How to run multiple sp_cop instances on Unix and Linux

To run multiple instances SharePlex on the same Unix or Linux machine, you run multiple instances of the **sp_cop** program, each running on a different port number. You link each **sp_cop** to a different variable-data directory. Each variable-data directory is identified by the port number of its **sp_cop**. Connection information to the source or target datastore is linked to each **sp_cop** instance.

1. Assign port numbers

Assign each instance of **sp_cop** a unique port number.

- For each **sp_cop** instance, obtain one port number that will be used by that instance for both TCP and UDP communication.
- For each **sp_cop** instance, use the same port number for the remote instances of **sp_cop** that will process the same replication data stream as the instance you are configuring.

The **sp_cop** process uses the TCP port for communication between two different systems in the network, such as data exchange between the Export and Import processes. If the ports are different, **sp_cop** on one system cannot connect to the **sp_cop** on another system to send or receive messages.

2. Create variable-data directories

For each instance of **sp_cop**, create a variable-data directory, and assign each to one of the port numbers you obtained. The variable-data directory contains the environment that is unique to a SharePlex instance.

1. Install SharePlex according to the instructions in the SharePlex Installation Guide. At the end of the installation, you should have one product directory, one variable-data directory associated with a port number, and one database account. This is your base instance of SharePlex.
2. Log in as a root user.
3. Shut down **sp_cop** if it is running.
4. Copy the original variable-data directory (with its sub-directories) to a new variable-data directory for each instance of **sp_cop** that you want to run. Include the port number in each name, as shown in the following examples.

   cp -p -r /splex/vardir/splex2100 /splex/vardir/splex2101

   cp -p -r /splex/vardir/splex2100 /splex/vardir/splex2102

3. Define the port numbers in the SharePlex environment

For each variable-data directory that you created, perform this procedure to set the port number that you reserved for this instance of SharePlex.

1. Export the SP_SYS_VARDIR variable to point to one of the new variable-data directories, for example **splex2101** in the preceding example.

   ksh shell:

   export SP_SYS_VARDIR=*/full_path_of_variable-data_directory*

   csh shell:

   setenv SP_SYS_VARDIR=*/full_path_of_variable-data_directory*

2. Export the SP_COP_TPORT and SP_COP_UPORT variables to point to the port number of the variable-data directory that you exported.

   ksh shell:

   export SP_COP_TPORT=*port*

   export SP_COP_UPORT=*port*

   csh shell:

   setenv SP_COP_TPORT *port*

   setenv SP_COP_UPORT *port*

3. Log in as SharePlex Administrator.

4. Run the **clean_vardir.sh** script. The script removes duplicate replication queues and restores each one to a fresh state. See clean_vardir.sh in the SharePlex Reference Guide for more information.
5. In the **rim** sub-directory of the exported variable-data directory, delete the **shstinfo.ipc** and **shmaddr.loc** files. (These files may not exist if **sp_cop** has never been started for this variable-data directory.)
6. Repeat these steps for each additional variable-data directory.

4. Establish connections to the source or target datastore

For each **sp_cop** instance, establish the connections that SharePlex will use to access the source or target data of this SharePlex instance.

1. Export the SP_SYS_VARDIR variable to point to one of the new variable-data directories, for example **splex2101** in the example.

   ksh shell:

   export SP_SYS_VARDIR=*/full_path_of_variable-data_directory*

   csh shell:

   setenv SP_SYS_VARDIR=*/full_path_of_variable-data_directory*

2. Run the appropriate Database Setup utility for the database. For more information, see Database Setup Utilities in the SharePlex Reference Guide.
3. Repeat these steps for each additional variable-data directory.

5. Start sp_cop instances

You can now run separate instances of **sp_cop** and **sp_ctrl** as needed.

1. Export the SP_SYS_VARDIR environment variable to point to the variable-data directory of the first **sp_cop** instance.

   ksh shell:

   export SP_SYS_VARDIR=*/full_path_of_variable-data_directory*

   csh shell:

   setenv SP_SYS_VARDIR=*/full_path_of_variable-data_directory*

2. Start **sp_cop** with the **-u** option, where *port* is the port assigned to the **sp_cop** instance.

   /splex/*proddir***/bin/sp_cop -u** *port* &

3. In **sp_ctrl**, use the **port** command to set the session to the port number of the **sp_cop** instance you want the commands to affect.

   ./sp_ctrl

   port *number*

4. Repeat these steps for each instance of **sp_cop** that you want to run.

NOTE: If you receive an error message similar to the following, find out if someone else started a session of **sp_cop** using the same port number and variable-data directory. If permissible, kill the processes associated with that session, then start **sp_cop** again.

```
Error cleaning up previous shared memory segment ###.

Cannot delete because there are users attached.

Check if SharePlex processes are running and kill them if necessary.
```

4

Execute Commands in sp_ctrl

This chapter contains instructions for using the **sp_ctrl** command interface to execute commands that configure, control, and monitor SharePlex. The **sp_ctrl** program resides in the **bin** sub-directory of the SharePlex product directory.

NOTE: See the SharePlex Reference Guide for more information about the commands shown in this topic.

Contents

How to Run sp_ctrl

You can run **sp_ctrl** on any machine where SharePlex is installed. The **sp_cop** program must be running in order to run **sp_ctrl**; otherwise it displays an error message similar to this:

```
Your tcp port is not set properly or "sp_cop" is not running.
```

Start sp_ctrl

There are two ways to run sp_ctrl:

- from the command shell of the operating system to issue one command, for example:

  $ /*productdir*/**bin**/**sp_ctrl** *command* [**on** *host*]

- by running the **sp_ctrl** command interface to issue one or more commands, for example:

  $ /*productdir*/**bin**/**sp_ctrl**

  sp_ctrl>*command* [**on** *host*]

where:

- *productdir* is the SharePlex product (installation) directory.
- *command* is the SharePlex command.

- **on** *host* represents one of the command options that allow you to issue a command on the local machine to control SharePlex on a remote machine (if supported by the command), as shown in the following example.

  ```
  $ /productdir/bin/sp_ctrl status on host:port
  ```

sp_ctrl prompt

The **sp_ctrl** prompt appears in one of two ways, depending on whether or not you set a default host and port number.

Prompt	Description
`sp_ctrl>`	Basic **sp_ctrl** prompt
`sp_ctrl(this_host:3304) >`	Prompt when a default system and port are set by issuing the **host** and **port** commands

Exit sp_ctrl

To exit the **sp_ctrl** command-line interface, issue the **exit** or **quit** command.

The **exit** or **quit** command only closes the **sp_ctrl** session. It does not stop the SharePlex replication processes.

Define a Default Port for sp_ctrl

If there is only one instance of SharePlex on a system, **sp_ctrl** detects the port number. However, if you configured more than one SharePlex instance on a system (where each one runs on a different port number) you must use the **port** command to set the session of **sp_ctrl** to the instance for which you want to issue commands.

`sp_ctrl >` **port** *number*

For more information, see Run Multiple Instances of SharePlex on page 48.

For more information about the **port** command, see the SharePlex Reference Guide.

Define a Default Host for sp_ctrl

To define a default machine for all interactive **sp_ctrl** sessions, use the **host** command. This command enables you to enter a series of commands without using the **on***host* option for each one.

`sp_ctr >`**host** *hostname*

The command changes the **sp_ctrl** prompt to include the host name:

```
sp_ctrl (sysA) >
```

The **host** setting applies only to the **sp_ctrl** session in which it is set.

Set a Default Editor for sp_ctrl

You can set the default editor that **sp_ctrl** runs when you issue a command that requires input to an ASCII text file, such as a configuration file. By default, **sp_ctrl** runs **vi** on Unix and Linux.

The default text editors are tested and proven to work with SharePlex. If you change the default editor, the new editor must be a native ASCII text editor. Do not use a word processing program or other non-ASCII program, even if you can save a file to ASCII from that program.

Change the editor on Unix or Linux

Set the EDITOR variable in either of these ways:

- Before you start **sp_ctrl**. This sets the editor only for that session of **sp_ctrl**.
- In the shell startup script on the local machine. This sets the editor permanently, until changed in the startup script. You can override this setting on a per-session basis.

Syntax - ksh shell

export EDITOR=*name_of_editor*

Syntax - csh shell

setenv EDITOR *name_of_editor*

Command Guidelines

Observe the following when issuing commands:

- To issue commands for a machine, **sp_cop** must be running on that machine.
- Enter the syntax exactly as shown in the command description in the SharePlex Reference Guide.
- The maximum string length of a SharePlex command is 255 characters, including spaces. To work around this operating-system limitation, use the **edit** command.
- Use the **redo** command to execute the previous command again without having to retype it. This command is useful when you are making frequent status checks with the information commands, for example using the **qstatus** command to monitor changes in queue volume.
- To view descriptions and syntax for SharePlex commands from within the **sp_ctrl** interface, issue the **help** command. To view just the syntax for a command, issue the usage command.
- Use the **usage** command to view the syntax for a SharePlex command. You can enter the entire command or just the first few keywords. For example, type **usage compare** to view syntax for both the **compare using** and **compare** commands.
- Use the **edit** command to edit a previously issued command.
- Use the **authlevel** command to determine your authorization level for issuing SharePlex commands on a system.

For more information, see About the SharePlex Security Groups on page 252.

Issue Commands on a Remote System

To issue a command that will affect a remote machine and to script commands that include a login name, password, port number, or combination of those items, use one of the [**on***host*] command options. These options are available for most commands.

The following table describes the command options for remote connection using the [**on***host*] options.

These options enable you to issue the command on a remote machine and to script commands that include a login name, password, port number, or combination of those items.

Option	Description
on *host*	Execute the command on a remote system (one other than the one where the current **sp_ctrl** session is running). You are prompted for login credentials for the remote system. If used, must be the last component of the command syntax. **Example:** `sp_ctrl(sysB)>`**status on SysA**
on *host:portnumber*	Execute the command on a remote system when a remote login and port number must be provided. If used, must be the last component of the command syntax. **Example:** `sp_ctrl(sysB)>`**status on SysA:8304**
on *login/password@host*	Execute the command on a remote system when a remote login, password, and host name must be provided. If used, must be the last component of the command syntax. **Example:**`sp_ctrl(sysB)>`**status on john/spot5489@SysA**
on *login/password@host:portnumber*	Execute the command on a remote system when a remote login, password, host name, and port number must be provided. If used, must be the last component of the command syntax. **Example:** `sp_ctrl(sysB)>`**status on john/spot5489@SysA:8304**

Issue Commands for Clustered Systems

To issue **sp_ctrl** commands on clustered systems, use the name set with the `SP_SYS_HOST_NAME` parameter as the host in the [**on** *host*] options when connecting from a remote system, or set it as the default for **sp_ctrl** by using the **host** command. For more information about configuring SharePlex within a cluster, see the SharePlex Installation and Setup Guide.

5

Set SharePlex Parameters for Oracle

SharePlex parameters control and tune various aspects of replication.

Contents

View and Set Parameters

A SharePlex Administrator (defined as a member of the ***SharePlex Admin group***) can change parameters that are designated as user-configurable and is the only user authorized to do so.

View parameters

Use the **list param** command in **sp_ctrl** to view user-configurable SharePlex parameters. It displays parameter names, current settings, default values (if the parameter has been changed), and set-at points. The set-at point indicates when changes to a parameter will take effect. Possible set-at points are:

- *Live* means a change takes effect immediately.
- *Restart Process* means a change takes effect after the affected SharePlex process is restarted.
- *Restart Cop* means a change takes effect after sp_cop is restarted.

Additional options are available for viewing:

- All SharePlex parameters.
- Only parameters whose values have changed.
- Parameters relating to a specific SharePlex module.

To view descriptions of the SharePlex parameters, see the SharePlex Reference Guide.

Set parameters

Parameters can be set in the following ways:

- With the **set param** command through the **sp_ctrl** interface. This is the preferred method because the new value remains intact no matter how many times replication stops and starts. The syntax is:

  set param *parameter_name value*

  Example:

- **set param SP_OCT_REPLICATE_ALL_DDL 1**

- As environment variables on Unix and Linux systems prior to starting **sp_cop**. The new value remains in effect only for that session of **sp_cop**.

Parameters for the Capture, Read, Export, Import, and Post processes can be set on a per-process basis when there are multiple instances of a process for an instance of SharePlex.

Set SharePlex parameters through sp_ctrl

The recommended way to change a SharePlex parameter is to use the **set param** command in **sp_ctrl**.

To restore a parameter's setting to its default value, use the **reset param** command.

To view descriptions of the SharePlex commands, see the SharePlex Reference Guide.

Set SharePlex parameters as environment variables

On Unix and Linux, a SharePlex parameter can be set as an environment variable. The environment variable overrides the setting in the **param-defaults** files, but only for the session of **sp_cop** for which it was set. If you shut down **sp_cop** and restart it without resetting the environment variable, SharePlex uses the default setting in the **param-defaults** file.

To set a SharePlex parameter as an environment variable on Unix and Linux systems, use one of the following commands. Set the environment variable before starting **sp_cop** or, if **sp_cop** is running, restart **sp_cop** for the new setting to take effect.

```
ksh shell:

     $ export parameter_name=value

csh shell:

     $ setenv parameter_name value
```

Because of the temporary nature of environment variables, avoid using them if possible; instead make your changes with the **set param** command. When you rely on environment variables, especially when there are multiple users of SharePlex, you incur the risk of someone forgetting to set them (or using an incorrect value) when they restart **sp_cop**. That can have a significant, negative impact on replication and can result in the need to resynchronize the data.

Where Parameter Information is Stored

The following files store SharePlex parameter settings:

- The **param-defaults** file stores default settings that were set by developers for optimal replication performance under most conditions. The **param-defaults** file resides in the **data** sub-directory of the SharePlex *product* directory. The data in this file does not change unless a new version of SharePlex is installed.

  IMPORTANT: ***Never edit this file.***

- The **paramdb** file stores user-defined parameter settings — values that were changed from their defaults by a SharePlex Administrator using the **set param** command. Also stored in this file are the SharePlex Oracle user and the SharePlex user's password.

  When users execute the **Reset Param** command on sp_ctrl, the earlier entry stored in the paramdb gets removed.

  The **paramdb** resides in the **data** sub-directory of the SharePlex *variable-data* directory. It starts out empty, and as SharePlex Administrators change parameter values, those values are added to it. User-defined parameter values override SharePlex default values when SharePlex is running. All of the settings in the **paramdb** file remain intact when a new version of SharePlex is installed.

6

Configure SharePlex to Replicate Data

SharePlex uses a configuration file to determine which tables to replicate and where to send the replicated data. This file also provides any special processing instructions, such as column mapping and data filtering. This chapter contains the information that you need to know in order to create a configuration file.

Once you understand the basics of how to create a configuration file and route data to your targets, you can move on to any of the more advanced configurations as needed. Documentation for those configurations is included in this guide.

Contents

Ensure Compatible Source-Target Mapping

The following guidelines help you ensure that the source and target objects that you want to map in a replication configuration are compatible.

Object names

For most replication strategies, the name and/or owner of a source object can be different from that of its target object. SharePlex replicates to the correct object because you specify it by owner and name within the configuration file. For high-availability configurations, the owner and name of a source table should be the same as the owner and name of the target table.

Source and target rows

Corresponding source and target rows must contain the same values to accurately reflect the source, unless transformation is being used.

One database in the configuration can have more or fewer rows than another database in the configuration. You can control which rows are replicated through horizontally partitioned replication. For more information, see Configure Horizontally Partitioned Replication on page 118.

Source and target columns

Corresponding source and target columns must:

- Contain compatible data types (same type, size, precision).
- Have identical names, unless you use column mapping in the configuration file. For more information, see Map Source and Target Columns on page 99.
- Be the same letter case if one of the databases is case-sensitive, but the source or target of that database is not case-sensitive. You can use column mapping to work around this issue. For more information, see Map Source and Target Columns on page 99.

A target table can have more columns than the source table.

- If the source and related target column names are identical, SharePlex will ignore the extra columns in the target table.
- If the source and target column names are not identical, SharePlex maps a one-to-one relationship in the order that the columns are defined in each table (for example, map the first column in the source to the first column in the target, map the second column to the second column, and so forth).
- To avoid Oracle errors if the extra (non-mapped) columns are NOT NULL, define default values for those columns. For more information, see Map Source and Target Columns on page 99.

A target table can have fewer columns than the number of columns in the source table, but you must use vertically partitioned replication to replicate only that subset of the source columns that matches the rows of the target. For more information, see Configure Vertically Partitioned Replication on page 141.

Create a Configuration File

To configure SharePlex to replicate data, you create a configuration file.

This is an ASCII text file in which you specify:

- The data that you want SharePlex to replicate, including any filtering or partitioning of rows or columns
- The target name and type
- The system (and database if applicable) to which the data must be delivered.

Only a SharePlex Administrator or operator has the authority to create a configuration file.

When your configuration file is completed, you activate the configuration with the **activate config** command to begin replication. For more information, see Start Replication on your Production Systems on page 254.

Create a configuration file

You can create a configuration file by hand in **sp_ctrl** or, if your data structure supports it, you can automate the creation of a configuration file with a script.

Create the configuration file on the system *from which data is to be replicated*, typically the source system, but strategies such as active-active replication require configuration files on more than one system.

To create a configuration file in sp_ctrl:

1. Run **sp_ctrl** from the **bin** sub-directory of the SharePlex product directory.
2. In sp_ctrl, issue the **create config** command.

   ```
   sp_ctrl> create config config_name
   ```

   This command opens a file in the default text editor that is set for the operating system.

   NOTE: You can change the default editor that **sp_ctrl** uses. For more information, see Set a Default Editor for sp_ctrl on page 55.

3. Complete the configuration file. For more information, see Structure of a configuration file on page 63.

   IMPORTANT! All configurations must reside in the **config** sub-directory of the SharePlex variable-data directory. Configuration files outside this directory cannot be activated. SharePlex places configurations in this directory by default when you create them through the **sp_ctrl** interface with the **create config** command. If you create the configuration directly through a text editor, make certain to save it to the **config** sub-directory.

To build a configuration file with a script:

NOTE: Valid for Oracle only

SharePlex provides the following scripts that can be used to automate the building of your configuration file if the source and target object names are identical. These scripts support Oracle Database source and targets only.

Option	Description	To get more information
config.sql	Builds a configuration file that includes all tables and sequences that are in the database. Source and target object names must be identical.	See Configuration Scripts in the SharePlex Reference Guide.
build_config.sql	Builds a configuration file that includes all tables in a schema. Source and target object names must be identical.	See Configuration Scripts in the SharePlex Reference Guide.

Structure of a configuration file

A basic configuration file looks like the following:

```
# comment: basic SharePlex configuration file

datasource_specification

#source specification          target specification          routing map

source_owner.object1          target_owner.object1          routing_map

source_owner.object2          target_owner.object2          routing_map

source_owner.object3          target_owner.object3          routing_map
```

The basic components of a configuration file are as follows:

Component	Description	Syntax examples
# Comments	Lines that describe the file or provide other information about the contents to viewers, but are not used by SharePlex. Precede each comment line with a pound (#) sign. Comments may be entered anywhere in the configuration file.	# This is a comment.
Datasource specification	Syntax that specifies the source database. This component	**Datasource:o.***SID*

Component	Description	Syntax examples
	must always be the first non-commented line of a configuration file. It has the following syntax elements, all on the same line with no spaces: • The keyword **Datasource** followed by a colon (**:**). The word "Datasource" is not case-sensitive. • The *database specification*. For more information, see Database Specifications in a Configuration File on page 67.	
Source specification	The fully qualified name of a supported source object, in the form of *owner.object*. The *owner* can be a schema or a database, depending on how the database stores objects logically. See: How to Qualify Object Names on page 65 How to Specify Case-Sensitive Names on page 66 You can use wildcards to specify multiple objects. Owner names cannot be wildcarded. For more information, see Use Wildcards to Specify Multiple Objects on page 84.	*src_owner.table* *src_owner.sequence*
Target specification	The target to which the replicated data is applied. Targets supported by SharePlex are: • The fully qualified name of a table (or tables) in a relational database. You can use wildcards to specify multiple objects. Owner names cannot be wildcarded. **For more information, see:** • How to Qualify Object Names on page 65 • How to Specify Case-Sensitive Names • Use Wildcards to Specify Multiple Objects on page 84 • Target Specifications in a Configuration File on page 68 • An Oracle sequence (or wildcard specification). • A file that contains XML or SQL records. • A JMS queue or topic. • A Kafka topic. • A change-history table that maintains a record of all changes made to a source table (also known as change data capture)	*tgt_owner.table* *tgt_owner.sequence* **!file**[:*tgt_owner.table*] **!jms**[:*tgt_owner.table*] **!kafka**[:*tgt_owner.table*] **!cdc:***tgt_owner.table*

Component	Description	Syntax examples
Routing map	One or more *routes* that send the data to the system that contains the target object specified with the *target specification*. A route consists of the following: 1. The name of the machine that hosts the target. 2. (Database targets only) The **@** symbol followed by the target database specification. For more information, see Database Specifications in a Configuration File on page 67. The database specification is absent if the target is JMS, Kafka, or a file. There can be no spaces between any characters in the routing map. For more information, see Routing Specifications in a Configuration File on page 69.	*host***@o.***SID* *host***@o.***PDBalias* *host***@o.***tns_alias* *host***@r.***database_name* *host* *host***@c.***SID* **Compound routing map:** *host* **@o.***SID***+***host***@r.***database*[...]

How to Qualify Object Names

So that SharePlex can determine the correct objects to capture from, and post to, you must qualify the object names in the configuration file in the same way that the database stores them logically. The general way this is indicated in SharePlex syntax is:

owner.object

Where:

- *owner* is the schema or database that contains the object (or objects, if wildcarded), depending on how that container is defined by the database.
- *object* is the name of the object or a wildcard specification to specify multiple objects.

When defining source or target objects in the configuration file, follow these guidelines for specifying the *owner* component:

Database	Fully qualified object name
Aurora	database_name.object_name
MySQL	database_name.object_name
Oracle	schema_name.object_name
PostgreSQL	schema_name.object_name
SQL Server	schema_name.object_name
SAP HANA	schema_name.object_name

How to Specify Case-Sensitive Names

This topic shows you how to specify case-sensitive names in the configuration file, for example when specifying table names or if you need to specify column names explicitly in a column mapping.

Case-sensitive object names

If the owner or name of an object is case-sensitive in the database, you must enclose that name within quotes in the SharePlex configuration file.

IMPORTANT: This applies whether the database itself requires a case-sensitive name to be within quotes, such as Oracle, or whether the database accepts names that are spelled out in their case-sensitive form without quotes, like SQL Server.

To enforce case-sensitive object names:

Specify the name in its correct case and enclose it within double quotes.

Correct way

- This is how to specify an object where both the owner and object names are both case-sensitive:

  "**Owner**"."**Object**"

- This is how to specify an object where only one of the components is case-sensitive:

  owner."**Object**" or "**Owner**".**object**

  The name that is not case-sensitive can be specified in any case.

Examples of both ways:

```
Datasource o.oraA
   sales."Emp"              "Sales"."Emp"                  sysB@o.oraB
```

Incorrect way

This is not correct, because both components are within one set of quotes:

"**Sales**.**Employees**"

Case-sensitive column names

Ordinarily, column names are not specified in the configuration file, unless source column names need to be mapped to different target column names by means of a column mapping (see Map Source and Target Columns on page 99). However, if the names of any pair of source and target columns have difference cases, you may need to include them in a column mapping to enforce their case sensitivity. Whether or not a column mapping is required depends on the target type: Oracle or Open Target.

To enforce case-sensitive column names for Oracle to Oracle replication

The Oracle Post process does not perform case conversion of column names automatically for Oracle to Oracle replication. If the case is different between source and target columns, you must use a column map to map the case of the source names to the case of the target names. To get Post to enforce the case, specify the name in its correct case and enclose it within double quotes.

This is an example of case-sensitive column name mapping in a column map:

```
      Datasource o.oraA
sales.emp(ID,"first","last") sales.emp(ID,"First","Last")        sysB@o.oraB
```

To enforce case-sensitive column names to Open Target:

The Open Target Post process performs case conversion of column names automatically. If replicating to target columns that have a different case from their source columns, no column mapping is needed.

Database Specifications in a Configuration File

A database specification is required in the following components of the configuration file:

- the datasource (source datastore) specification
- routing map (target datastore and location) specification

Database	Database type notation*	Database Identifiers
Oracle source	**o.**	Depending on the Oracle database configuration, use one of the following. This is the string that SharePlex will use to connect to the database. • The Oracle SID of a regular (non-CDB) Oracle database, as in **o.ora12**. • The TNS alias of a pluggable database (PDB) in an Oracle container database (CDB), as in **o.pdb1**. • The global TNS alias of an Oracle RAC cluster, as in **o.rac1**. SharePlex connects to an Oracle RAC instance through this TNS alias, which is mapped locally on each node to the Oracle SID of the local Oracle instance. For more information about creating this alias, see Preinstallation instructions for Oracle cluster in the SharePlex Installation Guide.
Open Targe targets	**r.**	Use to specify the name of an Open Target (non-Oracle) target database, as in **r.mydb**. **IMPORTANT!** Use the actual database name. **Do not** use the ODBC datasource name (DNS) or database instance name. If the database name is case-sensitive, specify it that way.
Oracle	**c.**	Use in a routing map to specify the Oracle SID, TNS alias, or global RAC TNS alias

Database	Database type notation*	Database Identifiers
change-history target		of an Oracle change history database, as in **c.ORA12CH**. In this configuration, SharePlex applies all source transactions as INSERTs to the target tables, to maintain a history of every operation performed. For more information, see Configure Replication to a Change History Target on page 149.

* **NOTE:** The dot is required.

Target Specifications in a Configuration File

The following table shows how to specify a target table or non-table target in a configuration file.

Target	Target Specification	Description
Database table	*tgt_owner.table*	The fully qualified name of a database table. For more information, see How to Qualify Object Names on page 65.
Database sequence	*tgt_owner.sequence*	The fully qualified name of a sequence. For more information, see How to Qualify Object Names on page 65
File	**!file**[:*tgt_owner.table*]	The **!file** designator directs Post to write change operations to a file in **SQL, XML, or JSON** format. The file name is applied internally by SharePlex. Optionally, you can specify the fully qualified name of a target table if the data will be consumed by a process that ultimately applies it to a database table.
JMS	**!jms**[:*tgt_owner.table*]	The **!jms** designator directs Post to write change operations to a JMS queue or topic in XML format. The queue or topic name can be defined by using the **target** command. Optionally, you can specify the fully qualified name of a target table if the data will be consumed by a process that ultimately applies it to a database table.
Kafka	**!kafka**[:*tgt_owner.table*]	The **!Kafka** designator directs Post to write change operations to a Kafka topic in XML or JSON format. The topic name can be defined by using the **target** command.

Target	Target Specification	Description
		Optionally, you can specify the fully qualified name of a target table if the data will be consumed by a process that ultimately applies it to a database table.
Change history table	**!cdc:***tgt_owner.table*	The **!cdc** designator directs Post to insert every data change to the table as a new row, rather than overlay the old data with new data. Specify the fully qualified name of the change history table. For more information, see Configure Replication to a Change History Target on page 149.

Routing Specifications in a Configuration File

The following instructions show you how to build a routing map based on where you want to send the source data. A routing map sends replicated data to the correct target on the correct target system, or systems.

For details about the components of these configurations, see:

Database Specifications in a Configuration File

Target Specifications in a Configuration File

Routing to one target

A *simple routing map* sends replicated data from one source object to one target object.

datasource_specification		
src_owner.table	*tgt_owner.table2*	*host2*[**@***database_specification*]
src_owner.table	*tgt_owner.table3*	*host3*[**@***database_specification*]

Routing to a cloud service

There are special routing requirements for database targets that are hosted by a cloud service such as EC2 and RDS on Amazon AWS, Azure SQL in Microsoft Azure, Google Cloud SQL for PostgreSQL, and Compute Virtual Machines in Oracle Cloud Infrastructure. Whether the service is *Infrastructure as a Service* (IaaS) or *Platform as a Service* (PaaS) makes a difference in how you install and configure SharePlex. The following explains these requirements.

IaaS targets

If replicating to a database target hosted in an IaaS cloud service, specify the full endpoint URL as the target host in the routing map.

datasource_specification		
src_owner.table	*tgt_owner.table2*	*endpointURL@database_specification*
src_owner.table	*tgt_owner.table3*	*endpointURL@database_specification*

For example, the following routing map routes to a cloud database on Amazon EC2:

```
ec2-12-345-678-910.compute-1.amazonaws.com@o.myora
```

Alternately, you can map the private IP address of the cloud service to a short name in the local **hosts** file, and then specify that name as the host in the routing map, for example:

```
shortname@o.myora
```

PaaS targets

If replicating to a database target hosted in a PaaS cloud service, there are special installation, setup, and routing requirements. Because SharePlex cannot be installed directly on a PaaS cloud server, you must install SharePlex on either the source server or an intermediary server, from which Post connects to the target cloud database. For more information, see Installation and setup for cloud-hosted databases in the SharePlex Installation and Setup Guide.

Routing to multiple targets

A *compound routing map* sends replicated data from one source object to multiple target objects. It enables you to specify the source and target objects once for all routes, rather than type a separate configuration entry for each route. Only one *target specification* can be used in a compound routing map, so all of the target objects must be identical as follows:

- All are of one type: All the same database object type *or* all a JMS queue *or* all a JMS topic *or* all a Kafka topic, *or* all a file (but no combination of these).
- All have the same fully qualified name, including any table specifications in a JMS, Kafka, or file target specification.
- All have identical column mappings or key mappings, if used. For more information about these mappings, see:
  - Define a Unique Key on page 92
  - Map Source and Target Columns on page 99

NOTES:

- Certain routing limitations apply when using vertically partitioned replication. For more information, see Configure Vertically Partitioned Replication on page 141.
- If any target has a different qualified name from the other targets of the same source object, you must use a simple routing map for that target.

datasource_specification		
src_owner.table	*tgt_owner.table*	*host1*[*@database_specification*]+*host2*[*@database_specification*][...]

Routing between objects on the same system

You can replicate between the following:

- For Oracle, you can replicate between objects that are in the same database or in different databases on the same system. You can replicate between objects that have the same name, so long as their owners are different.

When SharePlex replicates between objects on the same system, it does not create Import and Export processes. You can force SharePlex to create Import and Export processes by using the following routing map. If you do not need the Import or Export processes, omit the ***host**** portion of the routing map.

Configuration with replication to objects in the same or different database on the same system

datasource_specification		
src_owner.table	*tgt_owner.table*	*host*host*[@*database_specification*]

Routing limitations

- By default, SharePlex supports replication to a maximum of 19 direct target systems. That is the maximum number of processes that can read the export queue. To replicate to more than 19 targets, use named export queues. With each additional queue that you add, you can replicate to 19 additional targets. For more information, see Configure Named Export Queues on page 107.
- Each instance of **sp_cop** on a system permits a maximum of 1024 different routes. This limitation includes each route that uses a different named post queue (see Configure Named Post Queues on page 113.) If your replication strategy requires more than 1024 routes, consider using one or more intermediary systems to divide the routes among multiple **sp_cop** instances. For more information, see Configure Replication to Share or Distribute Data on page 155.
- By default, each **sp_cop** instance allows a total of 25 queues on a system. There will always be one capture queue on a source system and one post queue on a target. Therefore, you can have as many as 24 named export queues on a source system and 24 named post queues on a target system. If a system serves as both a source and target, you will have both a capture queue and a post queue. That allows you to create up to 23 named queues of either type (or a mix of both). If system memory permits, you can change the number of allowed queues by setting the SP_QUE_MAX_QUEUES parameter. See the SharePlex Reference Guide for more information about this parameter.

Configuration Examples by Data Source and Target

These are examples of basic configuration files according to each possible datasource type and target type.

Replicate from a regular Oracle instance to a regular Oracle instance

This configuration applies to on-premises and IaaS cloud deployments. See the System Requirements in the SharePlex Installation and Setup Guide for supported database versions and platforms.

Datasource:o.*SID*		
src_owner.table	*tgt_owner.table*	*host***@o.***SID*

Example

The following example replicates table `SCOTT.EMP` from Oracle instance `oraA` to target table `SCOTT.EMP2` in Oracle instance `oraB` on target system `sysprod`.

```
Datasource:o.oraA
SCOTT.EMP     SCOTT.EMP2     sysprod@o.oraB
```

Replicate from Oracle to target Oracle in PaaS Cloud

To replicate from an on-premises or IaaS-based Oracle source to a target Oracle database hosted in a PaaS cloud, the SharePlex target components (Import and Post) must run on the source server or on an intermediary server. Post connects through a remote connection using a TNS alias. To set up this topology, see Installation and setup for cloud-hosted databases in the SharePlex Installation and Setup Guide.

Datasource:o.*SID*		
src_owner.table	*tgt_owner.table*	*source_or_intermediary_host***@o.***SID*

Example

The following example replicates table `SCOTT.EMP` from Oracle instance `oraA` to target table `SCOTT.EMP2` in the PaaS cloud Oracle instance `oraB`. Post runs on intermediary target system `sysprod2`.

```
datasource:o.oraA
SCOTT.EMP      SCOTT.EMP2     sysprod2@o.oraB
```

Replicate from a regular Oracle instance to an open target database

This configuration applies to on-premises and IaaS cloud deployments. See the System Requirements in the SharePlex Installation and Setup Guide for supported database versions and platforms.

Datasource:o.*SID*		
src_owner.table	*tgt_owner.table*	*host***@r.***database_name*

Example

The following example replicates table `SCOTT.EMP` from Oracle instance `oraA` to target table `Scott2.Emp2` in Open Target database `mydb` on target system `sys2`. The target table is case-sensitive.

```
Datasource:o.oraA
SCOTT.EMP    "Scott2"."Emp2"    sys2@r.mydb
```

Replicate from a regular Oracle instance to a file in XML or SQL format

Datasource:o.*SID*		
src_owner.table	**!file**	*host*

Example

The following example replicates table `SCOTT.EMP` from Oracle instance `oraA` to a file on target system `sysprod`.

```
Datasource:o.oraA
SCOTT.EMP   !file   sysprod
```

Replicate from a regular Oracle instance to a JMS queue or topic

Datasource:o.*SID*		
src_owner.table	**!jms**	*host*

Example

The following example replicates table `SCOTT.EMP` from Oracle instance `oraA` to a JMS queue on target system `sysprod`.

```
Datasource:o.oraA
SCOTT.EMP   !jms   sysprod
```

Replicate from a regular Oracle instance to a Kafka topic

Datasource:o.*SID*		
src_owner.table	**!kafka**	*host*

Example

The following example replicates table `SCOTT.EMP` from Oracle instance `oraA` to a Kafka topic using SharePlex target system `sysprod`.

```
Datasource:o.oraA
SCOTT.EMP   !kafka  sysprod
```

Replicate from Oracle to Kafka using SSL encryption

This configuration is applicable for the Kafka target.

Pre-requisites:

Configure the Kafka target with the following parameters to post data onto Kafka before starting the replication with SSL encryption:

- sp_ctrl target x.kafka set kafka broker = <kafka-server-hostname>:<Kafka-server-port>
- sp_ctrl target x.kafka set kafka security.protocol = SSL
- sp_ctrl target x.kafka set kafka ssl.ca.location = <ca-cert-file-path>
- sp_ctrl target x.kafka set kafka ssl.certificate.location = <.pem-file-path>
- sp_ctrl target x.kafka set kafka ssl.key.location = <ssl-key-file-path>
- sp_ctrl target x.kafka set kafka ssl.key.password = <key-password>
- sp_ctrl target x.kafka set kafka sasl.mechanisms = PLAIN
- sp_ctrl target x.kafka set kafka api.version.request = true

NOTES:

- All the values in the <> need to be replaced with actual parameters from the Kafka producer.
- Before setting the target Kafka replication parameters, you need to first stop Poster, set the parameters, and then start the Poster.

This configuration applies to on-premises and IaaS cloud deployments. See the System Requirements in the SharePlex Installation and Setup Guide for supported database versions and platforms.

Datasource:o.*SID*		
src_owner.table	!kafka	*host*

Example:

The following example replicates table SCOTT.EMP from Oracle instance oraA to the Kafka server using SharePlex target system sysprod

```
Datasource:o.oraA
SCOTT.EMP     !kafka     sysprod
```

Replicate from Oracle to Kafka using SASL authentication

This configuration is applicable for the Kafka target.

Pre-requisites:

Configure the Kafka target with the following parameters to post data onto Kafka before starting the replication with SASL authentication.

- sp_ctrl target x.kafka set kafka api.version.request=true
- sp_ctrl target x.kafka set kafka sasl.mechanisms=PLAIN
- sp_ctrl target x.kafka set kafka sasl.username=<username>
- sp_ctrl target x.kafka set kafka sasl.password=<password>
- sp_ctrl target x.kafka set kafka security.protocol=SASL_PLAINTEXT

NOTES:

- All the values in the <> need to be replaced with actual parameters from the Kafka producer.
- Before setting the target Kafka replication parameters, you need to first stop Poster, set the parameters, and then start the Poster.

This configuration applies to on-premises and IaaS cloud deployments. See the System Requirements in the SharePlex Installation and Setup Guide for supported database versions and platforms.

Datasource:o.*SID*		
src_owner.table	!kafka	*host*

Example:

The following example replicates table SCOTT.EMP from Oracle instance oraA to the Kafka server using SharePlex target system sysprod.

```
Datasource:o.oraA
SCOTT.EMP     !kafka     sysprod
```

Replicate from Oracle to Kafka using Kerberos authentication

This configuration is applicable for the Kafka target.

Pre-requisites:

Configure the Kafka target with the following parameters to post data onto Kafka before starting the replication with kerberos authentication

- sp_ctrl target x.kafka set kafka sasl.kerberos.keytab = <kerberos-keytab-file>
- sp_ctrl target x.kafka set kafka sasl.kerberos.kinit.cmd = <kerberos-kinit-cmd>

NOTE: The `{broker.name}` property is no longer supported for the `sp_ctrl target x.kafka set kafka sasl.kerberos.kinit.cmd = <kerberos-kinit-cmd>` parameter.

- sp_ctrl target x.kafka set kafka sasl.kerberos.min.time.before.relogin = <relogin-time>
- sp_ctrl target x.kafka set kafka sasl.kerberos.principal = <kerberos-principal>
- sp_ctrl target x.kafka set kafka sasl.kerberos.service.name = <kerberos-service-name>
- sp_ctrl target x.kafka set kafka sasl.mechanisms = GSSAPI

NOTES:

- All the values in the <> need to be replaced with actual parameters from the Kafka producer.
- Before setting the target Kafka replication parameters, you need to first stop Poster, set the parameters, and then start the Poster.

This configuration applies to on-premises and IaaS cloud deployments. See the System Requirements in the SharePlex Installation and Setup Guide for supported database versions and platforms.

Datasource:o.*SID*		
src_owner.table	!kafka	*host*

Example:

The following example replicates table SCOTT.EMP from Oracle instance oraA to the Kafka server using SharePlex target system sysprod

```
Datasource:o.oraA
SCOTT.EMP    !kafka    sysprod
```

Replicate from Oracle to Kafka using mTLS authentication

This configuration is applicable for the Kafka target.

Note: Before setting the target Kafka replication parameters, you need to first stop Poster, set the parameters, and then start the Poster.

This configuration applies to on-premises and IaaS cloud deployments. See the System Requirements in the SharePlex Installation and Setup Guide for supported database versions and platforms.

Datasource:o.*SID*		
src_owner.table	!kafka	*host*

Example:

The following example replicates table SCOTT.EMP from Oracle instance oraA to the Kafka server using SharePlex target system sysprod

```
Datasource:o.oraA
SCOTT.EMP    !kafka    sysprod
```

Replicate data from Oracle to Azure Event Hubs

These configurations need to be done for replicating data from source to Azure Event Hubs. SharePlex for Kafka is used to communicate with Azure Event Hubs through the Kafka Event Hubs connectors.

Pre-requisites:

Configure the SharePlex source machine with the following Kafka parameters to post data onto Azure Event Hubs before starting the replication.

- sp_ctrl Target x.kafka set kafka api.version.request = true
- sp_ctrl target x.kafka set kafka broker = <Azure Event Hubs namespace>:<Kafka-server-port>
- sp_ctrl Target x.kafka set kafka sasl.mechanisms = PLAIN
- sp_ctrl Target x.kafka set kafka sasl.username = $ConnectionString
- sp_ctrl Target x.kafka set kafka sasl.password =<Primary key generated in Event Hubs namespace>
- sp_ctrl Target x.kafka set kafka security.protocol = SASL_SSL
- sp_ctrl Target x.kafka set kafka topic = <Kafka Event Hubs topic generated inside Event Hubs namespace>

NOTES:

- All the values in the <> need to be replaced with actual parameters from Azure Event Hubs.
- Before setting the target Kafka replication parameters, you need to first stop Poster, set the parameters, and then restart the Poster.

This configuration applies to on-premises and IaaS cloud deployments. See the System Requirements in the SharePlex Installation and Setup Guide for supported database versions and platforms.

Datasource:o.*SID*		
src_owner.table	!kafka	*src_hostname*

Example:

The following example replicates table SCOTT.EMP from Oracle instance oraA to the Azure Event Hub using SharePlex target system sysprod.

```
Datasource:o.oraA
SCOTT.EMP     !kafka     sysprod
```

Replicate data from Oracle to SQL Server

This configuration applies to on-premises and IaaS cloud deployments. See the System Requirements in the SharePlex Installation and Setup Guide for supported database versions and platforms.

Datasource:o.*SID*		
src_owner.table	dst_owner.table	*dst_hostname*

Example:

The following example replicates table SCOTT.EMP from Oracle instance oraA to the MS SQL Server on host sysprod

```
Datasource:o.oraA
SCOTT.EMP     SCOTT.EMP    sysprod@r.sp_ss
```

Replicate Data from Oracle to Azure SQL database

This configuration applies to on-premises and IaaS cloud deployments. See the System Requirements in the SharePlex Installation and Setup Guide for supported database versions and platforms.

Datasource:o.*SID*		
src_owner.table	dst_owner.table	*dst_hostname*

Example:

The following example replicates table SCOTT.EMP from Oracle instance oraA to the Azure SQL database on host sysprod

```
Datasource:o.oraA
SCOTT.EMP     SCOTT.EMP    sysprod@r.azuresqldb
```

Replicate data from Oracle to PostgreSQL database

This configuration applies to on-premises and IaaS cloud deployments. See the System Requirements in the SharePlex Installation and Setup Guide for supported database versions and platforms.

Datasource:o.*SID*		
src_owner.table	dst_owner.table	*dst_hostname*

Example:

The following example replicates table SCOTT.EMP from Oracle instance oraA to PostgreSQL target server on host sysprod:

```
Datasource:o.oraA
SCOTT.EMP     SCOTT.EMP    sysprod@r.sp_ss
```

Replicate data from Oracle to MySQL database

This configuration applies to on-premises and IaaS cloud deployments. See the System Requirements in the SharePlex Installation and Setup Guide for supported database versions and platforms.

Datasource:o.*SID*		
src_owner.table	dst_owner.table	*dst_hostname*

Example:

The following example replicates table SCOTT.EMP from Oracle instance oraA to MySQL target server on host sysprod:

```
Datasource:o.oraA
SCOTT.EMP     SCOTT.EMP    sysprod@r.sp_ss
```

Replicate from and to an Oracle pluggable database (PDB) in a container database (CDB)*

This configuration applies to on-premises and IaaS cloud deployments. See the System Requirements in the SharePlex Installation and Setup Guide for supported database versions and platforms.

Datasource:o.*PDBalias*		
src_owner.table	*tgt_owner.table*	*host***@o.***PDBalias*

Example

This example replicates table `SCOTT.EMP` from an Oracle PDB that uses the TNS alias of `aliasA` to target table `SCOTT.EMP` in an Oracle PDB that uses the TNS alias of `aliasB` on target system `sysprod`.

```
Datasource:o.aliasA
SSCOTT.EMP     SCOTT.EMP     sysprod@o.aliasB
```

* You can also replicate data from an Oracle PDB to any other supported target. For more information, see Configure Capture and Delivery on page 105.

Replicate to maintain a change history target

This configuration applies to on-premises and IaaS cloud deployments. See the System Requirements in the SharePlex Installation and Setup Guide for supported database versions and platforms.

Datasource: o.*SID*		
src_owner.table	**!cdc:***tgt_owner.table*	*host***@c.***SID*

Example

The following example replicates table `SCOTT.EMP` from Oracle instance `oraA` to change-history target table `SCOTT.EMP2` in Oracle instance `oraB` on target system `sysprod`.

```
Datasource:o.oraA
SCOTT.EMP      !cdc:SCOTT.EMP2      sysprod@c.oraB
```

For more information, see Configure Replication to a Change History Target on page 149.

Replicate data from Oracle to Oracle using Extended Data Types

These configurations need to be done for replicating data from Oracle to Oracle using extended data types.

Pre-requisite:

Your Oracle database should be supporting the Extended Data Type.

Limitation:

SharePlex does not replicate data with Extended Data Type when target type is SQL, JMS, File, or Kafka.

This configuration applies to on-premises and IaaS cloud deployments. See the System Requirements in the SharePlex Installation and Setup Guide for supported database versions and platforms.

Datasource:o.*SID*		
src_owner.table	tgt_owner.table	*host***@o.***SID*

Example

The following example replicates table `SCOTT.EMP` from Oracle instance `oraA` to target table `SCOTT.EMP2` in Oracle instance `oraB` on target system `sysprod`.

```
Datasource:o.oraA
SCOTT.EMP     SCOTT.EMP2     sysprod@o.oraB
```

Replicate data from Oracle to Snowflake

This configuration applies to on-premises and IaaS cloud deployments. See the System Requirements in the SharePlex Installation and Setup Guide for supported database versions and platforms.

Datasource:o.*SID*		
src_schema.table	dst_schema.table	*dst_hostname*

Example:

The following example replicates table scott.emp from Oracle instance oraA to the Snowflake on host sysprod

```
Datasource:o.oraA
"scott"."emp"    "SCOTT"."EMP"    sysprod@r.dbname
```

Replicate data from PostgreSQL to PostgreSQL database

This configuration applies to on-premises and IaaS cloud deployments. See the System Requirements in the SharePlex Installation and Setup Guide for supported database versions and platforms.

Datasource:r.*dbname*		
src_schema.table	dst_schema.table	*dst_hostname*

Example:

The following example replicates table SCOTT.EMP from a PostgreSQL instance dbnameA to a PostgreSQL target server on host *hostB*:

```
Datasource:r.dbnameA
scott.emp    scott.emp    hostb@r.mydb
```

Replicate data from PostgreSQL to Oracle database

This configuration applies to on-premises and IaaS cloud deployments. See the System Requirements in the SharePlex Installation and Setup Guide for supported database versions and platforms.

Datasource:r.*dbname*		
src_schema.table	dst_owner.table	*dst_hostname*

Example:

The following example replicates table SCOTT.EMP from a PostgreSQL instance dbnameA to an Oracle target server on host *hostB*:

```
Datasource:r.dbnameA
"scott"."emp"    "scott"."emp"    hostB@o.mydb
```

Replicate from a PostgreSQL instance to a Kafka topic

Datasource:r.*dbname*		
src_schema.table	**!kafka**	*host*

Example

The following example replicates table `SCOTT.EMP` from PostgreSQL instance `testdbA` to a Kafka topic using SharePlex target system `targetHost`.

```
Datasource:r.testdbA
scott.emp   !kafka  targetHost
```

Replicate data from PostgreSQL to SQL Server

This configuration applies to on-premises and IaaS cloud deployments. See the System Requirements in the SharePlex Installation and Setup Guide for supported database versions and platforms.

Datasource:r.*dbname*		
src_schema.table	dst_owner.table	*dst_hostname*

Example:

The following example replicates table SCOTT.EMP from PostgreSQL instance testdbA to MS SQL Server on host sysprod

```
Datasource: r.testdbA
SCOTT.EMP    SCOTT.EMP   sysprod@r.dbname
```

Replicate data from PostgreSQL to Snowflake

This configuration applies to on-premises and IaaS cloud deployments. See the System Requirements in the SharePlex Installation and Setup Guide for supported database versions and platforms.

Datasource:r.*dbname*		
src_schema.table	dst_schema.table	*dst_hostname*

Example:

The following example replicates table scott.emp from PostgreSQL instance testdbA to Snowflake on host sysprod

```
Datasource: r.testdbA
"scott"."emp"    "SCOTT"."EMP"   sysprod@r.dbname
```

Capture from Multiple Local Datasources

You can use one instance of SharePlex to capture from multiple datasources on a system. All of the configurations can be active at the same time.

NOTE: SharePlex does not support multiple active configuration files for the *same* datasource, but it does support multiple active configuration files if each replicates a *different* datasource.

To capture from multiple datasources:

1. Create a configuration file for the first datasource. In each routing map, include a named export queue. For more information, see Configure Named Export Queues on page 107.
2. Create a configuration file for the second datasource. In each routing map, specify a named export queue, but make certain it is different from any of the queues named in the first configuration file. It is important that data from one datasource does not process through the export queues of the other datasource.
3. Create additional configurations with dedicated named export queues, if needed.
4. When you activate the configuration files, use a separate **sp_ctrl** session for each one. For more information, see How to Activate Multiple Configuration Files on page 261.

Use Wildcards to Specify Multiple Objects

You can use wildcard characters to specify multiple objects of a schema in one entry of the configuration file. SharePlex replicates any objects that satisfy the wildcard, except those that you explicitly exclude.

NOTE: Only object names can be wildcarded. Owner names cannot be wildcarded.

Requirements and limitations of wildcard support

- The schemas that contain wildcarded object names must exist on the source and target before the configuration is activated.
- (Oracle) The objects themselves do not have to exist before the configuration is activated, provided the correct parameters are set to required values. For more information, see Control Oracle DDL Replication on page 215.
- **Wildcards are not allowed anywhere in a configuration entry that includes the following:**
    - Vertically partitioned replication. For more information, see Configure Vertically Partitioned Replication on page 141.
    - Horizontally partitioned replication. For more information, see Configure Horizontally Partitioned Replication on page 118.
    - (Oracle) Key definition. For more information, see Define a Unique Key on page 92.
    - (Oracle) Column mapping. For more information, see Map Source and Target Columns on page 99.

  The tables that use these features must be specified in the configuration file separately.

Supported wildcard syntax

SharePlex supports the following SQL wildcards

- Percent (%) wildcard to specify a string. (See the Use Wildcards to Specify Multiple Objects on page 84.)
- Underscore (_) wildcard to specify a single-character.
- For table names that contain a percent sign or an underscore character (for example **emp_salary**), SharePlex recognizes the backslash (\) as an escape character to denote the character as a literal, not a wildcard.

Specify wildcarded names in the configuration file

Use this template for help when specifying a wildcarded name in the configuration file.

Configuration with wildcarded object names

datasource_specification

expand *src_owner.wildcard_name* [**not** (*list*)] *tgt_owner.wildcard_name* *routing_map*

Description of syntax elements

Component	Description
expand	Indicates that the specification contains wildcard characters that must be expanded. When SharePlex detects the **expand** keyword, it queries the database for all objects that match the criteria in the wildcard specification. Without this required keyword, the wildcard characters are assumed to be part of an explicit object name, and no wildcard expansion is performed. **NOTE:** Leave a space between **expand** and the start of the source object specification.
src_owner.wildcard_name	• *src_owner* is the owner of the source objects. Owner names cannot be wildcarded. If wildcards are used in the owner name, SharePlex assumes that they are part of the owner (schema) name. • *wildcard_name* is the wildcarded name of the source objects. Rules: Oracle: The names of the target objects must be identical to those of the source objects, but the objects may belong to different owners.
not (*list*)	An exclusion list that defines objects to omit from the wildcard expansion. Use this option to exclude objects that you do not want to be replicated. **NOTE:** This **not** keyword does not have the same meaning as the SQL wildcard **NOT** operator. • The **not** keyword and parentheses are required elements. • *list* is a comma-separated list of tables owned by the same owner, either wildcarded or explicit. Example: **not (spo%, gen%, product).** Leave a space before and after the **not** keyword. A space is allowed

Component	Description
	after each comma in the list. **NOTE:** If an object that satisfies a wildcard is listed elsewhere in the configuration file, that entry overrides the processing or routing specified in the wildcarded entry. In this case, a **not** clause is *not* needed. See the **Examples**.
tgt_owner.*wildcard_name*	• *tgt_owner* is the owner of the target objects. • *wildcard_name* is the wildcarded name of the target objects. The target specification must be in the form of *owner*.%. Partially expanded target wildcarded names are not supported, such as *owner.tab*%.
routing_map	Any valid routing map. For more information, see Routing Specifications in a Configuration File on page 69

Validate a wildcard specification

To confirm that a wildcard specification will produce the specific list of tables that you want to replicate, issue the **verify config** command in **sp_ctrl** before you activate the configuration. This command produces a list of the objects that SharePlex will capture and replicate, as well as any problems that occurred. For more information about this command, see the SharePlex Reference Guide.

Examples of valid wildcard specifications

Example 1: The following wildcard specification directs SharePlex to activate all tables owned by **scott**, where the table name is like **prod%** except if the table name is like **%temp%**. All tables that match this description are replicated to tables of the same names on the target in the **hal** schema. Note that SharePlex automatically upshifts the names, so that it actually activates all tables where the table name is like 'PROD%' but not like '%TEMP%'.

```
Datasource:o.sidA
expand scott.prod% not (%temp%)    hal.%    sysa@o.sidB
```

Example 2: The following example shows how you can specify special handling for one of the tables in a wildcarded specification, in this case the **photo** table. All tables but **photo** are routed through the default post queue. The separate entry for the **photo** table overrides the wildcarded entry and processes the **photo** table through a named post queue. For more information, see Configure Named Post Queues on page 113.

```
Datasource:o.sidA
cust.%         cust.%         hostB@o.oraB
cust.photo     cust.photo     hostB:lobQ@o.oraB
```

The following are additional examples of valid wildcard specifications

```
Datasource:o.sidA
expand scott.%test%    scott.%    sysa@o.sidB
```

```
Datasource:o.sidA
expand scott.%t__t%    fred.%    sysa@o.sidB
```

```
Datasource:o.sidA
expand scott.% not (spo%, gen%, prodct)    scott.%     sysa@o.sidB

Datasource:o.sidA
expand scott.prod% not (%temp%)    hal.%    sysa@o.sidB
```

Examples of invalid wildcard specifications

The following example contains a wildcarded schema, which is not permitted.

```
Datasource:o.sidA
expand rob%.%test%    scott.%    sysa@o.sidB
```

The following example contains a partially wildcarded target object name, which is not permitted.

```
Datasource:o.sidA
expand scott.%test%    scott.%obj%    sysa@o.sidB
```

Use Wildcards to Specify Multiple Tables for PostgreSQL

You can use wildcard characters to specify multiple tables of a schema in one entry of the configuration file. SharePlex replicates any tables that satisfy the wildcard, except those that you explicitly exclude.

NOTE: Only table names can be wildcarded. Schema names cannot be wildcarded.

Requirements and limitations of wildcard support

The schemas that contain wildcarded table names must exist on the source and target before the configuration is activated.

Supported wildcard syntax

SharePlex supports the following PostgreSQL wildcards:

- Percent (%) wildcard to specify a string. (See the Examples on page 90)
- Underscore (_) wildcard to specify a single-character.
- For table names that contain a percent sign or an underscore character (for example **emp_salary**), SharePlex recognizes the backslash (\) as an escape character to denote the character as a literal, not a wildcard.

Specify wildcarded names in the configuration file

Use this template for help when specifying a wildcarded name in the configuration file.

Configuration with wildcarded table names

datasource_specification

expand *src_schema.wildcard_name* [**not** (*list*)] *tgt_schema.wildcard_name* *routing_map*

Description of syntax elements

Component	Description
expand	Indicates that the specification contains wildcard characters that must be expanded. When SharePlex detects the **expand** keyword, it queries the database for all tables that match the criteria in the wildcard specification. Without this

Component	Description
	required keyword, the wildcard characters are assumed to be part of an explicit table name, and no wildcard expansion is performed. **NOTE:** Leave a space between **expand** and the start of the source table specification.
src_schema.*wildcard_name*	• *src_schema* is the schema of the source tables. Schema names cannot be wildcarded. If wildcards are used in the schema name, SharePlex assumes that they are part of the schema name. • *wildcard_name* is the wildcarded name of the source tables. **PostgreSQL:** The names of the target tables must be identical to those of the source tables, but the tables may belong to different schemas.
not (*list*)	An exclusion list that defines tables to omit from the wildcard expansion. Use this option to exclude tables that you do not want to be replicated. **NOTE:** This **not** keyword does not have the same meaning as the SQL wildcard **NOT** operator. • The **not** keyword and parentheses are required elements. • *list* is a comma-separated list of tables owned by the same schema, either wildcarded or explicit. Example: **not (spo%, gen%, product).** Leave a space before and after the **not** keyword. A space is allowed after each comma in the list. **NOTE:** If an table that satisfies a wildcard is listed elsewhere in the configuration file, that entry overrides the processing or routing specified in the wildcarded entry. In this case, a **not** clause is *not* needed. See the **Examples**.
tgt_schema.*wildcard_name*	• *tgt_schema* is the schema of the target tables. • *wildcard_name* is the wildcarded name of the target tables. The target specification must be in the form of *schema*.%. Partially expanded target wildcarded names are not supported, such as *schema.tab*%.
routing_map	Any valid routing map.

Validate a Wildcard Specification

To confirm that a wildcard specification will produce the specific list of tables that you want to replicate, issue the **verify config** command in **sp_ctrl** before you activate the configuration. This command produces a list of the tables that SharePlex will capture and replicate, as well as any problems that occurred. For more information about this command, see SharePlex Reference Guide.

Examples

Examples of valid wildcard specifications

Example 1: The following wildcard specification directs SharePlex to activate all tables owned by **scott**, where the table name is like **prod%** except if the table name is like **%temp%**. All tables that match this description are replicated to tables of the same names on the target in the **hal** schema.

```
Datasource:r.dbname
expand scott.prod% not (%temp%)    hal.%    hostB@r.dbname
```

Example 2: The following example shows how you can specify special handling for one of the tables in a wildcarded specification, in this case the **photo** table. All tables but **photo** are routed through the default post queue. The separate entry for the **photo** table overrides the wildcarded entry and processes the **photo** table through a named post queue.

```
Datasource:r.dbname
cust.%         cust.%         hostB@r.dbname
cust.photo     cust.photo     hostB:queuename@r.dbname
```

The following are additional examples of valid wildcard specifications for PostgreSQL to PostgreSQL replication:

```
Datasource:r.dbname
expand scott.%test%    scott.%    hostB@r.dbname
```

```
Datasource:r.dbname
expand scott.%t__t%    fred.%    hostB@r.dbname
```

```
Datasource:r.dbname
expand scott.% not (spo%, gen%, prodct)    scott.%    hostB@r.dbname
```

```
Datasource:r.dbname
expand scott.prod% not (%temp%)    hal.%    hostB@r.dbname
```

The following is an example of valid wildcard specifications for PostgreSQL to Oracle replication:

```
Datasource:r.dbname
expand "scott"."%test%"    "scott"."%"    hostB@o.target_dbname
```

The following is an example of valid partial wildcard specifications for PostgreSQL to SQL Server replication:

```
Datasource:r.dbname
expand scott.%test%    scott.%test%    hostB@r.target_dbname
```

Examples of invalid wildcard specifications

The following example contains a wildcarded schema, which is not permitted.

```
Datasource:r.dbname
expand rob%.%test%    scott.%    hostB@r.dbname
```

The following example contains a partially wildcarded target table name, which is not permitted.

```
Datasource:r.dbname
expand scott.%test%    scott.%obj%    hostB@r.dbname
```

Define a Unique Key

If a table was not created with a primary or unique key, you can specify columns to use as a key when you specify the object in the configuration file. SharePlex uses the specified columns as a unique key in its WHERE clause to locate target rows for posting.

NOTES:

- Without a primary or unique key, SharePlex uses all of the columns of a table (or all of the columns in a column partition) as a key, which slows replication performance.
- When a key definition is specified for a table that has a PRIMARY or UNIQUE key, the key definition overrides the defined key. This can be useful if you do not want any of the existing keys to be used by SharePlex.

Define a unique key - Oracle to Oracle

The columns that you specify as a key must meet the following criteria:

- They cannot be LONG or LOB columns.
- They must be able to uniquely identify a row. Otherwise, replication could return out-of-sync errors or post to incorrect target rows.
- They must be part of the column partition if the table is configured for vertically partitioned replication. When using the **exclude** column notation in vertical partitioning, the excluded columns cannot be used in the key definition. For more information, see Configure Partitioned Replication on page 118.
- Include the columns in a supplemental log group. Otherwise, SharePlex must query the database for their values.
- Create an index on the target table and add the index to the SharePlex **hints** file, located in the variable-data directory, which directs the Post process to use the index.

Syntax for key definition

To create a key definition, type a space after the source object and use the following syntax, including the parentheses.

src_owner.*table* **!key (***column_list***)**

where:

- **!key** is a required keyword.
- *column_list* is a list of columns to include in the key. Separate column names with commas. A space after the comma is optional.

datasource_specification

src_owner.*table* **!key (***col_name*, *col2_name, ...***)** *tgt_owner.table* *host***@o.***SID*

Example

```
Datasource:o.ora1
scott.tab !key(name,ID)    scott.tab2    sysB@oraB
```

Define a unique key - PostgreSQL to PostgreSQL

The columns that you specify as a key must meet the following criteria:

- A unique key cannot be TEXT, BYTEA, JSON, JSONB, CHAR with more than 2000 characters, VARCHAR without size or more than 4000 characters.
- They must be able to uniquely identify a row. Otherwise, replication could return out-of-sync errors or post to incorrect target rows.
- They must be part of the column partition if the table is configured for vertically partitioned replication. When using the exclude column notation in vertical partitioning, the excluded columns cannot be used in the key definition. For more information, see Configure Partitioned Replication.
- Create an index on the target table, it directs the Post process to use the index.

Syntax for key definition

To create a key definition, type a space after the source object and use the following syntax, including the parentheses.

src_schema.table **!key (***column_list***)**

where:

- **!key** is a required keyword.
- *column_list* is a list of columns to include in the key. Separate column names with commas. A space after the comma is optional.

datasource_specification
src_schema.table **!key (***col_name*, *col2_name, ...***)** *tgt_schema.table* *host***@r.***dbname*

Example

```
Datasource:r.dbname
scott.tab !key(name,ID)    scott.tab2    sysB@dbname
```

Define a unique key - PostgreSQL to Oracle

The columns that you specify as a key must meet the following criteria:

- A unique key cannot be text, char with more than 2000 characters, varchar without size or more than 4000 characters.
- They must be able to uniquely identify a row. Otherwise, replication could return out-of-sync errors or post to incorrect target rows.
- They must be part of the column partition if the table is configured for vertically partitioned replication. When using the **exclude** column notation in vertical partitioning, the excluded columns cannot be used in the key definition. For more information, see Configure Partitioned Replication on page 118.
- Create an index on the target table and add the index to the SharePlex **hints** file, located in the variable-data directory, which directs the Post process to use the index.

Syntax for key definition

To create a key definition, type a space after the source object and use the following syntax, including the parentheses.

src_schema.table **!key (***column_list***)**

where:

- **!key** is a required keyword.
- *column_list* is a list of columns to include in the key. Separate column names with commas. A space after the comma is optional.

datasource_specification		
src_schema.table **!key (***col_name*, *col2_name, ...***)**	*tgt_owner.table*	*host***@o.***SID*

Example

```
Datasource:r.dbname
"scott"."tab" !key(name,ID)    "scott"."tab2"    sysB@o.oraB
```

Filter DML Operations for Oracle Database

You can configure SharePlex to filter out the following DML from replication when wildcarding is being used.

- Oracle DML type (INSERT, UPDATE, DELETE)
- DML related to Oracle sequences and Oracle SQL*Loader direct-path loads.

Filter out a DML type

You can configure SharePlex to filter any type of DML operation so that the operation is not replicated to the target table. DML filtering is compatible with most other SharePlex configuration syntax.

Configure a DML filter

To configure a DML filter, add the following syntax to the source table specification. Leave a space between the table specification and the filter specification. You can specify multiple operation types to filter. Any additional syntax for other features, such as a column list or key definition, must follow the DML filter specification.

!dml(*DML_type*[**,***DML_type*][**,...**])

Where *DML_type* is one of the following:

***DML_type* input**	**Operation type**
i	INSERT
d	DELETE
u	UPDATE

Examples

Example 1

The following example filters DELETE operations from being replicated to the target table.

```
Datasource:o.ora
scott.emp !dml(d)              scott.emp          prodsys@o.sysdb
```

Example 2

The following example filters DELETEs and INSERTs so that only UPDATEs are replicated to the target table. This example also shows how a DML filter is compatible with a column mapping specification.

```
Datasource:o.ora
scott.stock !dml(d,i) (ID, name)       scott.stock (SKU, prod)      sys2@o.sysdb
```

View current DML filters

Use the **verify config** command to view the DML that is being filtered for each table in the configuration file. This command can be used for an active or inactive configuration file.

```
sp_ctrl>  verify config myconfig
7: "SCOTT"."EMP" "SCOTT"."EMP" prodsys@o.proddb
Filter out >>>>> DELETES
Unique Key : (EMPLOYEE_ID)
```

Restrictions

- The **copy** and **compare** commands do not support tables that include DML filtering in their specifications.
- If there are multiple specifications of a source table in the configuration file, the DML filter specification must be identical for all of them. Multiple specifications of the same source table occur in the following instances:
  - Routing a source table to multiple targets without using a compound routing map. For more information, see Routing Specifications in a Configuration File on page 69.
  - Combining full-table replication with horizontally partitioned replication. For more information, see Configure Horizontally Partitioned Replication on page 118.

Filter DML related to specific Oracle objects from replication

You can prevent SharePlex from replicating sequences and SQL*Loader direct-path loads. By default the replication of these objects is enabled.

Filter out this object	Set this parameter	Value
Sequences	SP_OCT_REPLICATE_SEQUENCES	0
SQL*Loader direct-path loads	SP_OCT_REPLICATE_DLOAD	0

Filter DML Operations for PostgreSQL Database

You can configure SharePlex to filter out the PostgreSQL DML type (INSERT, UPDATE, DELETE) from replication when wildcarding is being used.

Filter out a DML type

You can configure SharePlex to filter any type of DML operation so that the operation is not replicated to the target table. DML filtering is compatible with most other SharePlex configuration syntax.

Configure a DML filter

To configure a DML filter, add the following syntax to the source table specification. Leave a space between the table specification and the filter specification. You can specify multiple operation types to filter. Any additional syntax for other features, such as a column list or key definition, must follow the DML filter specification.

!dml(*DML_type*[**,***DML_type*][**,...**])

Where *DML_type* is one of the following:

***DML_type* input**	**Operation type**
i	INSERT
d	DELETE
u	UPDATE

Examples

Example 1

The following example filters DELETE operations from being replicated to the target table.

```
Datasource:r.dbname
scott.emp !dml(d)              scott.emp          prodsys@r.dbname
```

Example 2

The following example filters DELETEs and INSERTs so that only UPDATEs are replicated to the target table. This example also shows how a DML filter is compatible with a column mapping specification.

```
Datasource:r.dbname
scott.stock !dml(d,i) !key (EMPLOYEE_ID)      scott.stock          sys2@r.dbname
```

View current DML filters

Use the **verify config** command to view the DML that is being filtered for each table in the configuration file. This command can be used for an active or inactive configuration file.

```
sp_ctrl> verify config myconfig detail
7: "SCOTT"."EMP" "SCOTT"."EMP" prodsys@o.proddb
Filter out >>>>> DELETES INSERTS
Unique Key : (EMPLOYEE_ID)
```

Restriction

If there are multiple specifications of a source table in the configuration file, the DML filter specification specified in the last line for the table is considered.

Map Source and Target Columns

When source and target column names are different, you can specify an explicit column mapping in the configuration file, to ensure that Post applies row data to the correct target columns.

Guidelines for using column mapping

- To use column mapping, you must map every column in a source table to a column in the target table, even if only some source and target names are different. When some columns are mapped but not others, the entry is considered to be a column partition for vertically partitioned replication, and data changes for non-listed columns are not replicated.
- If the spelling case is different between the source and target column names, enclose them within quotes.
- You can use horizontally partitioned replication and column mapping for the same source table, but you cannot combine column mapping with vertically partitioned replication.
- A target table can have more columns than the source table, but there must at least be a target column for every source column.
- ALTER TABLE to add a column to a table that is configured with column mapping is not supported.

Configure column mapping

The following syntax creates a column map. For more information, see Configure SharePlex to Replicate Data on page 60.

```
datasource_specification
src_owner.table  (src_col,src_col,...)  tgt_owner.table  (tgt_col,tgt_col,...)  routing_map
```

Configuration component	Description
datasource_specification	The datasource specification. For more information, see Database Specifications in a Configuration File on page 67.
src_owner.table and tgt_owner.table	The specifications for the source and target tables, respectively. For more information, see Create a Configuration File on page 62.
(src_col,src_col,...)	A list of the source columns. Follow these rules to specify a column list: • A column list must be enclosed within parentheses. • Separate each column name with a comma. A space after the comma is optional. • The maximum length of a column list is 174820 bytes (the maximum line length allowed in a configuration file).
(*tgt_col*,*tgt_col*,...)	A list of the target columns.

Configuration component	Description
	• List the target columns in the same logical order as their corresponding source columns. This is required regardless of the actual order of the target columns in the table, so that SharePlex builds the correct map in the object cache. For example, a change to the second column in the source list is replicated to the second column in the target list. • The syntax rules for the source list also apply to the target list.
routing_map	The routing map. For more information, see Routing Specifications in a Configuration File on page 69.

Configuration example

This example contains no case-sensitive columns.

```
Datasource o.oraA
sales.prod (ID,name,vendor) mfg.prod (UPC,product,supplier)     sysB@o.oraB
```

This example contains case-sensitive columns.

```
Datasource o.oraA
sales.prod
(ID,"name",vendor)                mfg.prod (UPC,"product",supplier)   sysB@o.oraB
```

Build a Configuration File using a Script

SharePlex provides the following scripts to automate the building of a configuration file to specify Oracle source objects.

- **config.sql**: configure all tables and optionally all sequences in the database.
- **build_config.sql**: configure multiple or all tables in a schema

Supported databases

Oracle

Use config.sql

The **config.sql** script enables you to build a configuration that lists all of the tables, and optionally all of the sequences, in all of the schemas of a database. This script saves time when establishing a high-availability replication strategy or other scenario where you want the entire database to be replicated to an identical secondary database.

Conditions for using config.sql

- Source and target table names must be the same.
- The script does not configure objects in the SYS, SYSTEM, and SharePlex schemas. These schemas cannot be replicated since they are system and/or instance-specific.
- The script does not support partitioned replication. You can use the **copy config** command to copy the configuration file that the script builds, then use the **edit config** command to add entries for tables that use partitioned replication. Activate the new configuration file, not the original one.
- You can use the **edit config** command to make any other changes as needed after the configuration is built.

To run config.sql:

1. Change directories to the **config** sub-directory of the SharePlex variable-data directory. The **config.sql** script puts configurations in the current working directory, and SharePlex configurations must reside in the **config** sub-directory.

   cd /*vardir*/**config**

2. Log onto SQL*Plus as SYSTEM.
3. Run **config.sql** using the full path from the **util** sub-directory of the SharePlex product directory.

   @ /*proddir*/**util/config.sql**

Refer to the following table when following the prompts:

Prompt	What to enter
Target machine	The name of the target machine, for example SystemB.
Source database SID	The ORACLE_SID of the source (primary) Oracle instance, for example oraA. Do not include the **o.** keyword. The ORACLE_SID is case-sensitive.
Target database SID	The ORACLE_SID of the target (destination) Oracle instance, for example oraB. Do not include the **o.** keyword. The ORACLE_SID is case-sensitive.
Replicate sequences	Enter **y** to replicate sequences or **n** not to replicate sequences.
SharePlex oracle username	The name of the SharePlex user in the source database. This entry prevents the SharePlex schema from being replicated, which would cause replication problems. If a valid name is not provided, the script fails.

NOTE: The name assigned by SharePlex to the configuration is **config.file**. If you run the script again to create another configuration file, it overwrites the first file. To preserve the original file, rename it before you create the second one.

Next steps:

- If any tables or owners are case-sensitive, open the configuration file with the **edit config** command in **sp_ctrl**, then use the text editor to enclose case-sensitive table and owner names within double-quote marks, for example “scott”.“emp”. The script does not add the quote marks required by Oracle to enforce case-sensitivity.

  ```
  sp_ctrl> edit config filename
  ```

- To ensure that the configuration is in the correct location, issue the **list config** command. If the name of the configuration is not shown, it was created in the wrong directory. Find the file and move it to the **config** sub-directory of the variable-data directory.

  ```
  sp_ctrl> list config
  ```

Use build_config.sql

The **build_config.sql** script enables you to build a configuration that contains multiple (or all) tables in a schema. It is an interactive script that prompts for each component of the configuration step by step. Instead of entering the information for each object and the routing individually, you can use a wildcard to select certain tables at once, or you can select all of the tables in the schema.

Conditions for using build_config.sql

- Source and target table names must be the same.
- The script does not support sequences. Before you activate the configuration that the script builds, you can use the **edit config** command in **sp_ctrl** to add entries for sequences.
- The script does not support partitioned replication. You can use the **copy config** command to copy the configuration that the script builds, then use the **edit config** command to add entries for the tables that use partitioned replication. Activate the new configuration, not the original.

- The script does not configure objects in the SYS, SYSTEM, and SharePlex schemas. These schemas cannot be replicated since they are system and/or instance-specific.
- You can run **build_config.sql** for different schemas, then combine those configurations into one configuration by using a text editor. Make certain to eliminate all but one **Datasource:o.SID** line, which is the first non-commented line of the file. Do not move the file out of the **config** sub-directory.
- You can use the **edit config** command to make any other changes as needed after the configuration is built.

To run build_config.sql:

1. Change directories to the **config** sub-directory of the SharePlex variable-data directory. The **build_config.sql** script puts configurations in the current working directory, and SharePlex configurations must reside in the **config** sub-directory.

   cd /*vardir*/**config**

2. Log onto SQL*Plus as SYSTEM.
3. Run **build_config.sql** using the full path from the **util** sub-directory of the SharePlex product directory.

   @ /*proddir*/**util/build_config.sql**

Refer to the following table when following the prompts.

Prompt	What to enter
Target machine	The name of the target machine, for example SystemB.
Source database SID	The ORACLE_SID of the source (primary) Oracle instance, for example oraA. Do not include the **o.** keyword. The ORACLE_SID is case-sensitive.
Target database SID	The ORACLE_SID of the target (destination) Oracle instance, for example oraB. Do not include the **o.** keyword. The ORACLE_SID is case-sensitive.
Owner of the source database tables	The owner of the source tables.
Owner of the target database tables	The owner of the target tables.
Table name to include (blank for all)	Do one of the following: • Press **Enter** to accept the default, which selects all tables that belong to the source owner. • Enter a wildcard (%) character and a string to select certain tables, for example **%e_salary%**. • Enter an individual table name.
Name of the output file to create	A name for the configuration. The script gives the file a **.lst** suffix, for example **Scott_config.lst**.

Next steps:

- If any tables or owners are case-sensitive, open the configuration with the **edit config** command in **sp_ctrl**, then use the text editor to enclose case-sensitive table and owner names within double-quote marks, for example “scott”.“emp”. The script does not add the quote marks required by Oracle to enforce case-sensitivity.

  ```
  sp_ctrl> edit config filename
  ```

- To ensure that the configuration is in the correct location, issue the **list config** command. If the name of the configuration is not shown, it was created in the wrong directory. Find the file and move it to the **config** sub-directory of the variable-data directory.

  ```
  sp_ctrl> list config
  ```

7

Configure Replication to and from a Container Database

SharePlex supports replication to and from Pluggable Databases (PDB) in Oracle multitenant container databases (CDB). This support is available on Unix and Linux platforms only.

Contents

Configure Capture and Delivery

SharePlex can replicate data from one PDB to:

- another PDB in the same CDB
- a PDB in a different CDB
- a regular (non-PDB) target

SharePlex can replicate data from a regular source database to a PDB in a target Oracle CDB.

In one configuration file, you can replicate to any number of target PDBs in the same CDB or a different CDB.

To capture from a PDB:

- In the configuration file, specify the **TNS alias** of a PDB as the datasource. For example, if the TNS alias is **pdb1**, the datasource specification is:

  Datasource: o.pdb1

- You can replicate from as many pluggable databases (PDBs) in the same CDB as desired: Create a separate configuration file for each PDB. Because each PDB is a different datasource, all configurations can be active at the same time.
- If replicating from more than one PDB on a system, use named export queues to separate the data streams from each one. This allows you to issue SharePlex commands that affect configurations, such as purge config or abort config, for one configuration without affecting the other configurations. For more information, see Configure Named Export Queues on page 107

To replicate to a PDB:

Specify the **TNS alias** of the target PDB in the routing map, as shown in the following example where **pdb2** is the target:

sys02@o.pdb2

PDB configuration examples

Example 1: This example shows two configuration files, one replicating from **pdb1** and the other replicating from **pdb2**, both replicating data to **pdb3**.

```
Datasource: o.pdb1
hr.emp    hr2.emp2    sys02@o.pdb3

Datasource: o.pdb2
sales.cust    sales2.cust2    sys02@o.pdb3
```

Example 2: This example shows one configuration file replicating from **pdb1** to **pdb2** and **pdb3**, both targets being on different systems.

8

Configure Named Queues

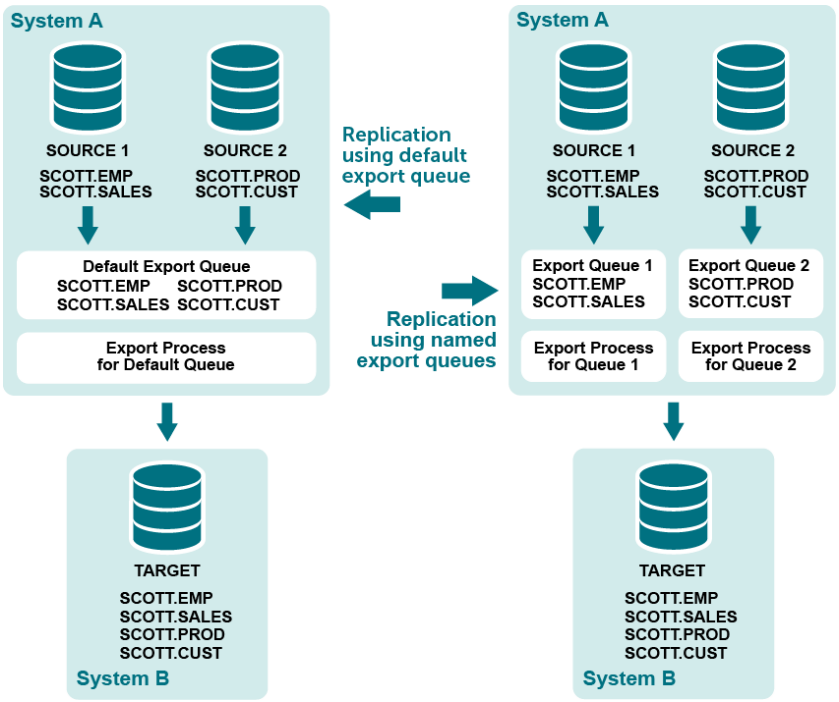
This chapter contains instructions for using the advanced SharePlex configuration options of *named queues*. These options provide an additional level of flexibility to divide and parallelize data to meet specific processing and routing requirements. Before proceeding, make certain you understand the concepts and processes in Configure SharePlex to Replicate Data.

Contents

Configure Named Export Queues

A named export queue is an optional, user-defined export queue that is attached to its own Export process. SharePlex creates each named Export queue and associated Export process in addition to the default export queue-process pair. When SharePlex creates a named export queue-process pair, it also creates a dedicated Import process, post queue, and Post process on the target to contain that data stream.

You direct SharePlex to create one or more named export queues when you create your configuration file. Any data that is not configured for processing through a named export queue is processed through the default export queue.



Supported sources and targets

- PostgreSQL to PostgreSQL, Oracle, SQL Server, and Kafka
- Oracle to all targets

Benefits of named export queues

Use named export queues to isolate the replication of:

- **Individual configurations:** By default, SharePlex sends data from all active configurations through one export queue-process pair per target system, but the use of named Export queues enables you to separate each of those replication streams into its own export queue and Export process. In this way, you ensure that **purge config** or **abort config** commands that are issued for one configuration do not affect any of the others.
- **Selected database objects:** You can use a named export queue to isolate certain objects such as tables that contain LOBs. Because each named export queue has its own Import process, post queue, and Post process on the target, you are able to isolate the data the entire way from source to target. For more information about the benefits of named post queues, see Configure Named Post Queues on page 113.

Additional benefits:

- You can stop the Export or Import process for one data stream, while allowing the others to continue processing.
- You can set SharePlex parameters to different settings for each export queue-process pair. This enables you to tune the performance of the Export processes based on the objects replicating through each one.

Considerations when using named export queues

- Make certain that each queue name is unique.
- You can combine named export queues with default export queues. Tables in the configuration with a standard routing map (*targetsys@database_spec* without a named queue specification) are replicated through a default export queue.
- All tables with referential integrity to one another must be in the same export queue.
- SharePlex has a maximum number of allowed queues. For more information, see Routing Specifications in a Configuration File on page 69.

Configure a named export queue: Oracle to all targets

Use the following syntax to define a routing map that includes a named export queue.

source_host:*export_queuename***target_host*[@*database*]

Configuration with named export queue in routing map

Datasource: o.*SID*		
src_owner.*table*	*tgt_owner*.*table*	*source_host*:*export_queue***target_host*[@*database_specification*]

Routing component	Description
source_host	The name of the source system.
export_queue	The name of the export queue. Queue names are case-sensitive on all platforms. Use one word only. Underscores are permissible, for example: **sys1:export_q1*sys2@o.myora**
target_host	The name of the target system.
database specification	One of the following for the datasource: **o.***oracle_SID* **r.***database_name* One of the following if the target is a database: **o.***oracle_SID* **o.***tns_alias* **o.***PDBname* **r.***database_name* **c.***oracle_SID*

NOTES:

- Allow no spaces between any components in the syntax of the routing map.
- For more information about the components of a configuration file, see Configure SharePlex to Replicate Data on page 60.

Examples

The following configuration files show two different datasources that are being replicated to two different databases on the same target system. Each datasource is routed through a named export queue.

Datasource:o.oraA

scott.emp	scott.emp	sysA:QueueA*sysB@o.oraC
scott.sales	scott.sales	sysA:QueueA*sysB@o.oraC

Datasource:o.oraB

scott.prod	scott.prod	sysA:QueueB*sysB@o.oraD
scott.cust	scott.cust	sysA:QueueB*sysB@o.oraD

The following shows how to separate a table that contains LOBs from the rest of the tables by using named export queues.

Datasource:o.oraA

scott.cust	scott.cust	sysA:QueueA*sysB@o.oraC
scott.sales	scott.sales	sysA:QueueA*sysB@o.oraC
scott.prod	scott.prod	sysA:QueueA*sysB@o.oraC
scott.emp_LOB	scott.emp_LOB	sysA:QueueB*sysB@o.oraC

Alternatively, you could simply define a named export queue for the LOB table and allow the remaining tables to be processed through the default export queue.

Datasource:o.oraA

scott.cust	scott.cust	sysB@o.oraC
scott.sales	scott.sales	sysB@o.oraC
scott.prod	scott.prod	sysB@o.oraC
scott.emp_LOB	scott.emp_LOB	sysA:lobQ*sysB@o.oraC

Configure a named export queue for PostgreSQL

Use the following syntax to define a routing map that includes a named export queue.

*source_host:export_queuename*target_host[@database]*

Supported targets

PostgreSQL, Oracle, SQL Server, and Kafka

Configuration with named export queue in routing map

Datasource:r.*dbname*		
src_schema.table	*tgt_schema.table*	*source_host:export_queue*target_host[@database_specification]*

Routing component	Description
source_host	The name of the source system.
export_queue	The name of the export queue. Queue names are case-sensitive on all platforms. Use one word only. Underscores are permissible, for example: **sys1:export_q1*sys2@r.dbname**
target_host	The name of the target system.
database specification	**r.***database_name*

NOTE: Allow no spaces between any components in the syntax of the routing map.

Examples

The following configuration files show two different datasources that are being replicated to two different databases on the same target system. Each datasource is routed through a named export queue.

Datasource:r.dbnameA

scott.emp	scott.emp	sysA:QueueA*sysB@r.dbnameC
scott.sales	scott.sales	sysA:QueueA*sysB@r.dbnameC

Datasource:r.dbnameB

scott.prod	scott.prod	sysA:QueueB*sysB@r.dbnameD
scott.cust	scott.cust	sysA:QueueB*sysB@r.dbnameD

The following shows how to separate a table that contains LOBs from the rest of the tables by using named export queues.

Datasource:r.dbnameA		
scott.cust	scott.cust	sysA:QueueA*sysB@r.dbnameC
scott.sales	scott.sales	sysA:QueueA*sysB@r.dbnameC
scott.prod	scott.prod	sysA:QueueA*sysB@r.dbnameC
scott.emp_LOB	scott.emp_LOB	sysA:QueueB*sysB@r.dbnameC

Alternatively, you could simply define a named export queue for the LOB table and allow the remaining tables to be processed through the default export queue.

Datasource:r.dbnameA		
scott.cust	scott.cust	sysB@r.dbnameC
scott.sales	scott.sales	sysB@r.dbnameC
scott.prod	scott.prod	sysB@r.dbnameC
scott.emp_LOB	scott.emp_LOB	sysA:lobQ*sysB@r.dbnameC

How to identify named export queues

You can view named export queues through `sp_ctrl`:

- Use the **qstatus** command to view all queues on a system.
- Use the **show** command to view all Export processes with their queues.

See the SharePlex Reference Guide for more infomation about theses commands.

Configure Named Post Queues

A named post queue is an optional component of the routing map in the configuration file. A named post queue is a user-defined post queue with its own Post process, which together operate in parallel to the default post queue and Post process. You can define one or more named post queue-process pairs to establish a set of parallel Post replication streams.

Supported sources and targets

- PostgreSQL to PostgreSQL, Oracle, SQL Server, and Kafka
- Oracle to all targets

Benefits of named post queues

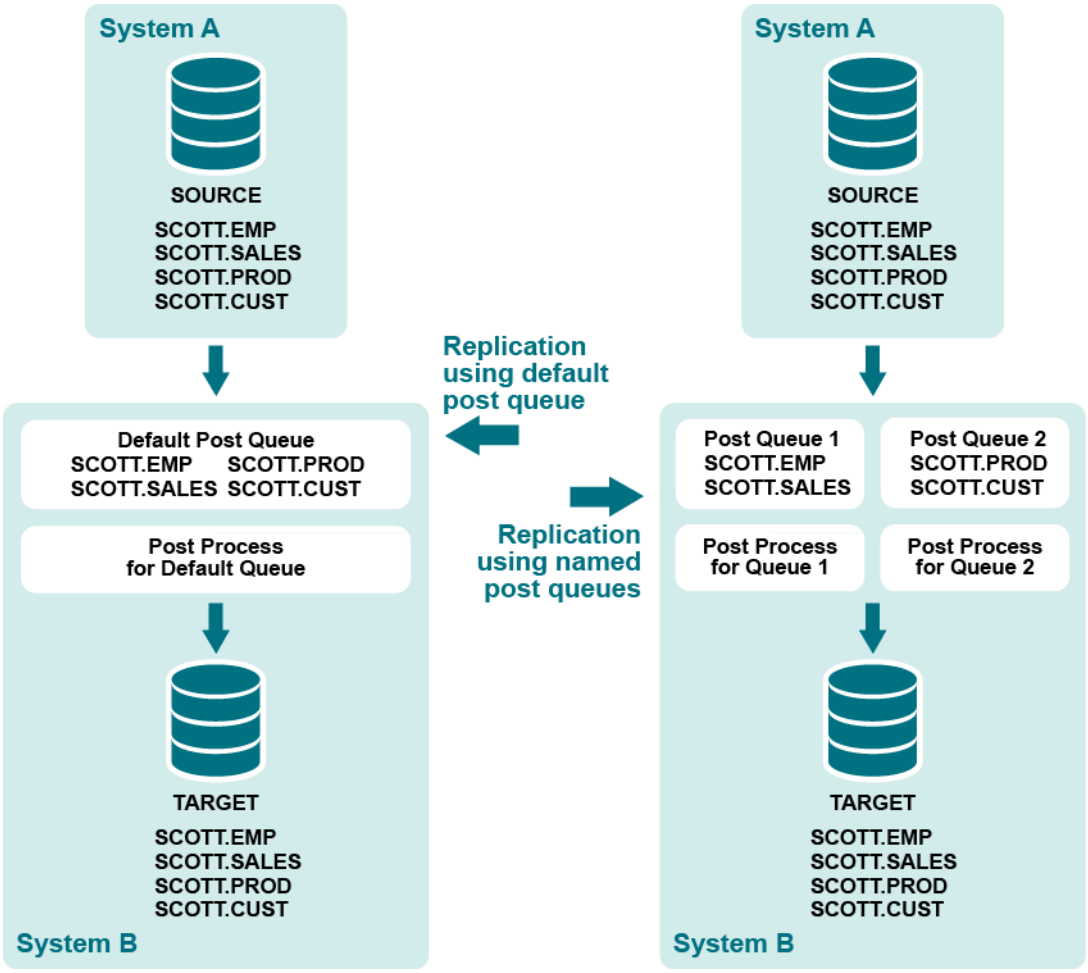
You can use named post queues to isolate data from different tables into two or more separate Post streams. By using named post queues, you can improve posting performance by isolating objects such as the following that cause processing bottlenecks:

- large tables
- objects that have LOB columns. Named post queues are recommended for objects that contain LOBs.
- objects that involve large transactions.
- any objects whose operations you want to isolate.

Process the remaining objects through additional named post queues, or use the default post queue. Objects in the configuration file with a standard routing map (*host@target*) are replicated through a default post queue.

You can use horizontal partitioning to divide the rows of very large tables into separate named post queues as an added measure of parallelism.

You can set SharePlex parameters to different settings for each queue-process pair. This enables you to tune the performance of the Post processes based on the objects replicating through each one.



Considerations when using named post queues

- The **analyze config** command can help you determine how to organize your tables into named queues, based on any dependencies they have and their individual transactional activity. Run the command over a period of time that captures typical database activity, and then view the command output.
- Assign each post queue a unique name.
- If objects are linked by relational dependencies, process all of those objects through the same named post queue. If interdependent objects are not replicated through the same post queue, parent and child operations may be applied out of order and will cause database errors. As an alternative to processing interdependent objects through the same queue, you can disable their referential constraints on the target. This may be acceptable, because the constraints are satisfied on the source system and then replicated to the target.

- When using multiple Posts, the target objects might not be changed in the same order as the corresponding source objects, possibly causing the target database to be inconsistent with the source database at any given point in time.
- If you implement named post queues for objects in an active configuration (thus changing the routing) SharePlex locks those objects to update its internal directions.
- SharePlex has a maximum number of allowed queues. For more information, see Routing Specifications in a Configuration File on page 69.

Configure a named post queue: Oracle to all targets

If you are using named export queues, SharePlex creates a named post queue-process pair for each one by default. If you are not using named export queues, use the following syntax to define a named post queue in the configuration file by adding the **:***queue* component to the routing map:

*host***:***queue***@***target*

Configuration with named post queue in routing map

Datasource: o.*SID*

src_owner.table *tgt_owner.table* *host*:*queue*[**@***database_specification*]

Routing component	Description
host	The name of the target system.
queue	The unique name of the post queue. Queue names are case-sensitive on all platforms. One word only. Underscores are permissible, for example: **sys2:post_q1@o.myora**
database_specification	One of the following for the datasource: **o.***oracle_SID* One of the following if the target is a database: • **o.***oracle_SID* **o.***tns_alias* **o.***PDBname* **r.***database_name* **c.***oracle_SID*

NOTES:

- Allow no spaces between any components in the syntax of the routing map.
- For more information, see Configure SharePlex to Replicate Data on page 60.

Examples

The following configuration creates one post queue named **Queue1** that routes data from table **scott.emp** and another post queue named **Queue2** that routes data from table **scott.cust**.

Datasource:o.oraA		
scott.emp	scott.emp	sysB:Queue1@o.oraC
scott.cust	scott.cust	sysB:Queue2@o.oraC

The following shows how a named post queue is specified when you are routing data in a pass-through configuration using an intermediary system. For more information, see Configure Replication to Share or Distribute Data on page 155.

Datasource:o.oraA		
scott.emp	scott.emp	sysB*sysC:Queue1@o.oraC

Configure a named post queue for PostgreSQL

If you are using named export queues, SharePlex creates a named post queue-process pair for each one by default. If you are not using named export queues, use the following syntax to define a named post queue in the configuration file by adding the **:***queue* component to the routing map:

*host***:***queue***@***target*

Supported targets

PostgreSQL, Oracle, SQL Server, and Kafka

Configuration with named post queue in routing map

Datasource:r.*dbname*		
src_schema.table	*tgt_schema.table*	*host***:***queue*[**@***database_specification*]

Routing component	Description
host	The name of the target system.
queue	The unique name of the post queue. Queue names are case-sensitive on all platforms. One word only. Underscores are permissible, for example: **sys2:post_q1@r.dbname**
database_specification	**r.***database_name*

NOTE: Allow no spaces between any components in the syntax of the routing map.

Examples

The following configuration creates one post queue named **Queue1** that routes data from table **scott.emp** and another post queue named **Queue2** that routes data from table **scott.cust**.

Datasource:r.dbname		
scott.emp	scott.emp	sysB:Queue1@r.dbname
scott.cust	scott.cust	sysB:Queue2@r.dbname

The following shows how a named post queue is specified when you are routing data in a pass-through configuration using an intermediary system.

Datasource:r.dbname		
scott.emp	scott.emp	sysB*sysC:Queue1@r.dbname

How to identify a named post queue

A named post queue is identified by the datasource (source of the data) and one of the following:

- the name of an associated named export queue (if the Import is linked to a named export queue)
- the user-assigned post-queue name (if the Import is linked to a default export queue).

You can view named post queues through **sp_ctrl**:

- Use the **qstatus** command to view all queues on a system.
- Use the **show** command to view all Post processes with their queues.

See the SharePlex Reference Guide for more infomation about theses commands.

9

Configure Partitioned Replication

This chapter contains instructions for using the advanced SharePlex configuration options of *horizontally partitioned* and *vertically partitioned* replication. These options provide an additional level of flexibility to divide, parallelize, and filter data to meet specific requirements. Before proceeding, make certain you understand the concepts and processes involved in creating configuration files.

Contents

Configure Horizontally Partitioned Replication

Use horizontally partitioned replication to divide the rows of a table into separate processing streams. You can use horizontally partitioned replication to:

- Replicate a subset of rows to a target, while retaining the rest of the rows in the source.
- Replicate different subsets of rows to different targets.
- Divide the replication of a source table into parallel Post queues for faster posting to the target table.

Supported sources and targets

- PostgreSQL to PostgreSQL, Oracle, SQL Server, and Kafka
- Oracle to all targets
- PGDB as a Service to PGDB as a Service
- PGDB as a Service to Oracle
- PGDB as a Service to PostgreSQL

Overview of horizontally partitioned replication: Oracle to all targets

To configure horizontally partitioned replication for a table, the steps are:

1. Define *row partitions* and link them to a *partition scheme*.
    - A row partition is a subset of rows in a source table that you want to replicate as a group.
    - A partition scheme is a logical container for row partitions.
2. Specify the name of the partition scheme in the SharePlex configuration file to include the partitions in replication.

Partition types

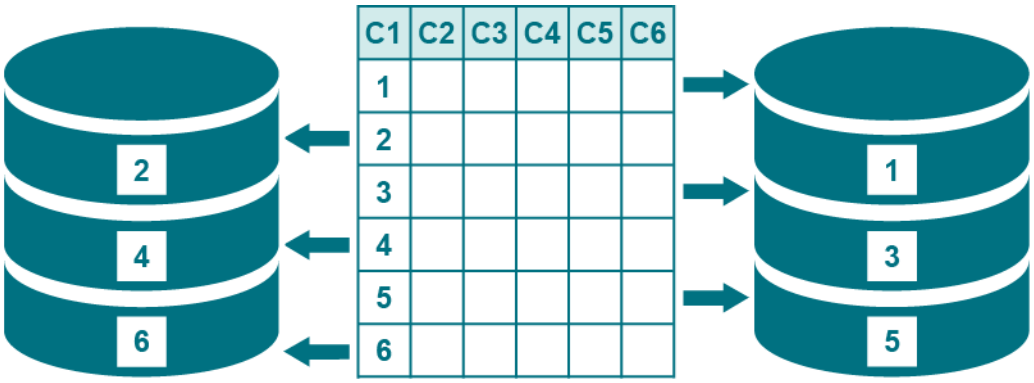
The row partitions in a partition scheme can be based on one of the following:

- **Columns**: A *column-based partition scheme* contains row partitions defined by a *column condition*. A column condition is a WHERE clause that defines a subset of the rows in the table.
- **Hash**: A *hash-based partition scheme* contains row partitions defined by a *hash value* that directs SharePlex to distribute rows evenly across multiple queues.

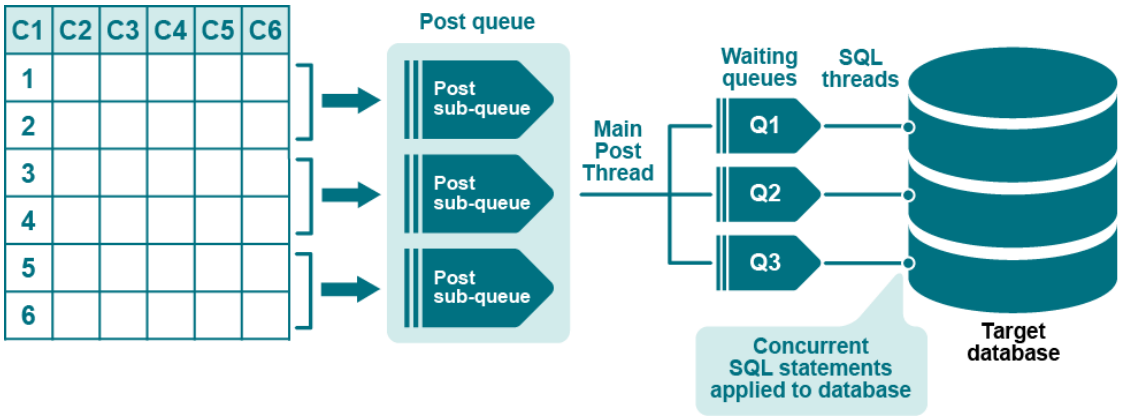
About column-based partition schemes

You can use row partitions based on column conditions for the following purposes:

- Use a single row partition to replicate only a subset of the rows of a table. For example, you can replicate only those rows where the value of the YEAR column is greater than 2014. The partition scheme in this case could be named "Since2014" or "Recent."
- Use multiple row partitions to divide the rows of a table so that each set of rows replicates to a different target. For example, a table named CORPORATE.SALES could have two row partitions named "East" and "West." The column conditions are defined accordingly, where the rows that satisfy REGION = EAST replicate to one location and the rows that satisfy REGION = WEST replicate to a different location. The partition scheme could be named "Sales_by_region."



- Use multiple row partitions to divide the rows of a table into parallel processing streams (parallel Export-Import-Post streams) for faster posting to a target table. For example, you can improve the flow of replication to a heavily updated target table. The use of column conditions for this purpose is appropriate only if the table contains a column that enables you to split the processing evenly among parallel Post processes.



About hash-based partition schemes

You can use row partitions based on a hash value to divide the rows of a table into parallel processing streams (parallel Export-Import-Post) for faster posting to a target table. The advantage of using a hash value over column conditions to create partitions is that the rows are divided evenly and automatically by SharePlex, without the need to reference table columns in WHERE clauses. However, unlike column-based partition schemes, you cannot use the SharePlex**compare** or **repair** commands for hash-based partition schemes.

Combine partitioned replication with full-table replication

You can combine horizontally partitioned and vertically partitioned replication for maximum control over how information is distributed.

For example:

- A corporate headquarters maintains a primary corporate database.
- Each of the corporation's four regional offices maintains its own database.
- Corporate headquarters uses vertically partitioned replication to share some column data of a table with the regional offices, but does not share any columns that contain sensitive data.
- The rows of the table are horizontally partitioned into four groups (East, West, North, South) for replication, so that each region receives only the record changes that apply to it.

Horizontally partitioned replication can be used in conjunction with full-table replication for the same table, for example to route groups of rows to different reporting systems and all rows to a backup system.

Limitations of use

Hash-based partitioning does not support the following:

- The **compare** and **repair** commands.
- Index-organized tables (IOT) and tables that contain LOBs or LONGs.
- Operations that delete or update a key value and then reinsert the same key value. This can cause unique constraint violations because of different rowids.
- Column-based partitioning on the same table.

Hash-based partitioning also does not support operations that cause rows to migrate into a different partition. Examples of such operations are:

- Update a value so that it moves to a new row partition
- Table reorganization
- Split a table partition or combine two partitions
- Export or import the table
- ALTER TABLE with the MOVE option
- ALTER TABLE with the SHRINK SPACE option
- FLASHBACK TABLE
- Redefine a table by using **dbms_redefinition**
- UPDATE to a non-partitioned table that changes row size so that the data does not fit into the current block
- DELETE of a row in a non-partitioned table and then re-insert.

Define partition schemes and row partitions

Use the **add partition** command to create row partitions and assign them to a partition scheme.

To partition rows based on column conditions

Issue **add partition** for each row partition that you want to create in a given partition scheme. When you create the first row partition, SharePlex creates the partition scheme as well.

`sp_ctrl>` **add partition to** *scheme_name* **setcondition =** *column_condition* **and route =** *routing_map* [**and name =** *name*] [**and tablename =***owner.table*] [**and description =***description*]

To partition rows based on a hash value:

Issue **add partition** once to specify the number of hash partitions to create.

`sp_ctrl>` **add partition to** *scheme_name* **set hash =** *value* **and route =** *value*

Add partition command syntax

NOTE: After you specify **add partition** with **to***scheme_name* and the **set** keyword, all other components can be in any order.

Component	Description
to *scheme_name*	**to** is a required keyword indicating the row partition is being added to *scheme_name*. *scheme_name* is the name of the partition scheme. The partition scheme is created by the first **add partition** command that you issue, which will also specify the first set of rows to partition. If you are making heavy use of horizontal partitioning, it may help to establish naming conventions for your partition schemes.
set	Required keyword that starts the definition of the row partition.
condition = *column_condition*	Creates a row partition based on a column condition. The condition must be in quotes. Use standard WHERE conditional syntax such as **((region_id = West) and region_id is not null)**. See How to create a valid column condition for additional information. The **condition** and **hash** components are mutually exclusive.
hash = *value*	Creates a row partition based on a hash value. The specified value determines the number of row partitions in the partition scheme. The **condition** and **hash** components are mutually exclusive.
route = *routing_map*	The route for this partition. This can be one of the following: **Partition based on a column condition:** Specify any standard SharePlex routing map, for example: **sysB@o.myora** or **sysB:q1@o.myora** or **sysB@o.myora+sysC@o.myora** (compound routing map). If the target is JMS, Kafka, or a file, then the target should be specified as x.jms, x.kafka, or x.file, for example: **sysA:hpq1@x.kafka**. To route a partition to multiple target tables that have different names, do the following: • Issue a separate **add partition** command for each different target name. Use the **tablename** option to specify the name. • In the configuration file, specify any of these target tables as the target table in the entry that uses this partition scheme. SharePlex will detect the other names when the configuration is activated.

Component	Description
	• Set the SP_ORD_FIRST_FIND parameter to 0 so that SharePlex checks all of the column conditions in the partition scheme. By default SharePlex assumes that any given row change will satisfy only one column condition in the partition scheme. For more information, see the SharePlex Reference Guide. **Partition based on a hash:** Use the following format to direct SharePlex to create a named post queue for each partition: *host*:*basename*|#{**o.***SID* | **r.***database*} where: • *host* is the name of the target system. • *basename* is the base name that is assigned to all queues. • **|#** directs SharePlex to number the queues sequentially by appending the base name with an integer, starting with 1 to the value set with **hash**. • **o.***SID* for an Oracle target or **r.***database* for an Open Target target.
name = *name*	(Recommended) A short name for this partition. This option is only useful for partitions based on column conditions. A name eliminates the need to type out long column conditions in the event that you need to modify or drop the partition in the future.
tablename = *owner.table*	(Optional) Use this option when there are multiple target tables and one or more have different names. Issue a separate **add partition** command for each name. The table name must be fully qualified. If case-sensitive, the name must be specified in quotes. **Example:** **add partition to scheme1 set name = p1 and condition = "C1 > 200" and route = sysb:p1@o.orasid and tablename = myschema.mytable**
description = *description*	(Optional) Description of this partition.

Examples

Partitions based on a column condition

Route different sets of rows through different post queues:

```
sp_ctrl> add partition to scheme1 set name = q1 and condition = "C1 >= 200" and route =
sysb:q1@o.orasid

sp_ctrl> add partition to scheme1 set name = q2 and condition = "C1 < 200" and route =
sysb:q2@o.orasid
```

Route different sets of rows to different target systems and different table names from the source:

```
sp_ctrl> add partition to scheme1 set name = east and condition = "area = east" and route =
sys1e@o.orasid and tablename = ora1.targ

sp_ctrl> add partition to scheme1 set name = west and condition = "area = west" and route =
sys2w@o.orasid and tablename = ora2.targ
```

Partitions based on a hash

Divide rows into four partitions, each processing through a different post queue:

```
sp_ctrl> add partition to scheme1 set hash = 4 and route = sysb:hash|#@o.ora112
```

How to create a valid column condition

The following are guidelines for creating column conditions. These guidelines do not apply to row partitions that are created with a hash value.

Choose appropriate columns

The types of columns on which you base your column conditions vary per datasource:

Base column conditions on columns whose values will not change, such as PRIMARY or UNIQUE key columns. The objective is to avoid a *partition shift*, where changes made to the conditional columns of a partition can cause the underlying data to satisfy the conditions of a different (or no) partition.

Partition shift case 1: The column value is updated so that the new value no longer satisfies any column condition:

- SharePlex performs the operation, but future operations on that row are not replicated. The reason: the row no longer satisfies a column condition.
- The source and target tables of the original partition are now out of synchronization, but Post returns no errors.

Partition shift case 2: A row that satisfies one column condition gets updated to meet a different condition:

- Post cannot find a matching target row. The reason: the original change was not replicated because it did not meet the column condition.
- Post returns an out-of-sync error.

You can use the following method to repair the out-of-sync rows that are caused by changes to the values of column conditions:

- Use the **compare** command to repair the out-of-sync rows. For more information about this command, see the SharePlex Reference Guide.

Additionally, you can ensure that data is replicated properly by setting the following parameter on the **source** prior to activating the configuration file.

- Set the SP_ORD_HP_IN_SYNC parameter to a value of 1. When this parameter is enabled, if an UPDATE changes a column (conditional column) value resulting in a row that no longer satisfies the correct condition, SharePlex corrects the row. Enabling this parameter causes some performance degradation, depending on how many tables are configured for horizontally partitioned replication. See the SharePlex Reference Guide for more information and a list of conditions corrected by this parameter.

NOTE: If you are using a column other than a key to base the column condition on, and you notice reduced performance with horizontally partitioned replication enabled, add a log group for that column.

Use supported data types

SharePlex supports the following data types in column conditions:

- NUMBER
- DATE
- CHAR
- VARCHAR
- VARCHAR2
- LONG VARCHAR

NOTES:

- For the dates, SharePlex uses MMDDSYYYYHH24MISS. For example:

```
hiredate<'1111 2011000000'
```

- Horizontally partitioned replication does not support the following:
  - data types other than the ones listed in this section. This also excludes large types like LOBs and object types such as VARRAYs and abstract data types.
  - Oracle TO_DATE function
  - UPDATEs or INSERTs on LONG columns larger than 100k
  - Sequences
  - TRUNCATEs of an Oracle partition

Use standard conditional syntax

The following list shows the conditional syntax that SharePlex supports in a column condition, where:

- *value* can be a string or a number. Enclose strings and dates within single quote marks (‘west’). Do not use quote marks for numbers .
- *column* is the name of a column in the table that you are configuring to use horizontally partitioned replication.

column **=** *value*
not (*column* **=** *value***)**
column **>** *value*
value **>** *column*
column **<** *value*
column **<=** *value*
column **>=** *value*
column **<>** *value*
column **!=** *value*
column **like** *value*
column **between** *value1* **and** *value2*
not (*column* **between** *value1* **and** *value2* **)**
column **is null**
column **is not null**

Conditions can be combined into nested expressions with parentheses and the **AND**, **OR**, and **NOT** logical connectives.

Example column conditions

not (col1 = 5)
(col2 = 5) and not (col3 = 6)
((col1 is not null) and (col2 = 5))

Additional guidelines

- NULLs are replicated by SharePlex in cases such as this one: **not (department_id = 90)**. If **department_id** is NULL, it is replicated. To avoid replicating records with NULLs, include the *column* **is not null** syntax as part of the condition, for example: **not (department_id = 90) and department_id is not null**.
- If parentheses are not used to indicate operator precedence, SharePlex supports operator precedence in the same order that SQL does. For example, a condition **not x and y** behaves the same way as **(not x) and y**. A condition **x and y or z** behaves the same as **(x and y) or z**. When a condition includes parentheses, the explicit precedence is respected.
- If the condition column is a VARCHAR column and the values used to define the partitions are string literals, the entire condition must be enclosed in double quotes, as in the following example: **add partition to***scheme***set route=***route***and condition="C2 = 'Fred'"**
- If the column name must be enclosed in quotes, then the entire condition must be enclosed in quotes, as in the following example: **add partition to***scheme***set route=***route***and condition="\"c2\" > 0"**
- **Do not**:
  - include references to other tables in the column condition.
  - exceed the 1024 bytes maximum defined storage.

- During the activation of a configuration that refers to partition schemes, SharePlex verifies the syntax in the column conditions of those schemes. If any syntax is incorrect, the activation fails. SharePlex prints an error to the Event Log that indicates where the error occurred.

Specify partition schemes in the configuration file

Use one configuration file for all of the data that you want to replicate from a given datasource, including tables that will have full-table replication and those that will use partitioned replication. For more information about how to create a configuration file, see Configure SharePlex to replicate data.To configure entries for horizontally partitioned replication, use the following syntax.

Datasource:o.*SID*		
src_owner.table	*tgt_owner.table*	!*partition_scheme*
!		*routing_map*

Component	Description
o.*database*	The datasource designation. Use the **o.** notation for an Oracle source. For *database*, specify the ORACLE_SID.
src_owner.table and tgt_owner.table	The specifications for the source and target tables, respectively.
!*partition_scheme*	The name of the partition scheme to use for the specified source and target tables. The **!** is required. The name is case-sensitive. Compound routing of multiple partition schemes is not supported, for example **!schemeA+schemeB**. Create a separate entry for each partition scheme that you want to use for the same source table. See **Examples**.
! *routing_map*	A *placeholder routing map*. It is required only if a route that you used in a partition scheme is not listed somewhere in the configuration file. SharePlex requires every route to be in the configuration file even if it is listed in a partition scheme. **NOTES:** • This option is valid only for partitions based on a column condition • If using named queues, list each queue route with this option • If routing the partition scheme to different targets, list each one with this option. You can use a compound routing map if the names of all target tables are identical. See **Examples**.

Examples

To specify a partition scheme:

Datasource: o.mydb		
scott.emp	scott.emp_2	!partition_emp

To specify multiple partition schemes for the same source table:

Datasource: o.mydb		
scott.emp	scott.emp_2	!partition_schemeA
scott.emp	scott.emp_3	!partition_schemeB

To specify a placeholder routing map:

! targsys1
! targsys2@o.ora2+targsys3@o.ora3
This placeholder is only required for partitions based on column conditions.

View the partitions and schemes

Use the **view partitions** command to view the row partitions in one partition scheme or all partition schemes in a horizontally partitioned replication configuration.

To view row partitions:

1. Run **sp_ctrl** on the source system.
2. Issue the following command with either option, depending on whether you want to view all partitions or just those for a particular partition scheme.

   `sp_ctrl>` **view partitions for** {*scheme_name* | **all**}

The following example shows both a hash-based partition scheme and a column-based partition scheme.

```
sp_ctrl> view partitions all

Scheme        Name           Route                           Hash     Condition
-----------   -------------  ------------------------------  ------   -------------
--
HASH4         hash           sys02:hash|#@o.ora112           4        ROWID
TEST_CT       highvalues     sys02:highvalues@o.ora112                sales>=10000
TEST_CT       lowvalues      sys02:lowvalues@o.ora112                 sales<10000
```

Hash4 hash-based partition scheme

- The Scheme column shows a partition scheme named **HASH4**.
- The Name column shows that the name for the partition definition is **hash**.
- The Route column shows that the partitions are created automatically and that the target is **o.ora112**.

- The Hash column has a value of 4, which indicates that this is a hash-based partition scheme with four partitions.
- The Condition column shows that the type of hash algorithm that is being used is the default of rowid, rather than block.

TEST_CT column-based partition scheme

- The Scheme column shows a partition scheme named TEST_CT. There are two entries for this name, indicating that it contains two partitions.
- The Name column shows the name of each partition, which by default is the name of the post queue or the value set with the **Name** option of the **add partition** command.
- The Route column shows that the names of the post queues are based on the parition name and that the target is **o.ora112**.
- The Hash column is empty for a column-based partition scheme.
- The Condition column shows the column condition that creates the row partition.

To view partition post queues:

The **qstatus** command on the target shows the post queues that are associated with horizontally partitioned replication.

Queues for TEST_CT column-based partition scheme

```
sp_ctrl sys02> qstatus

Queues Statistics for sys02

  Name:  highvalues (o.ora11-o.ora112) (Post queue)
    Number of messages:          0 (Age         0 min; Size          1 mb)
    Backlog (messages):          0 (Age         0 min)

  Name:  lowvalues (o.ora11-o.ora112) (Post queue)
    Number of messages:          0 (Age         0 min; Size          1 mb)
    Backlog (messages):          0 (Age         0 min)
```

Queues for HASH4 hash-based partition scheme:

```
Queues Statistics for sys02

  Name:  hash1 (o.ora11-o.ora112) (Post queue)
    Number of messages:          0 (Age         0 min; Size          1 mb)
    Backlog (messages):          0 (Age         0 min)

  Name:  hash2 (o.ora11-o.ora112) (Post queue)
    Number of messages:          0 (Age         0 min; Size          1 mb)
    Backlog (messages):          0 (Age         0 min)

 Name:  hash3 (o.ora11-o.ora112) (Post queue)
    Number of messages:          0 (Age         0 min; Size          1 mb)
    Backlog (messages):          0 (Age         0 min)
```

```
Name:  hash4 (o.ora11-o.ora112) (Post queue)
   Number of messages:          0 (Age         0 min; Size          1 mb)
   Backlog (messages):          0 (Age         0 min)
```

Make changes to partition schemes

The following commands or parameters are available to manage partition schemes. For more information, see the SharePlex Reference Guide.

Task	Command/Parameter	Description
Modify a partition	**modify partition** command	Modifies any of the attributes of a row partition definition.
Remove a partition scheme	**drop partition scheme** command	Removes the partition scheme and all row partitions within it.
Change hash algorithm	SP_OCF_HASH_BY_BLOCK	Change the hash algorithm from the default of rowid-based to block-based. Set to 1 to enable block-based algorithm.

Overview of Horizontally Partitioned Replication for PostgeSQL and PostgreSQL Database as a Service

Supported targets

PostgreSQL, Oracle, SQL Server, and Kafka

NOTES:

- PostgreSQL to SQL server replication does not support the BOOLEAN, TIME, TIME WITH TIME ZONE, and BYTEA data types for horizontally partitioned data.
- PostgreSQL to PostgreSQL replication does not support the JSON and JSONB data types for horizontally partitioned data.

To configure horizontally partitioned replication for a table, the steps are:

1. Define *row partitions* and link them to a *partition scheme*.
    - A row partition is a subset of rows in a source table that you want to replicate as a group.
    - A partition scheme is a logical container for row partitions.
2. Specify the name of the partition scheme in the SharePlex configuration file to include the partitions in replication.

Partition type

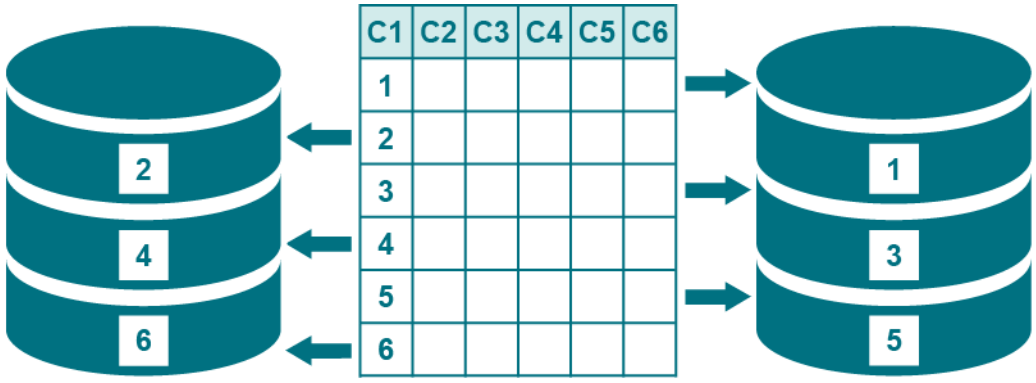
- **Columns**: A *column-based partition scheme* contains row partitions defined by a *column condition*. A column condition is a WHERE clause that defines a subset of the rows in the table.

About column-based partition schemes

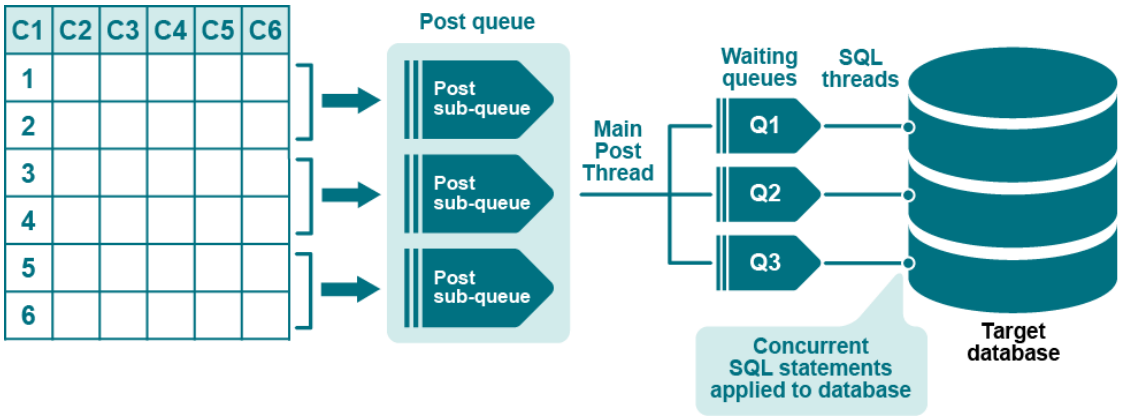
You can use row partitions based on column conditions for the following purposes:

- Use a single row partition to replicate only a subset of the rows of a table. For example, you can replicate only those rows where the value of the YEAR column is greater than 2014. The partition scheme in this case could be named "Since2014" or "Recent."

- Use multiple row partitions to divide the rows of a table so that each set of rows replicates to a different target. For example, a table named CORPORATE.SALES could have two row partitions named "East" and "West." The column conditions are defined accordingly, where the rows that satisfy REGION = EAST replicate to one location and the rows that satisfy REGION = WEST replicate to a different location. The partition scheme could be named "Sales_by_region."



- Use multiple row partitions to divide the rows of a table into parallel processing streams (parallel Export-Import-Post streams) for faster posting to a target table. For example, you can improve the flow of replication to a heavily updated target table. The use of column conditions for this purpose is appropriate only if the table contains a column (recommended to use primary or non-null unique key) that enables you to split the processing evenly among parallel Post processes.



Combine partitioned replication with full-table replication

You can combine horizontally partitioned and vertically partitioned replication for maximum control over how information is distributed.

For example:

- A corporate headquarters maintains a primary corporate database.
- Each of the corporation's four regional offices maintains its own database.
- Corporate headquarters uses vertically partitioned replication to share some column data of a table with the regional offices, but does not share any columns that contain sensitive data.

- The rows of the table are horizontally partitioned into four groups (East, West, North, South) for replication, so that each region receives only the record changes that apply to it.

Horizontally partitioned replication can be used in conjunction with full-table replication for the same table, for example to route groups of rows to different reporting systems and all rows to a backup system.

Define partition schemes and row partitions

Use the **add partition** command to create row partitions and assign them to a partition scheme.

To partition rows based on column conditions:

Issue **add partition** for each row partition that you want to create in a given partition scheme. When you create the first row partition, SharePlex creates the partition scheme as well.

`sp_ctrl>` **add partition to** *scheme_name* **setcondition =** *column_condition* **and route =** *routing_map* [**and name =** *name*] [**and tablename =***schema.table*] [**and description =***description*]

Add partition command syntax

NOTE: After you specify **add partition** with **to***scheme_name* and the **set** keyword, all other components can be in any order.

Component	Description
to *scheme_name*	**to** is a required keyword indicating the row partition is being added to *scheme_name*. *scheme_name* is the name of the partition scheme. The partition scheme is created by the first **add partition** command that you issue, which will also specify the first set of rows to partition. If you are making heavy use of horizontal partitioning, it may help to establish naming conventions for your partition schemes.
set	Required keyword that starts the definition of the row partition.
condition = *column_condition*	Creates a row partition based on a column condition. The condition must be in quotes. Use standard WHERE conditional syntax such as **((region_id = West) and region_id is not null)**. See How to create a column condition for additional information.
route = *routing_map*	The route for this partition. This can be one of the following: **Partition based on a column condition:** Specify any standard SharePlex routing map, for example: **sysB@r.dbname** or **sysB:q1@r.dbname** or **sysB@r.dbname+sysC@r.dbname** (compound routing map). **To route a partition to multiple target tables that have different names, do the following:**

Component	Description
	• Issue a separate **add partition** command for each different target name. Use the **tablename** option to specify the name. • In the configuration file, specify any of these target tables as the target table in the entry that uses this partition scheme. SharePlex will detect the other names when the configuration is activated. • Set the `SP_ORD_FIRST_FIND` parameter to 0 so that SharePlex checks all of the column conditions in the partition scheme. By default SharePlex assumes that any given row change will satisfy only one column condition in the partition scheme. For more information, see the SharePlex Reference Guide.
name = *name*	(Recommended) A short name for this partition. This option is only useful for partitions based on column conditions. A name eliminates the need to type out long column conditions in the event that you need to modify or drop the partition in the future.
tablename = *schemaname.table*	(Optional) Use this option when there are multiple target tables and one or more have different names. Issue a separate **add partition** command for each name. The table name must be fully qualified. If case-sensitive, the name must be specified in quotes. **Example:** **add partition to scheme1 set name = p1 and condition = "C1 > 200" and route = sysb:p1@r.dbname and tablename = myschema.mytable**
description = *description*	(Optional) Description of this partition.

Examples

Partitions based on a column condition

Route different sets of rows through different post queues:

```
sp_ctrl> add partition to scheme1 set name = q1 and condition = "C1 >= 200" and route = sysb:q1@r.dbname

sp_ctrl> add partition to scheme1 set name = q2 and condition = "C1 < 200" and route = sysb:q2@r.dbname
```

Route different sets of rows to different target systems and different table names from the source:

```
sp_ctrl> add partition to scheme1 set name = east and condition = "area = east" and route = sys1e@r.dbname and tablename = schema1.targ

sp_ctrl> add partition to scheme1 set name = west and condition = "area = west" and route = sys2w@r.dbname and tablename = schema2.targ
```

How to create a valid column condition

The following are guidelines for creating column conditions.

Choose appropriate columns

The types of columns on which you base your column conditions vary per datasource:

Base column conditions on columns whose values will not change, such as PRIMARY or UNIQUE key columns. The objective is to avoid a *partition shift*, where changes made to the conditional columns of a partition can cause the underlying data to satisfy the conditions of a different (or no) partition.

Partition shift case 1: The column value is updated so that the new value no longer satisfies any column condition:

- SharePlex performs the operation, but future operations on that row are not replicated. The reason: the row no longer satisfies a column condition.
- The source and target tables of the original partition are now out of synchronization, but Post returns no errors.

Partition shift case 2: A row that satisfies one column condition gets updated to meet a different condition:

- Post cannot find a matching target row. The reason: the original change was not replicated because it did not meet the column condition.
- Post returns an out-of-sync error.

Additionally, you can ensure that data is replicated properly by setting the following parameter on the **source** prior to activating the configuration file.

- Set the `SP_ORD_HP_IN_SYNC` parameter to a value of 1. When this parameter is enabled, if an UPDATE changes a column (conditional column) value resulting in a row that no longer satisfies the correct condition, SharePlex corrects the row. Enabling this parameter causes some performance degradation, depending on how many tables are configured for horizontally partitioned replication. See the SharePlex Reference Guide for more information and a list of conditions corrected by this parameter.

NOTE: If you are using a column other than a key to base the column condition on, and you notice reduced performance with horizontally partitioned replication enabled, add a log group for that column. In PostgreSQL, you can set replica identity FULL to use this parameter

Use supported data types

SharePlex supports the following data types in column conditions:

- SMALLINT
- INT
- BIGINT
- NUMERIC
- CHAR (<=2000 in length)
- VARCHAR (1<=4000 in length)

- DATE
- BOOLEAN (condition = "column_name =1" or condition = "column_name = 0")

NOTES:

- For the dates, SharePlex uses MMDDSYYYYHH24MISS. For example:

```
hiredate<'1111 2011000000'
```

- Horizontally partitioned replication does not support the following:
  - Data types other than the ones listed in this section.
  - Large datatypes like TEXT, BYTEA, CHAR > 2000 length, VARCHAR > 4000 length, VARCHAR without length will not be supported in column condition.
  - UPDATEs or INSERTs on LONG columns larger than 100k (length of data to be updated or insertsin the column)

Use standard conditional syntax

The following list shows the conditional syntax that SharePlex supports in a column condition, where:

- *value* can be a string or a number. Enclose strings and dates within single quote marks ('west'). Do not use quote marks for numbers .
- *column* is the name of a column in the table that you are configuring to use horizontally partitioned replication.

column **=** *value*
not (*column* **=** *value***)**
column **>** *value*
value **>** *column*
column **<** *value*
column **<=** *value*
column **>=** *value*
column **<>** *value*
column **!=** *value*
column **like** *value*
column **between** *value1* **and** *value2*
not (*column* **between** *value1* **and** *value2* **)**
column **is null**
column **is not null**

Conditions can be combined into nested expressions with parentheses and the **AND**, **OR**, and **NOT** logical connectives.

Example column conditions

not (col1 = 5)
(col2 = 5) and not (col3 = 6)
((col1 is not null) and (col2 = 5))

Additional guidelines

- NULLs are replicated by SharePlex in cases such as this one: **not (department_id = 90)**. If **department_id** is NULL, it is replicated. To avoid replicating records with NULLs, include the *column* **is not null** syntax as part of the condition, for example: **not (department_id = 90) and department_id is not null**.

- If parentheses are not used to indicate operator precedence, SharePlex supports operator precedence in the same order that SQL does. For example, a condition **not x and y** behaves the same way as **(not x) and y**. A condition **x and y or z** behaves the same as **(x and y) or z**. When a condition includes parentheses, the explicit precedence is respected.
- If the condition column is a VARCHAR column and the values used to define the partitions are string literals, the entire condition must be enclosed in double quotes, as in the following example: **add partition to***scheme***set route=***route***and condition="C2 = 'Fred'"**
- If the column name must be enclosed in quotes, then the entire condition must be enclosed in quotes, as in the following example: **add partition to***scheme***set route=***route***and condition="\"c2\" > 0"**
- **Do not**:
  - include references to other tables in the column condition.
  - exceed the 1024 bytes maximum defined storage.
- During the activation of a configuration that refers to partition schemes, SharePlex verifies the syntax in the column conditions of those schemes. If any syntax is incorrect, the activation fails. SharePlex prints an error to the Event Log that indicates where the error occurred.

Specify partition schemes in the configuration file

Use one configuration file for all of the data that you want to replicate from a given datasource, including tables that will have full-table replication and those that will use partitioned replication. For more information about how to create a configuration file, see Configure SharePlex to Replicate Data . To configure entries for horizontally partitioned replication, use the following syntax.

Datasource:r.*dbname*		
srcschemaname.table	targetschemaname.table	!*partition_scheme*
!		*routing_map*

Component	Description
r.*dbname*	The datasource designation. Use the **r.** notation for an PostgreSQL source.
src_schema.table and tgt_ schema.table	The specifications for the source and target tables, respectively.
!*partition_scheme*	The name of the partition scheme to use for the specified source and target tables. The **!** is required. The name is case-sensitive. Compound routing of multiple partition schemes is not supported, for example **!schemeA+schemeB**. Create a separate entry for each partition scheme that you want to use for the same source table. See **Examples**.
! *routing_map*	A *placeholder routing map*. It is required only if a route that you used in a partition scheme is not listed somewhere in the configuration file. SharePlex requires every route to be in the configuration file even if it is

Component	Description
	listed in a partition scheme. **NOTES:** • This option is valid only for partitions based on a column condition. • If using named queues, list each queue route with this option • If routing the partition scheme to different targets, list each one with this option. You can use a compound routing map if the names of all target tables are identical.

Examples

To specify a partition scheme:

Datasource: r.mydb		
scott.emp	scott.emp_2	!partition_emp

To specify multiple partition schemes for the same source table:

Datasource: r.mydb		
scott.emp	scott.emp_2	!partition_schemeA
scott.emp	scott.emp_3	!partition_schemeB

To specify a placeholder routing map:

! targsys1
! targsys2@r.dbname2+targsys3@r.dbname3

This placeholder is only required for partitions based on column conditions.

View the partitions and schemes

Use the **view partitions** command to view the row partitions in one partition scheme or all partition schemes in a horizontally partitioned replication configuration.

To view row partitions:

1. Run **sp_ctrl** on the source system.
2. Issue the following command with either option, depending on whether you want to view all partitions or just those for a particular partition scheme.

   `sp_ctrl>`**view partitions for** {*scheme_name* | **all**}

The following example shows a column-based partition scheme:

```
sp_ctrl> view partitions all
```

Scheme	Name	Route	Tablename	Condition
product	lessQuantity	10.250.40.27@r.testdb	splex.prod_1	id between 1 and 100
product	moreQuantity	10.250.40.27@r.testdb	splex.prod_2	id between 101 and 200
product	largeQuantity	10.250.40.27@r.testdb	splex.prod_3	id between 201 and 300
sales_by_region	east	10.250.40.27@r.testdb	splex.sales_dst1	((region = 'East') and region is not null)
sales_by_region	west	10.250.40.27@r.testdb	splex.sales_dst2	((region = 'west') and region is not null)
city_scheme	Pune	10.250.40.27:pune_queue@r.testdb	splex.student_target1	((stud_name = 'Pune') and stud_name is not null)
city_scheme	Mumbai	10.250.40.24:mumbai_queue@r.testdb	splex.student_target2	((stud_name = 'Mumbai') and stud_name is not null)

city_scheme column-based partition scheme

- The Scheme column shows a partition scheme named city_scheme. There are two entries for this name, indicating that it contains two partitions.
- The Name column shows the name of each partition, which by default is the name of the post queue or the value set with the **Name** option of the **add partition** command.
- The Route column shows that the names of the post queues are based on the parition name and that the target is **r.testdb**.
- The Condition column shows the column condition that creates the row partition.

To view partition post queues:

The **qstatus** command on the target shows the post queues that are associated with horizontally partitioned replication.

Queues for `city_scheme` column-based partition scheme

```
sp_ctrl (pslinuxpgsp11:2200)> qstatus

Queues Statistics for pslinuxpgsp11

  Name:  pune_queue (r.testdb-r.testdb) (Post queue)
    Number of messages:          0 (Age         0 min; Size          1 mb)
    Backlog (messages):          0 (Age         0 min)

sp_ctrl (pslinuxpgsp08:2200)> qstatus

Queues Statistics for pslinuxpgsp08

  Name:  mumbai_queue (r.testdb-r.testdb) (Post queue)
    Number of messages:          0 (Age         0 min; Size          1 mb)
    Backlog (messages):          0 (Age         0 min)
```

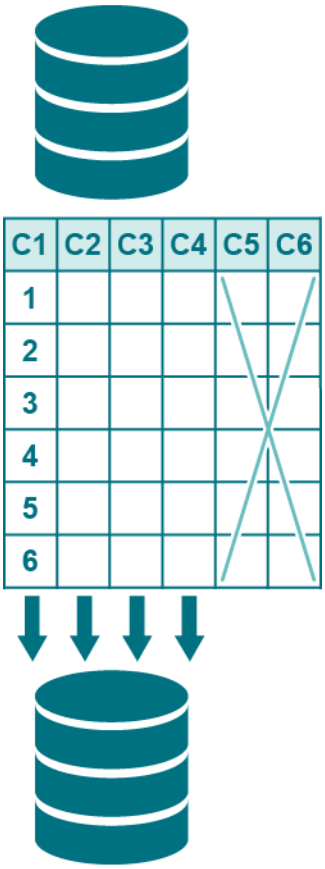
Make changes to partition schemes

The following commands or parameters are available to manage partition schemes. For more information, see the SharePlex Reference Guide.

Task	Command/Parameter	Description
Modify a partition	**modify partition** command	Modifies any of the attributes of a row partition definition.
Remove a partition scheme	**drop partition scheme** command	Removes the partition scheme and all row partitions within it.

Configure Vertically Partitioned Replication

Use vertically partitioned replication to replicate a subset of the columns of a table. For example, you can replicate data changes made to C1, C2, C3, and C4, but not changes made to C5 and C6, as shown in the diagram.



Supported sources and targets

- PostgreSQL to PostgreSQL, Oracle, SQL Server, and Kafka
- Oracle to all targets
- PGDB as a Service to PGDB as a Service
- PGDB as a Service to Oracle
- PGDB as a Service to PostgreSQL

Guidelines for using vertically partitioned replication

Follow these guidelines when creating a configuration file that includes vertically partitioned replication:

- Vertically partitioned replication is appropriate for reporting and other data sharing strategies, but it is not appropriate for high availability environments. Once you configure a table for vertically partitioned replication, SharePlex does not recognize the other columns, so data in those columns is not replicated.
- You can combine horizontally partitioned and vertically partitioned replication for maximum control over which information is distributed, and to where.

  For example: A company has a headquarters and regional divisions. The headquarters maintains the corporate database, and each region maintains a regional database. The headquarters uses vertically partitioned replication to share some of the column data of a table to those locations, while retaining other sensitive data at headquarters. Row changes made to the shared columns are further partitioned horizontally, for replication to the appropriate regional database.
- A table cannot be configured to replicate some columns to one target system and all columns to another (combination of vertically partitioned replication and full-table replication). You can, however, configure full-table replication to an identical table on one target, and then configure vertically partitioned replication from that target to a second target that contains the table that requires only the partition columns.
- A target table can, but does not have to, contain all of the same columns as its source table. The target can contain just the columns being replicated from the source table. The names of corresponding source and target columns do not need to be the same. Corresponding columns must contain the same data types (same type, size, precision).

Overview of vertically partitioned replication: Oracle to all targets

To configure vertically partitioned replication, you specify either a *column partition* or an *exclusion column partition* in the configuration file:

- A *column partition* specifies the columns that you want to include in replication. Only data changes that are made to the specified columns get sent to the target.
- An *exclusion column partition* specifies columns to be excluded from replication. No data from those columns is replicated to the target.

Follow these rules to specify either type of column partition:

- There can be one partition per source table. A column partition and an exclusion partition are mutually exclusive.
- A column list must be enclosed within parentheses.
- Separate each column name with a comma. A space after the comma is optional.
- The maximum length of a partition is 174820 bytes (the maximum line length allowed in a configuration file). Therefore, the actual number of columns that you can list depends on the length of each name.
- The columns can be contiguous or non-contiguous in the source table. For example, you can replicate the first, third and seventh columns of a table.
- Key columns are *not* required to be included in the partition.
- If using horizontally partitioned and vertically partitioned replication together for this table, all of the columns in the partition scheme must be part of the column condition.
- Use one configuration file for all of the data that you want to replicate from a given datasource, including tables that will have full-table replication and those that will use partitioned replication.

To configure entries for vertically partitioned replication, use the following syntax. For more information about how to create a configuration file, see Configure SharePlex to replicate data.

```
datasource_specification

# table specification with column partition

src_owner.table   (src_col,src_col,...)        tgt_owner.table   [(tgt_col,tgt_col,...)]        routing_map

# table specification with exclusion column partition

src_owner.table   !(src_col,src_col,...)       tgt_owner.table                                  routing_map
```

Configuration component	Description
src_owner.table and tgt_owner.table	The specifications for the source and target tables, respectively.
(src_col, src_col,...)	Specifies a *column partition* that lists the columns to include in replication. No other column data is replicated, including data in columns that are added after the start of replication (assuming DDL replication is enabled).
!(src_col,src_col,...)	Specifies an *exclusion column partition* that lists the columns to *exclude* from replication. All other column data is replicated, including data in columns that are added after the start of replication (assuming DDL replication is enabled). **NOTE:** When using an *exclusion column partition*, the corresponding source and target column names must be identical, and the excluded columns cannot be used in a key definition. For more information, see Define a Unique Key .

Configuration component	Description
(*tgt_col*,*tgt_col*,...)	The target columns. Use this option to map source columns to target columns that have different owners or names. If the source and target columns have identical owners or names, the target columns can be omitted. **To map source columns to target columns, follow these rules:** • The syntax rules for the source column partition also apply to the target column list. • The target columns must have identical definitions as their source columns, except for their names. • List the target columns in the same logical order as their corresponding source columns. This is required regardless of the actual order of the target columns in the table, so that SharePlex builds the correct correlations in the object cache. For example, a change to the second column in the source list is replicated to the second column in the target list.

routing_map	The routing map for the column partition. The routing map can be one of the following:
	
	• If using horizontally partitioned replication for the source table, specify a partition scheme, as in: **!partition_scheme**.
	• If not using horizontally partitioned replication for the source table, specify a routing map as follows:
	  • Use a simple routing map like **sysB@o.myora** if replicating the column partition to one target. A route with a named export or post queue is supported. For more information, see:
	  Configure Named Export Queues
	  Configure Named Post Queues
	  • Use a compound routing map like **sysB@o.myora+sysC@o.myora2** if replicating the column partition to multiple target systems.
	
	IMPORTANT! A compound routing map must be used, rather than listing multiple targets in separate entries, because *only one column condition per source table* can be listed in the configuration file. To use a compound routing map, the owners and names of all of the target tables must be identical. For more information, see Routing Specifications in a Configuration File.

Configuration examples

The following is a vertically partitioned replication configuration replicating to multiple targets by using a compound routing map. To use a compound routing map for this source table, all targets must be named **scott.sal**.

Datasourceo.oraA		
scott.emp (c1,c2)	**scott.sal**	**sysB@o.oraB+sysC@o.oraC**

The following is a vertically partitioned replication configuration replicating to a single target where the target columns have different names from those of the source.

Datasourceo.oraA		
scott.emp (c1,c2)	**scott.sal (c5,c6)**	**sysB@o.oraB**

The following configuration file is **not valid** because it repeats the same column partition of **scott.emp (c1, c2)** twice in the configuration file.

Datasourceo.oraA		
scott.emp (c1,c2)	**scott.cust (c1,c2)**	**sysB@o.oraB**
scott.emp (c1,c2)	**scott.sales (c1,c2)**	**sysC@o.oraC**

Overview of vertically partitioned replication for PostgreSQL and PostgreSQL Database as a Service

Supported targets

PostgreSQL, Oracle, SQL Server, and Kafka

NOTE: PostgreSQL to SQL server replication does not support the BOOLEAN, TIME, TIME WITH TIME ZONE, and BYTEA data types for vertically partitioned data.

To configure vertically partitioned replication:

To configure vertically partitioned replication, you specify either a *column partition* or an *exclusion column partition* in the configuration file:

- A *column partition* specifies the columns that you want to include in replication. Only data changes that are made to the specified columns get sent to the target.
- An *exclusion column partition* specifies columns to be excluded from replication. No data from those columns is replicated to the target.

Follow these rules to specify either type of column partition:

- There can be one partition per source table. A column partition and an exclusion partition are mutually exclusive.
- A column list must be enclosed within parentheses.
- Separate each column name with a comma. A space after the comma is optional.
- The maximum length of a partition is 174820 bytes (the maximum line length allowed in a configuration file). Therefore, the actual number of columns that you can list depends on the length of each name.
- The columns can be contiguous or non-contiguous in the source table. For example, you can replicate the first, third and seventh columns of a table.
- Key columns are *not* required to be included in the partition. There can be a performance impact if you are not partitioning key columns.
- Use one configuration file for all of the data that you want to replicate from a given datasource, including tables that will have full-table replication and those that will use partitioned replication.

To configure entries for vertically partitioned replication, use the following syntax:

datasource_specification

table specification with column partition

src_schema.table (src_col,src_col,...) tgt_schema.table [(tgt_col,tgt_col,...)] routing_map

table specification with exclusion column partition

src_schema.table !(src_col,src_col,...) tgt_schema.table routing_map

Configuration component	Description
src_schema.table and tgt_ schema.table	The specifications for the source and target tables, respectively.
(*tgt_col*,*tgt_col*,...)	The target columns. Use this option to map source columns to target columns that have different schemas or names. If the source and target columns have identical schemas or names, the target columns can be omitted. **To map source columns to target columns, follow these rules:** • The syntax rules for the source column partition also apply to the target column list. • The target columns must have identical definitions as their source columns, except for their names. • List the target columns in the same logical order as their corresponding source columns. This is required regardless of the actual order of the target columns in the table, so that SharePlex builds the correct correlations in the object cache. For example, a change to the second column in the source list is replicated to the second column in the target list.

routing_map	The routing map for the column partition. The routing map can be one of the following:
	• If using horizontally partitioned replication for the source table, specify a partition scheme, as in: **!partition_scheme**.
	• If not using horizontally partitioned replication for the source table, specify a routing map as follows:
	  • Use a simple routing map like **sysB@r.dbname** if replicating the column partition to one target. A route with a named export or post queue is supported. For more information, see:
	  Configure Named Export Queues
	  Configure Named Post Queues
	  • Use a compound routing map like **sysB@r.dbname+sysC@r.dbname2** if replicating the column partition to multiple target systems.
	IMPORTANT! A compound routing map must be used, rather than listing multiple targets in separate entries, because *only one column condition per source table* can be listed in the configuration file. To use a compound routing map, the schemas and names of all of the target tables must be identical.

Configuration examples

The following is a vertically partitioned replication configuration replicating to multiple targets by using a compound routing map. To use a compound routing map for this source table, all targets must be named **scott.sal**.

Datasource: r.dbname		
scott.emp (c1,c2)	scott.sal	sysB@r.dbname1 + sysC@r.dbname2

The following is a vertically partitioned replication configuration replicating to a single target where the target columns have different names from those of the source.

Datasource: r.dbname		
scott.emp (c1,c2)	scott.sal (c5,c6)	sysB@r.dbname1

The following configuration file is **not valid** because it repeats the same column partition of **scott.emp (c1, c2)** twice in the configuration file.

Datasource: r.dbname		
scott.emp (c1,c2)	scott.cust (c1,c2)	sysB@r.dbname1
scott.emp (c1,c2)	scott.cust (c1,c2)	sysC@r.dbname2

10

Configure Replication to a Change History Target

This chapter contains instructions for how to configure SharePlex to maintain a change-history target. SharePlex enables you to maintain this history, while also replicating the same data set to maintain up-to-date targets.

Contents

Overview of the Change-History Target

A change history target differs from a replication target in that a change history target maintains a record of every change that occurs to a source object or objects, rather than simply maintaining a mirror of the current state of the source data. While regular replication *overlays* current target data with change data, change history *inserts* the change data to the target as a new record. The old data is preserved as a step-by-step record of change. The historical data can be queried and analyzed for such purposes as data mining or resolving customer disputes.

By using SharePlex to maintain change history on a secondary server, you can offload the overhead from the production database. Such overhead includes the SQL work of adding the history rows, the extra storage of those rows, and the query activity against the historical data.

NOTE: File, JMS, and Kafka targets support change history by default, because every source change is written as a separate XML record. There is no overlaying of old data with new. Metadata that is supported for these targets is included automatically when Post writes the XML. For a list of supported metadata, see the **target** command in the SharePlex Reference Guide.

Capabilities

This replication strategy supports the following:

- Identical or different source and target names
- Use of vertically partitioned replication
- Use of horizontally partitioned replication
- Use of named export and post queues
- Combination of regular replication and change-history replication of the same source object(s)

Supported sources

Oracle

Supported targets

Oracle target

Operations supported

SharePlex supports adding a change history row for these operations:

- INSERT
- UPDATE
- DELETE
- TRUNCATE
- ALTER TABLE to DROP COLUMN

NOTE: Post does not drop the column from the table, but does create a change history row.

- ALTER TABLE to ADD COLUMN

NOTE: Post adds a column to the table, but does not create a change history row.

- ALTER TABLE to MODIFY the data type of a column

Operations not supported

- Changes made to UDT or VARRAY columns.

NOTE: SharePlex replicates tables with the UDT fields in the base type ONLY. In case of columns containing multiple subtypes, replication is applicable only for base type fields.

- DBMS_LOB operation that is used to change a part of a LOB column (The value stored for that column on the target will not be the complete LOB column.)

How SharePlex maintains change history

In a change history configuration, each target table serve as a *history table* that records every change made to the source data as a continuous series of rows.

Each new change row that SharePlex inserts includes the following:

- the values of the key columns
- the after image of the changed columns. For inserts and updates, the after image consists of the new values of the columns that were changed (or added in the case of an insert). For deletes, the after image consists of the key values plus the other columns set to null.
- (optionally) a set of metadata values that provide context for the change. For example, there is metadata that captures the userid of the user who made the change and the source system where the change was made (useful when change data is tracked from multiple databases).

SharePlex can be configured to include the before image of update operations in the history or to control which operation types are included in the history. For example, you could include only updates and deletes.

Configure Change History

To configure change history, you use special syntax in the SharePlex configuration file and, optionally, configure filter rules and other attributes to customize the history to your needs.

Create a change-history configuration file

Perform the following steps to create a change-history configuration file:

1. Make certain that SharePlex is installed and the system is prepared according to the instructions in the SharePlex Installation Guide.
2. Create the Oracle target history tables with the same name and structure as the source tables whose history they will track, *but omit all constraints on all columns*.

   IMPORTANT: The Oracle target tables must **not** have PRIMARY KEY, FOREIGN KEY, UNIQUE, NOT NULL, or CHECK constraints, nor can columns be defined with a DEFAULT value. Because this is a history of changes, a row may have the same image as another row that has the same key. Post does not perform integrity checks on a change-history target.
3. Disable triggers on the target tables.
4. Allow no DML or DDL to be performed on the target tables except by SharePlex.

5. On the source system, create a configuration file using the following syntax. For more information about how to create a configuration file, see Configure SharePlex to Replicate Data on page 60.

```
datasource_specification
src_owner.table          !cdc:tgt_owner.table          host@c.SID
```

where:

- **Datasource:o.***SID* is the ORACLE_SID of the source Oracle instance.
- *src_owner.table* is the fully qualified name of a source object (owner.object) or a wildcarded specification.
- **!cdc:** identifies the target as a change-history table.
- *tgt_owner.table* is the fully qualified name of the target history table or a wildcarded specification.
- *host* is the target system.
- **c.***SID* specifies the target Oracle instance.

6. (Optional) Run the following script on the target tables to add default metadata columns with their default names. Post automatically populates the default metadata columns without any additional configuration. You can customize the script to meet your requirements.

   product_dir/**util**/**add_change_tracking_columns.sql**

   NOTES:

   - The script only adds the default columns. To add optional columns, or to change a column name, use the **target** command to add them to the Post configuration. For a list of default and optional metadata columns, see the **target** command in the SharePlex Reference Guide.
   - The default columns are automatically added to new tables that are added to the SharePlex change history configuration.

Additional change history configuration options

This section describes how you can customize the SharePlex change history configuration.

Customize column names

You can use the **target** command with the *colname* option to customize the name of any target metadata column. For instructions, see the SharePlex Reference Guide.

Add the before image to each change row

You can include the before image of updates in the target table by setting the SP_OPO_TRACK_PREIMAGE parameter to **U**. This parameter causes Post to insert two rows to the target table for every change made to the tracked source table: one for the after image and one for the before image. The before image is composed of the key values plus the before values of the columns that were changed, unless the SP_OCT_USE_SUPP_KEYS parameter is used.

When before images are enabled, the SHAREPLEX_SOURCE_OPERATION column values for the two records will be:

UPDATE BEFORE

UPDATE AFTER

NOTE: The before row will not include the before image of any LOB columns, because the redo log does not contain the before image of LOBs.

You can override the global setting of SP_OPO_TRACK_PREIMAGE at the table level by using the **set cdc preimage** option of the **target** command.

For more information about SP_OPO_TRACK_PREIMAGE and the target command, see the SharePlex Reference Guide.

Include all columns of an operation in the history

This option is valid for Oracle data only. To include the values of all table columns in each target history record, rather than only the changed columns, configure the following:

1. Turn on supplemental logging for all columns of the source tables that are being tracked. For example:

   Alter table emp ADD SUPPLEMENTAL LOG DATA (ALL) COLUMNS;

2. Set the SP_OCT_USE_SUPP_KEYS parameter to 1.
3. Set the SP_OCT_INCLUDE_UNCHANGED_COL to 1.

NOTE: When both SP_OCT_USE_SUPP_KEYS and SP_OPO_TRACK_PREIMAGE are enabled, the before image includes all column values as they were before the change.

Disable change history of an operation type

To disable the change history of a DML operation type, set the SP_OPO_TRACK_OPERATIONS parameter to the appropriate value or values. Separate values with a slash (/). For example, to maintain change history only for inserts and updates, change the parameter to **I/U**.The default is **I/U/D** which sends all DML operation types to the history records.

Set rules and filters

You can use the **set rule** option of the **target** command to apply conditions on columns to control whether a change is applied to the target history table. For example, you can specify a rule that if column 1 and column 3 are changed, then apply the operation and discard any changes that apply to other columns. For instructions, see the SharePlex Reference Guide.

Include COMMITs

By default, the COMMIT record is not included in the history tables. To configure Post to insert a row for every COMMIT, set the `SP_OPO_TRACK_COMMITS` parameter to 1.

11

Configure a Replication Strategy

This chapter contains instructions for configuring SharePlex to support different replication objectives. Production implementations can vary widely from basic configurations with one source and target, to multiple instances of SharePlex with named queues,multiple targets, partitioned data, and more.

It is difficult to foresee and document every possible way that an organization may want to deploy SharePlex. The goal of this documentation is to present instructions for setting up the basic deployment types in a way that is clear enough for you to be able to combine them and expand upon them to suit your needs. Additional deployment assistance is available through our Professional Services organization.

Contents

Configure Replication to Share or Distribute Data

These instructions show you how to set up SharePlex for the purpose of sharing or distributing data from one source system to one or more target systems.

This strategy supports business requirements such as the following:

- reporting to support real-time decision making
- data sharing to support research and transparency requirements
- data integration throughout an enterprise
- customer service inquiries and other query-intensive applications
- data auditing and archiving

Supported sources

Oracle and PostgreSQL

Supported targets

All

Capabilities

This replication strategy supports the following:

- Replication to one or more target systems
- Replication between databases on the same system
- Replication between schemas in the same database (Oracle)
- Identical or different source and target names
- Use of vertically partitioned replication
- Use of horizontally partitioned replication
- Use of named export and post queues
- Use of transformation (Oracle)

Requirements

- Prepare the system, install SharePlex, and configure database accounts according to the instructions in the SharePlex Installation Guide.
- No DML or DDL should be performed on the target tables except by SharePlex. Tables on the target system that are outside the replication configuration can have DML and DDL operations without affecting replication.

- If sequences are unnecessary on the target system, do not replicate them. It can slow down replication. Even if a sequence is used to generate keys in a source table, the sequence values are part of the key columns when the replicated rows are inserted on the target system. The sequence itself does not have to be replicated.

IMPORTANT! These instructions assume you have a full understanding of SharePlex configuration files. They use abbreviated representations of important syntax elements.

For more information, see Configure SharePlex to Replicate Data on page 60.

Conventions used in the syntax

In the configuration syntax in this topic, the placeholders represent the following:

- *source_specification[n]* is the fully qualified name of a source object (owner.object) or a wildcarded specification.
- *target_specification[n]* is the fully qualified name of a target object or a wildcarded specification.
- *host* is the name of a system where SharePlex runs. Different systems are identified by appending a letter to the names, like *hostB*.
- *db* is a database specification. The database specification consists of either **o.** or **r.** prepended to the Oracle SID, TNS alias, or database name, as appropriate for the connection type. A database identifier is not required if the target is JMS, Kafka, or a file.

IMPORTANT! Configure SharePlex to Replicate Data on page 60.

Replicate within the local system

Replication on the same system supports the following configurations:

- Within one Oracle/PostgreSQL instance, replicate to different tables within the same schema or to the same table in different schemas.
- Replicate to from an Oracle/PostgreSQL instance to any SharePlex-supported target on the same system.

Configuration options

datasource_specification		
source_specification1	*target_specification1*	*hostA*[**@***db*]
source_specification2	*target_specification2*	*hostA*[**@***db*]

Example for Oracle

This example shows how you can replicate data to the same Oracle instance, to a different Oracle instance (Unix and Linux only), and to different target types, all on the same local system.

Datasource:o.oraA		
hr.emp	hr.emp2	hostA@o.oraA
hr.sal	hr.sal2	hostA@o.oraB
fin.*	fin.*	hostA@r.mss
act.*	!file	hostA

Example for PostgreSQL :

This example shows how you can replicate data to the same PostgreSQL instance, to a different PostgreSQL instance (Linux only), and to different target types, all on the same local system.

Datasource:r.pgA		
hr.emp	hr.emp2	hostA@r.pgA
hr.sal	hr.sal2	hostA@r.pgB
fin.*	fin.*	hostA@r.mss

Configuration when using SharePlex Manager

Replication from and to the same machine omits an Export process. However, SharePlex Manager expects an export queue to exist. If using this configuration with SharePlex Manager, you must explicitly configure an export queue as follows. The hostA* component in the routing map creates the export queue and an Export process, which sends the data to an Import process, then the post queue.

datasource_specification		
source_specification1	*target_specification1*	*hostA***hostA*[**@***db*]
source_specification2	*target_specification2*	*hostA***hostA*[**@***db*]

Replicate to a remote target system

Configuration options

datasource_specification		
source_specification1	*target_specification1*	*hostB*[**@***db*]
source_specification2	*target_specification2*	*hostB*[**@***db*

Example

The last line in this example shows how you can replicate data to different target types on the same remote target system.

Datasource:o.oraA		
hr.emp	hr.emp2	hostB@o.oraB
hr.sal	hr.sal2	hostB@o.oraB
fin.*	!file	hostB

Replicate to multiple target systems

This topology is known as *broadcast replication*. It provides the flexibility to distribute different data to different target systems, or all of the data to all of the target systems, or any combination as needed. It assumes the source system can make a direct connection to all of the target systems. All routing is handled through one configuration file. For more information, see Configure Replication through an Intermediary System on page 207.

Configuration options

If the target specification is identical on all targets: If the target specification of a source object is identical on all target systems, you can use a compound routing map, rather than type a separate entry for each route. For more information, see Configure SharePlex to Replicate Data on page 60.

datasource_specification		
source_specification1	*target_specification1*	*hostB*[**@***db*]+*hostC*[**@***db*][+...]
source_specification2	*target_specification2*	*hostC*[**@***db*]+*hostD*[**@***db*][+...]

If the target specification is not identical on all targets

- When the target specification of a source object is different on some or all target systems, you must use a separate configuration entry to specify each one that is different.
- You can use a compound routing map for routes where the target specifications are identical.

datasource_specification		
source_specification1	*target_specification1*	*hostB*[**@***db*]
source_specification1	*target_specification2*	*hostC*[**@***db*]

Example (Oracle as a source)

NOTE: This example does not cover all possible source-target combinations. The last entry in this example shows the use of horizontally partitioned replication to distribute different data from the **sales.accounts** table to different regional databases.

Datasource:o.oraA		
hr.emp	hr.emp2	hostB@o.oraB
hr.emp	hr."Emp_3"	hostC@r.mssB
cust.%	cust.%	hostD@o.oraD+hostE@o.oraE
sales.accounts	sales.accounts	!regions

Example (PostgreSQL as a source and target)

Datasource:r.source_DB		
hr.emp	hr.emp2	hostB@r.dbnameA
hr.emp	hr."Emp_3"	hostC@r.dbnameB
cust.%	cust.%	hostD@r.demoC+hostE@r.demoD

IMPORTANT! Use the same port number for SharePlex on all systems.

Configure Replication to Maintain a Central Datastore

These instructions show you how to set up SharePlex for the purpose of *consolidated replication*: replicating from multiple source systems to one central target system.

This strategy supports business requirements such as the following:

- Real-time reporting and data analysis
- The accumulation of big data into a central datastore/mart or warehouse

Supported sources

Oracle and PostgreSQL

Supported targets

Oracle and open targets

Capabilities

This replication strategy supports the following:

- Identical or different source and target names
- Use of vertically partitioned replication
- Use of horizontally partitioned replication
- Use of named export and post queues

Requirements

- Prepare the system, install SharePlex, and configure database accounts according to the instructions in the SharePlex Installation Guide.
- No DML or DDL should be performed on the target tables except by SharePlex. Tables on the target system that are outside the replication configuration can have DML and DDL operations without affecting replication.
- Each source system must replicate a different set of data to the central target. If any source systems replicate the same data to the central target system, it is considered to be active-active replication. For more information, see Configure Peer-to-Peer Replication on page 165.
- If sequences are unnecessary on the target system, do not replicate them. It can slow down replication. Even if a sequence is used to generate keys in a source table, the sequence values are part of the key columns when the replicated rows are inserted on the target system. The sequence itself does not have to be replicated.

Deployment options

You have two options for deploying SharePlex to replicate from many source systems to one target system.

- One instance of SharePlex processes all of the incoming data from all sources. For more information, see Deploy with one instance of SharePlex on the target system on page 161.
- Multiple instances of SharePlex each process the incoming data from a different source. For more information, see Deploy with multiple instances of SharePlex on the target system on page 162.

In either deployment, if any source system cannot make a direct connection to the target system, you can use cascading replication for that route to enable SharePlex to cascade the data an intermediary system that allows connection to the target. For more information, see Configure Replication through an Intermediary System on page 207.

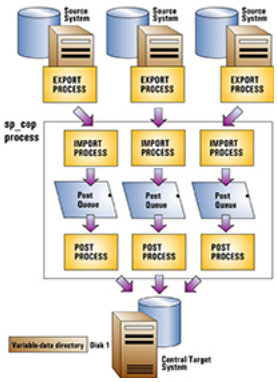
NOTE: The SharePlex **compare** and **repair** commands cannot be used in a cascading configuration.

Deploy with one instance of SharePlex on the target system

You can use one instance of SharePlex to process all incoming data on the target. For each source system, SharePlex creates an Import process on the central target system when replication starts. That, in turn, creates post queues and Post processes for each source-target replication stream, all controlled by one **sp_cop** process. You can control each source-target stream separately, but the post queues all share the same SharePlex variable-data directory on the target system.

A deployment with a single variable-data directory has the following potential risks:

- An event that interrupts processing to and from the disk that contains the variable-data directory will affect all replication streams.
- Any cleanup utilities that you use will affect all of the replication streams, because the cleanup is performed across the entire variable-data directory.
- A **purge config** command that is issued on one source system also deletes the data that is replicated from the other source systems, because the purge affects the entire variable-data directory. The use of named post queues eliminates this risk, but adds complexity to the naming, monitoring and management of the SharePlex objects in the deployment.



To use this deployment:

1. Install SharePlex in the normal manner, with one port number and one variable-data directory on each system (sources and target).
2. Make certain that when you install SharePlex, you create a database account for SharePlex for each installation.

IMPORTANT! Use the same port number for SharePlex on all systems.

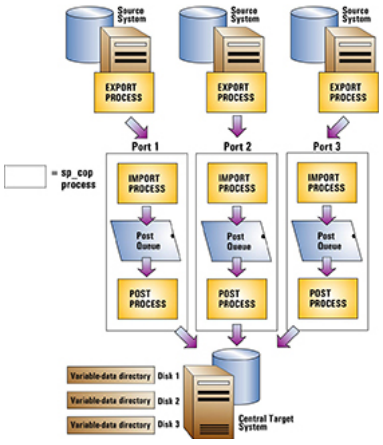
Deploy with multiple instances of SharePlex on the target system

You can deploy multiple instances of SharePlex on the target, one for each source system. A SharePlex instance is composed of the following elements:

- A unique **sp_cop** process
- A unique variable-data directory
- A unique port number on which **sp_cop** runs
- A unique database account that the processes of that instance use to interact with the database.

By running multiple, distinct instances of SharePlex, you can isolate each source-target replication stream from the others. It enables you to:

- Avoid contention problems that can occur if multiple processes must compete for access to the same queues in a single variable-data directory.
- Purge one configuration, or clean up and resynchronize one replication stream, while allowing the others to continue processing data.
- Place the variable-data directories on separate disks so that problems with one disk do not affect the variable-data directories on the other disks.



To use this deployment

Install on the target system first, if possible. This enables you to establish a port number for each variable-data directory, which you can then refer to when you set up SharePlex on the corresponding source system.

Steps on the target system

Select either of the setup options presented in Run Multiple Instances of SharePlex on page 48. These procedures will guide you through the steps to establish independent instances of SharePlex on the target. If you already installed SharePlex on the target, a variable-data directory, database account, and port number already exist. You can dedicate that SharePlex instance to one of the source systems, and then create additional instances on the target per those instructions.

Steps on the source systems

Install one instance of SharePlex on each source system, as directed in the SharePlex Installation Guide. Match the port numbers of those instances to the port numbers of their associated target variable-data directories. If you already installed SharePlex on the source systems, you can change the port numbers as needed. For more information, see Set the SharePlex Port Number on page 400.

Configuration

Create a configuration file on each source system that replicates the objects from that system to the central target. For more information about how to create a configuration file, see Configure SharePlex to Replicate Data on page 60.

datasource_specification		
source_specification	*target_specification*	*central_host*[**@***db*]

where:

- *source_specification* is the fully qualified name of a source object (owner.object) or a wildcarded specification.
- *target_specification* is the fully qualified name of a target object (owner.object) or a wildcarded specification.
- *central_host* is the target system.
- *db* is a database specification. The database specification consists of either **o.** or **r.** prepended to the Oracle SID, TNS alias, or database name, as appropriate for the connection type. A database identifier is not required if the target is JMS, Kafka or a file.

Example

This example shows data from datasource **oraA** on **hostA** and datasource **oraB** on **hostB** replicating to **oraC** on system **hostC**.

Data from hostA

Datasource:o.oraA		
hr.*	**hr.***	**hostC@o.oraC**
fin.*	**fin***	**hostC@o.oraC**

Data from hostB

Datasource:o.oraA		
cust.*	**hr.***	**hostC@o.oraC**
mfg.*	**mfg.***	**hostC@o.oraC**

Recommended target configuration

Each source system in a consolidated configuration sends a discrete data stream that flows to its own Post process on the target. You can assign a unique identifier of your choosing to each source system, and then configure the Post process to include that identifier in each insert or update that it posts on the target.

By identifying rows in this manner, your environment is prepared to support the SharePlex **compare** and **repair** commands (which require a source ID) as well as any other work that may require the selection or identification of rows by their source. The **compare** and **repair** processes will use the source ID value to select only the rows that are valid for that source.

To configure each Post to write a source ID:

1. Create or alter the target table to include a column named SHAREPLEX_SOURCE_ID. This is the column that will contain the source ID value.

   NOTE: You can change this name by running the **target** command with the **set metadata** option, before continuing further. See the SharePlex Reference Guide for more information.

2. Choose a unique ID for each of the source systems. Any single alphanumeric string is permitted.
3. On the target, run **sp_ctrl** for each Post process.
4. For each Post process, issue the **target** command with the **set source** option. This command sets the source ID that will be posted by that Post process. The following example shows the command for three Post processes:

   `sp_ctrl>` **target sys4 queue Q1 set source east**

   `sp_ctrl>` **target sys4 queue Q2 set source central**

   `sp_ctrl>` **target sys4 queue Q3 set source west**

Configure Peer-to-Peer Replication

These instructions show you how to set up SharePlex for the purpose of maintaining multiple databases, where applications on each system can make changes to the same data, while SharePlex keeps all of the data synchronized through replication. This is known as *peer-to-peer*, or *active-active*, replication. In this strategy, the databases are usually mirror images of each other, with all objects existing in their entirety on all systems. Although similar in benefit to a high-availability strategy, the difference between the two is that peer-to-peer allows concurrent changes to the same data, while high availability permits changes to the secondary database only in the event that the primary database goes offline.

This strategy supports the following business requirements:

- Maintain the availability of mission-critical data by operating multiple instances in different locations.
- Distribute heavy online transaction processing application (OLTP) loads among multiple points of access.
- Limit direct access to an important database, while still enabling users outside a firewall to make updates to their own copies of the data.

An example of peer-to-peer replication is an e-commerce company with three identical databases. When users access the application from a web browser, the web server connects to any of those databases sequentially in a round-robin configuration. If one of the databases is unavailable, the server connects to a different available database server. Thus the configuration serves not only as a failover resource, but also as a means of distributing the load evenly among all the peers. Should the company need to produce business reports, user access to one of the databases can be stopped temporarily, and that database can be used to run the reports.

NOTE: Data changes made in peer-to-peer replication are prevented from looping back from one machine to another because Capture ignores transactions performed on the local system by the Post process.

Peer-to-peer replication is not appropriate for all replication environments. It requires a major commitment to database design that might not be practical when packaged applications are in use. It also requires the development of conflict resolution routines to prioritize which transaction SharePlex posts to any given database if there are multiple changes to the same data at or near the same time.

Supported source-target combinations

- Oracle to Oracle
- Oracle to PostgreSQL
- Oracle to PostgreSQL Database as a Service as source
- PostgreSQL to PostgreSQL
- PostgreSQL to Oracle
- PostgreSQL to PostgreSQL Database as a Service
- PostgreSQL Database as a Service as source to PostgreSQL
- PostgreSQL Database as a Service as source to Oracle
- PostgreSQL Database as a Service as source to PostgreSQL Database as a Service

Capabilities

This replication strategy supports the following:

- Use of named export and post queues

This replication strategy does not support the following:

- Replication of LOBs. If tables with LOBs are included in replication the LOBs will be bypassed by conflict resolution, causing the potential for data to be out of synchronization.
- Column mapping and partitioned replication is not appropriate in a peer-to-peer configuration.

Requirements

- Every table involved in peer-to-peer replication must have a primary key or a unique key with no nullable columns. Each key must uniquely identify the same *owner.table.row* among all of the databases that will be involved in replication, and the logging of the key columns must be enabled in the database. See additional requirements in this topic.
- Prepare the system, install SharePlex, and configure database accounts according to the instructions in the SharePlex Installation Guide.
- Enable supplemental logging for primary keys, unique keys, and foreign keys on all databases in the peer-to-peer configuration.
- Enable archive logging on all systems.
- You must understand the concepts of synchronization. For more information, see Understand the Concept of Synchronization on page 35.
- Set the SP_OPX_CREATE_ORIGIN_PG to 1 before activation. Set it on the PostgreSQL peer for PostgreSQL to Oracle replication and on both peers for PostgreSQL to PostgreSQL replication.

Overview

In peer-to-peer replication, DML changes are allowed on copies of the same tables in different databases, usually on different systems, while SharePlex keeps them all current through replication. If a record is changed in more than one database at (or near) the same time, conflicts can occur, and conflict-resolution logic must be applied to resolve the discrepancy.

What causes a conflict in peer-to-peer replication?

To understand how SharePlex determines a conflict, refer to the following examples of normal and conflict situations. In the examples, three systems (SysA, SysB and SysC) are used. For the detailed information about what is a conflict, see What is a conflict?

The following tables are used in the example:

Scott.employee_source

jane.employee_backup

The column names and definitions are identical:

```
EmpNo     number(4) not null,
SocSec    number(11) not null,
EmpName   char(30),
Job       char(10),
Salary    number(7,2),
Dept      number(2)
```

The values for both tables in a synchronized state are:

EmpNo (key)	SocSec	EmpName	Job	Salary	Dept
1	111-22-3333	Mary Smith	Manager	50000	1
2	111-33-4444	John Doe	Data Entry	20000	2
3	000-11-2222	Mike Jones	Assistant	30000	3
4	000-44-7777	Dave Brown	Manager	45000	3

Example of peer-to-peer replication without a conflict

1. At 9:00 in the morning, UserA on SysA changes the value of the Dept column to 2, where EmpNo is 1. SharePlex replicates that change to SysB and SysC, and both databases remain synchronized.
2. At 9:30 that same morning, UserB on SysB changes the value of Dept to 3, where EmpNo is 1. SharePlex replicates that change to SysA and SysC, and the databases are still synchronized.

Now the row looks like this:

EmpNo (key)	SocSec	EmpName	Job	Salary	Dept
1	111-22-3333	Mary Smith	Manager	50000	3

Example of peer-to-peer replication with an UPDATE conflict

1. At 11:00 in the morning, UserA on SysA updates the value of **Dept** to 1, where **EmpNo** is 1. At 11:02 that morning, the network fails. Captured changes rest in the export queues on all systems.
2. At 11:05 that morning, **before** the network is restored, UserB on SysB updates the value of **Dept** to 2, where **EmpNo** is 1. The network is restored at 11:10 that morning. Replication data transmission resumes.
3. When SharePlex attempts to post the change from UserA to the database on SysB, it expects the value in the **Dept** column to be 3 (the pre-image), but the value is 2 because of the change made by UserB. Because the pre-images do not match, SharePlex generates an out-of-sync error.
4. When SharePlex attempts to post the change from UserB to SysA, it expects the value of the column to be 3, but the value is 1 because of the change made by UserA. SharePlex generates an out-of-sync error.
5. When SharePlex attempts to post the changes made by UserA and User B to the database on SysC, both of those statements fail because the pre-images do not match. SharePlex generates an out-of-sync error.

NOTE: For more information, see Appendix A: Peer-To-Peer Diagram on page 417.

Deployment

To deploy peer-to-peer replication, perform the following tasks:

1. Evaluate the data for suitability to a peer-to-peer environment. Make any recommended alterations. For more information, see Evaluate the data on page 168.
2. Configure SharePlex so that data from each system replicates to all other systems in the peer-to-peer environment. For more information, see Configure Oracle to Oracle Replication on page 173.
3. Develop conflict resolution routines that provide rules for how Post handles conflicts. For more information, see Set up conflict resolution routines on page 174.
4. Create a conflict resolution file. SharePlex refers to this file to determine the correct procedure to use when a conflict occurs. For more information, see Configure Peer-to-Peer Replication on page 165.

Evaluate the data

To successfully deploy SharePlex in a peer-to-peer configuration, you must be able to:

- isolate keys
- prevent changes to keys
- control sequence generation
- control trigger usage
- eliminate cascading deletes
- designate a trusted host
- define priorities

These requirements must be considered during the architectural phase of the project, because they demand cooperation with the application. Consequently, many packaged applications are not suitable for a peer-to-peer deployment because they were not created within those guidelines.

Following are more detailed explanations of each of the requirements.

Keys

The only acceptable key in peer-to-peer replication is a primary key. If a table has no primary key but has a unique, not-NULL key, you can convert that key to a primary key. LONG columns cannot be part of the key.

If you cannot assign a primary key, and you know all rows are unique, you can create a unique index on all tables.

The primary key must be unique among all of the databases in the peer-to-peer replication network, meaning:

- it must use the same column(s) in each corresponding table in all databases.
- key columns for corresponding rows must have the same values.

The primary key must be created to contain enough information about a row so there can be no question about the uniqueness of that row, and so that there will be a conflict if a replicated operation would violate uniqueness.

The primary key value cannot be changed.

Supplemental logging of primary and unique keys must be enabled in the database.

Using only a sequence as the primary key probably will not suffice for peer-to-peer replication. For example, suppose the sample table uses sequences to generate values for key column EmpNo. Suppose UserA gets the next sequence value on SysA and inserts a row for “Jane Wilson.” UserB gets the next sequence value on SysB and also inserts a row for “Jane Wilson.” Even if the sequence numbers are different on each system, so there are no unique key violations on the replicated INSERTs, data integrity is compromised because there are now two entries for “Jane Wilson” in the databases, each with a different key. Subsequent UPDATEs will fail. The solution is to include other unique columns in the key, so that there is enough information to ensure uniqueness and ensure a conflict that can then be resolved through resolution logic.

Sequences

SharePlex does not support peer-to-peer replication of sequences. If the application uses sequences to generate all or part of a key, there must be no chance for the same range of values to be generated on any other system in the peer-to-peer configuration. You can use a sequence server or you can maintain sequences separately on each server and make sure you partition a unique range to each one. Quest recommends using *n*+1 sequence generation (where *n* = the number of systems in replication). Depending on the type of application, you can add a location identifier such as the system name to the sequence value in the primary key to enforce uniqueness.

Triggers

DML changes resulting from triggers firing on a source system enter the redo log and are replicated to the target system by SharePlex. If the same triggers fire on the target system, they return out-of-sync errors.

To handle triggers in a peer-to-peer configuration, you can do either of the following:

- Disable the triggers.
- Keep them enabled, but alter them to ignore the SharePlex user on all instances in the peer-to-peer configuration. SharePlex provides the **sp_add_trigger.sql** script for this purpose. This script puts a WHEN clause into the procedural statement of the trigger that tells it to ignore the Post process. For more information about this script, see the SharePlex Reference Guide.

ON DELETE CASCADE constraints

ON DELETE CASCADE constraints can remain enabled on all instances in the peer-to-peer replication configuration, but you must set the following parameters to direct Post to ignore those constraints:

- SP_OPO_DEPENDENCY_CHECK parameter to 2
- SP_OCT_REDUCED_KEY parameter to 0
- SP_OPO_REDUCED_KEY parameter to 0 (although in other replication scenarios this parameter can be set to different levels, it must be set to 0 in a peer-to-peer configuration)

Balance values maintained by using UPDATEs

Applications that use UPDATE statements to record changes in quantity, such as inventory or account balances, pose a challenge for peer-to-peer replication. The following example of an online bookseller explains the reason why.

The bookseller’s Inventory table contains the following columns.

Book_ID (primary key)

Quantity

Suppose the following sequence of events takes place:

1. A customer buys a book through the database on one server. The quantity on hand reduces from 100 books to 99. SharePlex replicates that UPDATE statement to the other server. (UPDATE inventory SET quantity = 99 WHERE book_ID = 51295).
2. Before the original UPDATE arrives, another customer buys two copies of the same book on another server (UPDATE inventory SET quantity = 98 WHERE book_ID = 51295), and the quantity on that server reduces from 100 books to 98.
3. When the Post process attempts to post the first transaction, it determines that the pre-image (100 books) on the first system does not match the expected value on the second system (it is now 98 as a result of the second transaction). Post returns an out-of-sync error.

A conflict resolution procedure could be written, but how would the correct value be determined? The correct value in both databases after the two transactions should be 97 books, but no matter which of the two UPDATE statements is accepted, the result is incorrect.

For this reason, peer-to-peer replication is not recommended for applications maintaining account or inventory balances using UPDATEs. If you can use a debit/credit method of maintaining balances, you can use INSERT statements (INSERT into inventory values “n”,...) instead of UPDATE statements. INSERT statements do not require a before-and-after comparison with a WHERE clause, as do UPDATE statements.

If your application must use UPDATE statements, you can write a conflict resolution procedure to determine the absolute (or net) change resulting from different UPDATE statements on different systems. For example, in the case of the preceding online bookseller example, when the first customer’s purchase is replicated to the second system, the following conflict resolution procedure fires:

if existing_row.quantity <> old.quantity then old.quantity - new.quantity = quantity_change; update existing_row set quantity = existing_row.quantity - quantity_change;

The conflict resolution logic tells SharePlex that, if the quantity value of the existing row in the target database (98) does not equal the old value (pre-image of 100), then subtract the new value (the replicated value of 99) from the pre-image to get the net change (1). Then, issue an UPDATE statement that sets the Quantity column to 98-1, which equals 97.

When the second user’s change is replicated to the first system, the same conflict resolution procedure fires. In this case, the net change (pre-image of 100 minus the new value of 98) is 2. The UPDATE statement on this system also results in a value of 97, which is 99 (the existing row value after the first customer’s purchase) minus the net change of 2. The result of this procedure’s logic is that the Quantity columns on each system are updated to 97 books, the net effect of selling three books.

The following example illustrates this concept using an account balance within a financial record:

account_number (primary key)

balance

1. Suppose a row (an account) in the example table has a balance of $1500 on SysA. CustomerA makes a deposit of $500 on that system. The application uses an UPDATE statement to change the balance to $2000. The change is replicated to SysB as an UPDATE statement (such as UPDATE...SET balance=$2000 WHERE account_number=51295).

2. Before the change arrives, CustomerA's spouse makes a withdrawal of $250 on SysB, and the application updates the database on that system to $1250. When CustomerA's transaction arrives from SysA and Post attempts to post it to SysB, there is a conflict, since the pre-image from the source system is $1500, but the pre-image on the target is $1250 because of the spouse's transaction — not a match.

You can write a conflict resolution routine to accommodate this kind of transaction by calculating the absolute (or net) change in the account, then using that value to resolve the conflict. For example:

if existing_row.balance <> old.balance then old.balance - new.balance = balance_change; update existing_row set balance = existing_row.balance - balance_change;

The result of this procedure would be to update the account balance to $1750, the net effect of depositing $500 and withdrawing $250. On SysB, the routine directs SharePlex to subtract the new (replicated) balance of 2000 from the old balance of 1500 for a net change of -500. The UPDATE statement sets the balance value to 1250 - (-500) = 1750, the correct value.

On SysA, the replicated value of 1250 is subtracted from the old balance of 1500 to get the net change of 250. The UPDATE statement subtracts that value from the existing balance of 2000 to get the correct value of 1750.

Priority

When the environment is established to avoid or resolve conflict when SharePlex searches for the correct row to change, the only remaining conflict potential is on fact data — which change to accept when the values for the same column in the same row differ on two or more systems. For this, your application must be able to accept the addition of *timestamp* and *source* columns, with *source* being the name of the local system for the table.

The following explains how those columns play a vital role when using a conflict resolution routine to establish priority.

Trusted source

You must assign a particular database or server to be the prevailing, or trusted, source for two reasons:

- The conflict resolution routine has the potential to get quite large and complex the more systems you have. There are bound to be failures that require resynchronization at some point. One of the systems in the configuration must be considered the true source from which all other systems will be resynchronized if necessary.
- You can write your conflict resolution routines so that operations from the trusted source system take priority over conflicting operations from other systems. For example, changes on the server at corporate headquarters could take priority over the same changes made by a branch office.

Timestamp

It is recommended that you include a timestamp column in the tables and assign priority in the conflict resolution routine to the earliest or latest timestamp. However, the timestamp must not be part of a key, or it will cause conflicts. SharePlex cannot locate rows if a key value changes — and the key value will change if one of the columns is a timestamp.

For timestamp priority to work, you must make sure all of the servers involved agree on the date and time. Tables on servers in different time zones can use Greenwich Mean Time (GMT).

To handle the situation where servers involved are in different time zones, you can specify a 'TIMESTAMP WITH LOCAL TIME ZONE' column in tables to be used by the routine, and make sure that the 'DBTIMEZONE' of databases in peer to peer replication is the same.

The default date format for SharePlex conflict resolution is MMDDYYYY HH24MISS. Tables with default dates must use that format, or conflict resolution will return errors. Before creating a table with a default date, use the following command to change the date format in SQL*Plus.

ALTER SESSION SET nls_date_format = 'MMDDYYYYHH24MISS'

Configure Oracle to Oracle Replication

The configuration files on the systems in a peer-to-peer configuration are identical with the exception of the datasource specification and the routing.

Conventions used in the syntax

In the configuration syntax in this topic, the placeholders represent the following items in the environment. This documentation assumes three systems, but there can be more.

- *hostA* is the first system.
- *hostB* is the second system.
- *hostC* is the third system.
- *ownerA.object* is the fully qualified name of an object on *hostA* or a wildcarded specification.
- *ownerB.object* is the fully qualified name of an object on *hostB* or a wildcarded specification.
- *ownerC.object* is the fully qualified name of an object on *hostC* or a wildcarded specification.
- oraA is the Oracle instance on *hostA*.
- *oraB* is the Oracle instance on *hostB*.
- *oraC* is the Oracle instance on *hostC*.

IMPORTANT! See Configure SharePlex to Replicate Data on page 60 for more information about the components of a configuration file.

Configuration on hostA

Datasource:o.*oraA*		
ownerA.object	*ownerB.object*	*hostB***@o.***oraB*
ownerA.object	*ownerB.object*	*hostB***@o.***oraB*
ownerA.object	*ownerC.object*	*hostC***@o.***oraC*
ownerA.object	*ownerC.object*	*hostC***@o.***oraC*

NOTE: If all owner names and table names are the same on all systems, you can use a compound routing map for each of these configuration files.

For example, the compound routing for replication from *hostA* is as follows:

Datasource:o.*oraA*		
owner.object	*owner.object*	*hostB***@o.***oraB***+***hostC***@o.***oraC*

Configuration on hostB

Datasource:o.*oraB*		
ownerB.object	*ownerA.object*	*hostA***@o.***oraA*
ownerB.object	*ownerA.object*	*hostA***@o.***oraA*
ownerB.object	*ownerC.object*	*hostC***@o.***oraC*
ownerB.object	*ownerC.object*	*hostC***@o.***oraC*

Configuration on hostC

Datasource:o.*oraC*		
ownerC.object	*ownerA.object*	*hostA***@o.***oraA*
ownerC.object	*ownerA.object*	*hostA***@o.***oraA*
ownerC.object	*ownerB.object*	*hostB***@o.***oraB*
ownerC.object	*ownerB.object*	*hostB***@o.***oraB*

Example

Datasource:o.oraA		
hr.emp	**hr.emp**	**hostB@o.oraB**
hr.sal	**hr.sal**	**hostB@o.oraB**
cust.%	**cust.%**	**hostB@o.oraB**

Set up conflict resolution routines

For information on setting up the conflict resolution routines for Oracle to Oracle, see User defined conflict resolution routines for Oracle to Oracle.

Configure PostgreSQL or PostgreSQL Database as a Service to PostgreSQL Replication

The configuration files on the systems in a peer-to-peer configuration are identical with the exception of the datasource specification and the routing.

Conventions used in the syntax

In the configuration syntax in this topic, the placeholders represent the following items in the environment. This documentation assumes three systems, but there can be more.

- *hostA* is the first system.
- *hostB* is the second system.
- *schemaA.object* is the fully qualified name of an object on *hostA* or a wildcarded specification.
- *schemaB.object* is the fully qualified name of an object on *hostB* or a wildcarded specification.

Configuration on hostA

Datasource:r.demoA		
schemaA.object	*schemaB.object*	*hostB***@r.***demoB*
schemaA.object	*schemaB.object*	*hostB***@r.***demoB*

Configuration on hostB

Datasource:r.*demoB*		
schemaB.object	*schemaA.object*	*hostA***@r.***demoA*
schemaB.object	*schemaA.object*	*hostA***@r.***demoA*

Example

Datasource:r.demoA		
hr.emp	**hr.emp**	**hostB@r.demoB**
hr.sal	**hr.sal**	**hostB@r.demoB**

Set up conflict resolution routines

For information on setting up the conflict resolution routines for PostgreSQL or PostgreSQL Database as a Service to PostgreSQL, see User defined conflict resolution routines for PostgreSQL to PostgreSQL.

SharePlex prepared routines

For information on SharePlex prepared routines for PostgreSQL or PostgreSQL Database as a Service to PostgreSQL replication, see SharePlex prepared routines.

Configure PostgreSQL or PostgreSQL Database as a Service to Oracle replication

The configuration files on the systems in a peer-to-peer configuration are identical with the exception of the datasource specification and the routing.

Conventions used in the syntax

In the configuration syntax in this topic, the placeholders represent the following items in the environment. This documentation assumes three systems, but there can be more.

- *hostA* is the PostgreSQL system.
- *hostB* is the Oracle system.
- *SchemaA.object* is the fully qualified name of an object on *hostA* or a wildcarded specification.
- *ownerB.object* is the fully qualified name of an object on *hostB* or a wildcarded specification.

Configuration on hostA

Datasource:r.*dbname*		
schemaA.tablename	*ownerB.object*	*hostB***@o.***oraB*
schemaA.tablename	*ownerB.object*	*hostB***@o.***oraB*

Configuration on hostB

Datasource:o.*oraB*		
ownerB.object	*schemaA.tablename*	*hostA***@r.***dbname*
ownerB.object	*schemaA.tablename*	*hostA***@r.***dbname*

Example for HostA

Datasource:r.dbname		
"demo"."data2k"	**"demo"."data2k"**	**hostB@o.dbname**

Example for HostB

Datasource:o.dbname		
"demo"."data2k"	**"demo"."data2k"**	**hostB@r.dbname**

Set up conflict resolution routines

For information on setting up the conflict resolution routines for PostgreSQL or PostgreSQL Database as a Service to Oracle, see User defined conflict resolution routines for PostgreSQL to Oracle.

SharePlex prepared routines

For information on SharePlex prepared routines for PostgreSQL or PostgreSQL Database as a Service to Oracle replication, see SharePlex prepared routines.

Develop Conflict Resolution Routines

This section provides information on various user-defined conflict resolution routines.

What is a conflict?

A conflict is defined as an out-of-sync condition — source and target tables are not identical. You can predict that out-of-sync (conflict) situations will occur when a DML statement constructed by SharePlex fails to execute on a row in the target table because of the following reasons:

- Post applies a replicated INSERT but a row with the same key already exists in the target. Post applies the following logic:
    - If all of the current values in the target row are the same as the INSERT values, Post considers the rows to be in-sync and discards the operation.
    - If any of the values are different from those of the INSERT, Post considers this an out-of-sync condition.

  NOTE: You can configure Post so that it does not consider non-key values when posting an INSERT (applicable only when replicating data from Oracle to Oracle). See the `SP_OPO_SUPPRESSED_OOS` parameter in the SharePlex Reference Guide.

- Post applies a replicated UPDATE but either cannot find a row in the target with the same key value as the one in the UPDATE or Post finds the correct row but the row values do not match the before values in the UPDATE. Post applies the following logic:
    - If the current values in the target row match the after values of the UPDATE, Post considers the rows to be in-sync and discards the operation.
    - If the values in the target row do not match the before or after values of the UPDATE, Post considers this an out-of-sync condition.

  NOTE: You can configure Post so that it returns an out-of-sync message if the current values in the target row match the after values of the UPDATE (applicable only when replicating data from Oracle to Oracle). See the `SP_OPO_SUPPRESSED_OOS` parameter in the SharePlex Reference Guide.

- A DELETE is performed on the source data, but Post cannot locate the target row by using the key. When Post constructs its DELETE statement, it includes only the key value in its WHERE clause. If the row does not exist in the target, Post discards the operation.

User defined conflict resolution routines for Oracle to Oracle

To create conflict resolution routines, you write PL/SQL procedures that direct the action of SharePlex when a conflict occurs. Business rules vary widely from company to company, so it is impossible to create a standard set of conflict resolution rules and syntax that apply in every situation. You will probably need to write your own routines. It is good practice to write more than one procedure, such as making site or system priority the primary routine and timestamp a secondary routine. SharePlex invokes one routine after another until one succeeds or there are no more procedures available.

SharePlex provides the following tools that can be used as a basis for your routines:

- A generic PL/SQL interface that you can use to write basic routines based on DML operation types. For more information, see How to write a routine using the SharePlex generic interface on page 180.
- Packaged routines that perform basic conflict resolution based on a key or column value, which can be used as a backup measure in case the custom routines fail. For more information, see User defined conflict resolution routines for Oracle to Oracle on page 180.

IMPORTANT!

- This documentation provides guidelines, examples and templates to assist you, but do not use them as your own routines.
- Test your conflict resolution routines before you put them into production to make sure they work as intended, and to make sure that one routine does not counteract another one.
- By default, SharePlex does not stop for out-of-sync conditions. If failed attempts at conflict resolution are not resolved, the databases can become more and more out of synchronization. Check the Event Log frequently to monitor for out-of-sync warnings by using the **show log** command in **sp_ctrl**. See the SharePlex Reference Guide for more information about **show log** and other SharePlex commands.
- Updates are occasionally made to the conflict resolution logic, so refer to the Release Notes and documentation for your version of SharePlex for any additional information that augments or supersedes these instructions.

How to write a routine using the SharePlex generic interface

SharePlex provides a generic conflict resolution PL/SQL package that can be used to pass information to and from the procedural routines that you write.

NOTE: Custom routines are supported for Oracle to Oracle source-target combinations only.

Before you get started, understand the following guidelines:

- The same PL/SQL package is used for both generic conflict resolution and transformation (its name is **sp_cr**). **Use either generic conflict resolution or transformation for a table, but not both.** Transformed tables cannot be compared by SharePlex and conflict resolution cannot succeed. If both are used, SharePlex only calls the transformation routine. If appropriate, you can use generic conflict resolution and transformation for different tables in the same configuration. For more information, see Configure Data Transformation on page 230.
- Conflict resolution cannot be used for DDL changes.

- Any table to be accessed through PL/SQL for conflict resolution requires implicitly granted privileges from the owner of the object to SharePlex.
- Conflict resolution does not support changes to LONG or LOB columns.

NOTE: If you ran the SharePlex conflict resolution demonstration in the SharePlex Installation and Setup Guide, you can view a sample generic conflict resolution routine by viewing the **od_employee_gen** routine that was installed in the database used for the demonstration.

Procedure interface

Follow this template to create your procedure.

(**table_info in out***splex*.**sp_cr.row_typ**, **col_values in***splex*.**sp_cr.col_def_tabtyp**)

where:

- *splex* is the SharePlex schema.
- **sp_cr** is the name of the package that contains the PL/SQL record and table structures.
- **row_typ** is the name of the PL/SQL record that passes in/out variables (see Package definition).
- **col_def_type** is the name of the PL/SQL table that stores column information (see col_def_type table).

Package definition

SharePlex defines PL/SQL record and table structures in a public package named **sp_cr** in the SharePlex database schema. The package uses the following parameters.

```
CREATE or REPLACE PACKAGE sp_cr AS
     TYPE row_typ IS RECORD
     src_host VARCHAR2(32),
     src_ora_sid VARCHAR2(32),
     src_ora_time VARCHAR2(20),
     source_rowid VARCHAR2(20),
     target_rowid VARCHAR2(20),
     statement_type VARCHAR2(6),
     source_table VARCHAR2(78),
     target_table VARCHAR2(78),
     oracle_err NUMBER,
     status NUMBER,
     action NUMBER,
     reporting NUMBER
     );
```

IN variables

For each row operation that causes a conflict, SharePlex passes this metadata information to your procedure.

Variable	Description
src_host	The name of the source system (where the operation occurred). It is case-sensitive and is passed using the same case as on the source system, for example **SysA**. If there are named post queues in use on the target system, this variable consists of the name of the post queue, for example **postq1**. **NOTE:** Maximum length is 32 characters. A host name longer than 32 will be truncated to 32 characters.
src_ora_sid	The ORACLE_SID of the source database. It is case-sensitive and is passed in the same case as in the **oratab** file or V$PARAMETER table.
src_ora_time	The timestamp of the change record in the source redo log.
source_rowid	The row ID of the source row. It is passed as a literal within single quotes, for example '123456'.
target_rowid	The row ID of the corresponding row in the target database. SharePlex obtains the row ID by querying the target database. It is passed as a literal within single quotes, for example '123456'. If the row cannot be found using the PRIMARY key, the value is NULL.
statement_type	A letter, either I, U or D, indicating whether the operation is an INSERT, UPDATE or DELETE statement.
source_table	The owner and name of the source table, expressed as *owner.table*. This value is case-sensitive and matches the way the table is named in the database. It is passed within double quotes, for example "scott"."emp."
target_table	The owner and name of the target table, expressed as *owner.table*. This value is case-sensitive and matches the way the table is named in the database. It is passed within double quotes, for example "scott"."emp."
oracle_err	This is different, depending on whether the procedure is being used for conflict resolution or transformation. **Transformation**: SharePlex passes a value of 0 for this variable. This variable is only used for conflict resolution. **Conflict resolution:** The Oracle error number that caused the conflict.

OUT variables

These variables direct the action of SharePlex based on whether the procedure succeeded or failed).

Variable	Description
status	Defines whether or not the procedure succeeded. You must specify a value for this parameter.

Variable	Description
	• A value of 0 implies successful execution. It acts differently, depending on whether the procedure is used for conflict resolution or transformation. **Transformation**: Post does not write any SQL. SharePlex does not write any error messages to the Event Log when transformation succeeds. It continues its processing by reading the next replicated operation in the post queue. • **Conflict resolution**: A value of 0 directs SharePlex to proceed with the SQL statement. SharePlex does not write any log entries to the Event Log when conflict resolution succeeds. • A value of 1 implies that the procedure was unsuccessful. In this case, the action SharePlex takes depends on what you specified as the action variable. • (**Transformation** only) A value of 7 implies unsuccessful execution and instructs the Post process to stop.
action	Defines the action that you want SharePlex to take. This is different, depending on whether the procedure is used for transformation or conflict resolution. **Transformation**: You must specify a value of 0 for this parameter, which directs SharePlex NOT to post the SQL statement. Your transformation routine is responsible for posting the results of the transformation either to the target table or another table. The outcome of this action depends on what you specify for the reporting variable **Conflict resolution**: Specifies the action to take as a result of an *unsuccessful* conflict resolution procedure. You must specify a value for this parameter. • A value of 0 directs SharePlex NOT to post the SQL statement. The outcome of this action depends on what you specify for the **reporting** variable. In addition, it directs SharePlex to try the next conflict resolution procedure that you listed in the conflict resolution file, if one exists. • The value of 1 is reserved for internal SharePlex use. Do not use it. • A value of 2 directs SharePlex to try the next conflict resolution procedure that you listed in the conflict resolution file, if one exists.
reporting	Determines how SharePlex reports unsuccessful procedural results. You must specify a value for this parameter. • A value of 0 directs SharePlex NOT to report an error or write the failed SQL statement to the *SID*_**errlog.sql** log. • Values 1 and 2 are reserved for internal SharePlex use. Do not use them.

Variable	Description
	• A value of 3 directs SharePlex to write the failed SQL statement to the *SID*_**errlog.sql** log and report an error to the Event Log.

col_def_type table

SharePlex creates a **col_def_tabtyp** PL/SQL table for each replicated operation. This table stores column information. It is different depending on whether the procedure is used for transformation or conflict resolution.

- **Transformation**: For each row operation, SharePlex writes column information to **col_def_type**.
- **Conflict resolution**: For each row operation that causes a conflict, SharePlex writes column information to **col_def_tabtyp**.

All fields are passed by SharePlex to your routine, although not all will have values if SharePlex cannot locate the row.

Following is the data type that is used to populate the **col_def_tabtyp** table.

```
type col_def_typ is record
     (column_name user_tab_columns.column_name%type
     ,data type user_tab_columns.data type%type
     ,is_key boolean
     ,is_changed boolean
     ,old_value varchar2(32764)
     ,new_value varchar2(32764)
     ,current_value varchar2(32764)
     );
type col_def_tabtyp is table of col_def_typ
```

Description of col_def_tabtyp

Column	Description
column_name	Tells your procedure the name of the column that was replicated from the source table, for example **emp_last_name**. This value is not case-sensitive.
data type	Tells your procedure the data type of the data in the replicated column, for example VARCHAR2. This value is always in capital letters.
is_key	Tells your procedure whether or not the column is a key column. If it is a key column, SharePlex passes a value of **TRUE**. If the column is not part of a key, SharePlex passes a value of **FALSE**.
is_changed	Tells your procedure whether or not the column value has changed. If it is changed, SharePlex passes a value of **TRUE**. If the column is not changed, SharePlex passes a value of **FALSE**.

Column	Description
	• For INSERTs, **is_changed** is **TRUE** for non-NULL values, because none of the columns existed in the database. If a **NULL** value is inserted, **is_changed** is **FALSE**. • For UPDATEs, **is_changed** is **TRUE** for non-key columns. For key columns, **is_changed** normally is **FALSE**, but SharePlex will pass a value for a changed key column. **Conflict resolution only**: If a key value also was changed on the target system, SharePlex will not be able to locate the correct row, and conflict resolution could fail. • For DELETEs, **is_changed** is always **FALSE**, because SharePlex replicates only the key values for a DELETE statement.
old_value	Tells your procedure the old value of the replicated column, before it was changed on the source system. This column is **NULL** for INSERTs, because the row did not exist in the target database before the INSERT. **Conflict resolution only**: This is the pre-image against which SharePlex compared the source and target columns as part of its synchronization check for UPDATEs and DELETEs. If the old value passed by SharePlex does not match the **current_value** value obtained from the target row, then there is a conflict.
new_value	Tells your procedure the new value of the replicated column, as changed on the source system.
current_value	Tells your procedure the current value of the column in the target table. If SharePlex cannot locate the target row, the value is **NULL**.

Example entries in col_def_tabtyp table per operation type

The following tables illustrate the possible outcomes of each type of operation.

INSERT operation

column_name	is_changed	old_value	new_value	current_value[1]	is_key
C1	TRUE	NULL	bind	NULL	FALSE
C2	TRUE	NULL	bind	NULL	TRUE
C3	FALSE	NULL	NULL	NULL	TRUE | FALSE

[1] When an INSERT fails, it is because a row with the same PRIMARY key already exists in the target database. SharePlex does not return the current value for INSERTs.

UPDATE operation

column_name	is_changed	old_value	new_value	current_value[1, 2]	is_key
C1	TRUE	bind	bind	NULL | *target_value*	FALSE

column_name	is_changed	old_value	new_value	current_value[1, 2]	is_key
C2	FALSE	bind	NULL	NULL | *target_value*	TRUE
C3	TRUE	bind	bind	NULL | *target_value*	TRUE

[1] (Conflict resolution) When an UPDATE fails, it is because SharePlex cannot find the row by using the PRIMARY key and the pre-image. If the row cannot be found, SharePlex searches for the row by using only the PRIMARY key. If SharePlex finds the row, it returns the current value for the key column as well as the changed columns. If SharePlex cannot find the row by using just the PRIMARY key, then SharePlex returns a **NULL**.

[2] (Transformation) For an UPDATE, SharePlex cannot locate a row using the PRIMARY key and the pre-images, because the pre-images are different due to transformation. As an alternative, it searches for the row using just the PRIMARY key. If it finds it, SharePlex returns the current value for the key column as well as the changed columns. If it cannot locate the row using just the PRIMARY key, then current_value is NULL

DELETE operation

column_name	is_changed	old_value	new_value	current_value[1]	is_key
C1	FALSE	bind	NULL	NULL	TRUE

[1] When a DELETE fails, it is because SharePlex could not find the row by using the PRIMARY key. Therefore, SharePlex returns a NULL.

List the routines in conflict_resolution.*SID*

After you create the conflict resolution procedure(s), construct the conflict resolution file. This file tells SharePlex which procedures to use for which objects and operation types, and in which order.

Where to find the conflict resolution file

A blank **conflict_resolution**.*SID* file, where *SID* is the ORACLE_SID of the target instance, was included in the **data** sub-directory of the SharePlex variable-data directory when SharePlex was installed. Use the file on the target system.

If this file does not exist, you can create one in ASCII format in an ASCII text editor. It must be named **conflict_resolution**.*SID*, where *SID* is the ORACLE_SID of the target instance.

NOTE: the *SID* is case-sensitive.

IMPORTANT! There can be only one **conflict_resolution**.*SID* file per active configuration.

How to make entries in the conflict resolution file

Use the following template to link a procedure to one or more objects and operation types.

owner.object	**{i | u | d | iud}**	*owner.procedure*

where:

- *owner.object* is the owner and name of a *target* object, or a wildcarded entry. (See Syntax rules on page 187)
- **i**| **u** | **d** is the type of operation that creates the conflict that is resolved with the specified procedure. You can specify any or all operation types, for example **id** or **iud**. Upper or lower case are both valid.
- *owner.procedure* is the owner and name of the conflict resolution procedure that will handle the specified object and operation type.

Syntax rules

- There must be at least one space between the object specification, the operation type specification, and the procedure specification.
- You can use the **LIKE** operator and a SQL wildcard (%) to specify multiple objects by using a search string. (See the Example.)
- You can use an underscore (_) to denote a single-character wildcard. For table names that contain an underscore character (for example emp_sal), SharePlex recognizes the backslash (\) as an escape character to denote the underscore as a literal and not a wildcard, for example: **like:scott.%_corp_emp**. If you are not using the LIKE operator, the backslash escape character is not required if an object name contains an underscore.
- The order in which you list the procedures in the conflict resolution file determines their priority of use (in descending order). If you list a table-specific procedure, SharePlex uses it before procedures that are specified with a wildcarded object name.
- You a comment line anywhere in the file. Start a comment line with the pound symbol (#).

Example conflict resolution file

scott.sal	IUD	scott.sal_cr
like:scott.%_corp_emp	IUD	scott.emp_cr1
like:scott.%_corp_emp	IUD	scott.emp_cr2
like:scott%	IUD	scott.emp_cr3
scott.cust	U	scott.sal_cr

How it works:

- The **scott.sal_cr routine** is used for the **scott.sal** table before the **scott.emp_cr1** procedure is used for that table.
- The **scott.emp_cr1** procedure is used before the **scott.emp_cr2** procedure for all tables meeting the search criteria, and so forth.
- For **scott.cust**, a procedure is called for UPDATEs before the other routines are used for all operations.

How to specify SharePlex prepared routines in the conflict resolution file

To use the SharePlex prepared routines for all tables in the replication configuration, use the **!DEFAULT** parameter instead of specifying an owner and object name.

A custom routine takes priority over a SharePlex prepared routine. A prepared routine is used only if the custom routine fails. This is true regardless of the order in which the **!DEFAULT**-associated routine appears in the file.

The **!DEFAULT** parameter is case-sensitive. It must be typed in all capital letters.

In the following example, the **!UpdateUsingKeyOnly** procedure is used for UPDATEs and DELETEs for all tables, including **james.table1**, although the user-defined procedures listed for **james.table1** take precedence.

!DEFAULT	U	!UpdateUsingKeyOnly
!DEFAULT	D	!UpdateUsingKeyOnly
james.table1	U	james.procedure_upd
james.table1	I	james.procedure_ins
james.table1	D	james.procedure_del

How to change the conflict resolution file while replication is active

You can change the conflict resolution file any time during replication to add and remove tables and procedures. After you change the conflict resolution file, stop and re-start the Post process.

Log information about resolved conflicts for Oracle database

You can configure the Post process to log information about successful conflict resolution operations if you are using the SharePlex prepared routines. This feature is disabled by default. For more information, see Log information about resolved conflicts for Oracle database on page 203.

User defined conflict resolution routines for PostgreSQL or PostgreSQL Database as a Service to PostgreSQL

To create conflict resolution routines, you write PL/PGSQL procedures that direct the action of SharePlex when a conflict occurs. Business rules vary widely from company to company, so it is impossible to create a standard set of conflict resolution rules and syntax that apply in every situation. You will probably need to write your own routines. It is good practice to write more than one procedure, such as making site or system priority the primary routine and timestamp a secondary routine. SharePlex invokes one routine after another until one succeeds or there are no more procedures available.

SharePlex provides the following tools that can be used as a basis for your routines:

- A generic PL/PGSQL interface that you can use to write basic routines based on DML operation types. For more information, see How to write a routine using the SharePlex generic interface on page 189.
- The customized routines that perform basic conflict resolution based on a key or column value, which can be used as a backup measure in case the custom routines fail. For more information, see User defined conflict resolution routines for PostgreSQL or PostgreSQL Database as a Service to PostgreSQL on page 189.

IMPORTANT!

- This documentation provides guidelines, examples and templates to assist you, but do not use them as your own routines.
- Test your conflict resolution routines before you put them into production to make sure they work as intended, and to make sure that one routine does not counteract another one.
- By default, SharePlex does not stop for out-of-sync conditions. If failed attempts at conflict resolution are not resolved, the databases can become more and more out of synchronization. Check the Event Log frequently to monitor for out-of-sync warnings by using the **show log** command in **sp_ctrl**. See the SharePlex Reference Guide for more information about **show log** and other SharePlex commands.
- Updates are occasionally made to the conflict resolution logic, so refer to the Release Notes and documentation for your version of SharePlex for any additional information that augments or supersedes these instructions.

How to write a routine using the SharePlex generic interface

SharePlex provides a generic conflict resolution interface that can be used to pass information to and from the procedural routines that you write.

NOTE: Custom routines are supported from Oracle to PostgreSQL and PostgreSQL to PostgreSQL source-target combinations only.

Before you get started, understand the following guidelines:

- Conflict resolution cannot be used for DDL changes.
- Any table to be accessed through PL/PGSQL procedure for conflict resolution requires implicitly granted privileges from the schema of the object to SharePlex.

- Custom conflict resolution does not support large data types, such as CHAR > 2000, VARCHAR > 4000 or without length, TEXT, and BYTEA.

Procedure interface

Follow this template to create your procedure.

(table_info sp_cr.row_typ, col_values sp_cr.col_def_typ[], INOUT status INTEGER, INOUT action INTEGER, INOUT reporting INTEGER)

where:

- **sp_cr** is the name of the schema that contains the PL/PGSQL record and table structures.
- **row_typ** is the name of the PL/PGSQL record that passes in variables (see Schema definition).
- **col_def_typ** is the name of the PL/PGSQL table that stores column information (see col_def_typ type).

Schema definition

SharePlex defines PL/PGSQL record and table structures in a schema named **sp_cr** in the SharePlex database.

The schema uses the following parameters:

```
CREATE SCHEMA IF NOT EXISTS sp_cr;

CREATE TYPE sp_cr.row_typ AS

(src_host VARCHAR(32),

src_db VARCHAR(32),

src_time VARCHAR(20),

statement_type VARCHAR(6),

source_table VARCHAR(128),

target_table VARCHAR(128),

native_error INTEGER,

sql_state VARCHAR(10)

);
```

IN Variables

For each row operation that causes a conflict, SharePlex passes this metadata information to your procedure.

Variable	Description
src_host	The name of the source system (where the operation occurred). It is case-sensitive and is passed using the same case as on the source system, for example **SysA**. If there are named post queues in use on the target system, this variable consists of the name of the post queue, for example **postq1**. **Note: Maximum length is 32 characters. A host name longer than 32 will be truncated to 32 characters.**

Variable	Description
src_db	The database name of source.
src_time	The timestamp of the change record when it is received by the Capture process.
statement_type	A letter, either I, U or D, indicating whether the operation is an INSERT, UPDATE or DELETE statement.
source_table	The owner and name of the source table, expressed as *owner.table*. This value is case-sensitive and matches the way the table is named in the database.
target_table	The owner and name of the target table, expressed as *owner.table*. This value is case-sensitive and matches the way the table is named in the database.
native_error	This field is generate by ODBC API for conflicting DML.*
sql_state	This field is generate by ODBC API for conflicting DML.*

NOTE: *In case of Update and Delete operation, if row not found, then native error is set to 100 and SQL_state is set to '00000'.

OUT Variables

These variables direct the action of SharePlex based on whether the procedure succeeded or failed).

Variable	Description
status	Defines whether or not the procedure succeeded. You must specify a value for this parameter. • A value of 0 implies successful execution. It acts differently, depending on whether the procedure is used for conflict resolution. • **Conflict resolution**: A value of 0 directs SharePlex to proceed with the SQL statement. SharePlex does not write any log entries to the Event Log when conflict resolution succeeds. • A value of 1 implies that the procedure was unsuccessful. In this case, the action SharePlex takes depends on what you specified as the action variable.
action	Defines the action that you want SharePlex to take. This is different, depending on whether the procedure is used for conflict resolution. **Conflict resolution**: Specifies the action to take as a result of an *unsuccessful* conflict resolution procedure. You must specify a value for this parameter.

Variable	Description
	• A value of 0 directs SharePlex NOT to post the SQL statement. The outcome of this action depends on what you specify for the **reporting** variable. In addition, it directs SharePlex to try the next conflict resolution procedure that you listed in the conflict resolution file, if one exists. • The value of 1 is reserved for internal SharePlex use. Do not use it. • A value of 2 directs SharePlex to try the next conflict resolution procedure that you listed in the conflict resolution file, if one exists.
reporting	Determines how SharePlex reports unsuccessful procedural results. You must specify a value for this parameter. • A value of 0 directs SharePlex NOT to report an error or write the failed SQL statement to the *database*_**errlog.sql** log. • Values 1 and 2 are reserved for internal SharePlex use. Do not use them. • A value of 3 directs SharePlex to write the failed SQL statement to the *database*_**errlog.sql** log and report an error to the Event Log.

col_def_typ type

SharePlex creates a **col_def_typ** type for each replicated operation. This type stores column information.

- **Conflict resolution**: For each row operation that causes a conflict, SharePlex writes column information to **col_def_typ**.

All fields are passed by SharePlex to your routine, although not all will have values if SharePlex cannot locate the row.

Following is the data type that is used to populate the **col_def_typ** table.

type col_def_typ is record

(column_name user_tab_columns.column_name%type

,data type user_tab_columns.data type%type

```
CREATE TYPE sp_cr.col_def_typ AS

(column_name VARCHAR,

datatype VARCHAR,

is_key BOOLEAN,

is_changed BOOLEAN,

old_value VARCHAR ,

new_value VARCHAR

);
```

```
type col_def_tabtyp is table of col_def_typ
```

Description of col_def_typ

Column	Description
column_name	Tells your procedure the name of the column that was replicated from the source table, for example **emp_last_name**. This value is not case-sensitive.
data type	Tells your procedure the data type of the data in the replicated column, for example VARCHAR. This value is always in capital letters.
is_key	Tells your procedure whether or not the column is a key column. If it is a key column, SharePlex passes a value of **TRUE**. If the column is not part of a key, SharePlex passes a value of **FALSE**.
is_changed	Tells your procedure whether or not the column value has changed. If it is changed, SharePlex passes a value of **TRUE**. If the column is not changed, SharePlex passes a value of **FALSE**. • For INSERTs, **is_changed** is **TRUE** for non-NULL values, because none of the columns existed in the database. If a **NULL** value is inserted, **is_changed** is **FALSE**. • For UPDATEs, **is_changed** is **TRUE** for non-key columns. For key columns, **is_changed** normally is **FALSE**, but SharePlex will pass a value for a changed key column. **Conflict resolution only**: If a key value also was changed on the target system, SharePlex will not be able to locate the correct row, and conflict resolution could fail. • For DELETEs, **is_changed** is always **FALSE**, because SharePlex replicates only the key values for a DELETE statement.
old_value	Tells your procedure the old value of the replicated column, before it was changed on the source system. This column is **NULL** for INSERTs, because the row did not exist in the target database before the INSERT. **Conflict resolution only**: This is the pre-image against which SharePlex compared the source and target columns as part of its synchronization check for UPDATEs and DELETEs. If the old value passed by SharePlex does not match the **current_value** value obtained from the target row, then there is a conflict.
new_value	Tells your procedure the new value of the replicated column, as changed on the source system.

Example entries in col_def_typ table per operation type

The following tables illustrate the possible outcomes of each type of operation.

INSERT operation

column_name	is_changed	old_value	new_value	is_key
C1	TRUE	NULL	bind	FALSE
C2	TRUE	NULL	bind	TRUE
C3	FALSE	NULL	NULL	TRUE | FALSE

UPDATE operation

column_name	is_changed	old_value	new_value	is_key
C1	TRUE	bind	bind	FALSE
C2	FALSE	bind	NULL	TRUE
C3	TRUE	bind	bind	TRUE

DELETE operation

column_name	is_changed	old_value	new_value	is_key
C1	FALSE	bind	NULL	TRUE

List the routines in conflict_resolution.*dbname*

After you create the conflict resolution procedure(s), construct the conflict resolution file. This file tells SharePlex which procedures to use for which objects and operation types, and in which order.

Where to find the conflict resolution file

A blank `conflict_resolution.testdb`, where `testdb` is a DB name, was included in the **data** sub-directory of the SharePlex variable-data directory when SharePlex was installed. Use the file on the target system.

If this file does not exist, you can create one in ASCII format in an ASCII text editor. It must be named **conflict_resolution**.*DB*, where *DB* is the database name.

IMPORTANT! There can be only one **conflict_resolution**.*DB* file per active configuration.

How to make entries in the conflict resolution file

Use the following template to link a procedure to one or more objects and operation types.

SchemaName.tableName	IUD	*schema.procedure*

where:

- IUD is the type of operation that creates the conflict that is resolved with the specified procedure.
- *schema*.*procedure* is the schema and name of the conflict resolution procedure that will handle the specified object and operation type.

Syntax rules

- There must be at least one space between the object specification, the operation type specification, and the procedure specification.
- The order in which you list the procedures in the conflict resolution file determines their priority of use (in descending order). If you list a table-specific procedure, SharePlex uses it before procedures that are specified with a wildcarded object name.

Example conflict resolution file

scott.sal	IUD	scott.sal_cr
scott.cust	IUD	scott.cust_cr
!DEFAULT	IUD	scott.sal_cr5

How it works:

- The **scott.sal_cr routine** is used for the **scott.sal** table before the **scott.sal_cr5** procedure is used for that table.
- For **scott.cust**, a procedure **scott.cust_cr** is called for IUD before the default routines are used for all operations.

How to specify SharePlex prepared routines in the conflict resolution file

To use the SharePlex prepared routines for all tables in the replication configuration, use the **!DEFAULT** parameter instead of specifying an schema and object name.

A custom routine takes priority over a SharePlex prepared routine. A prepared routine is used only if the custom routine fails. This is true regardless of the order in which the **!DEFAULT**-associated routine appears in the file.

The **!DEFAULT** parameter is case-sensitive. It must be typed in all capital letters.

!DEFAULT	IUD	proc0
schema.table1	IUD	proc1
james.table1	IUD	proc2
james.table1	IUD	proc3

How to change the conflict resolution file while replication is active

You can change the conflict resolution file any time during replication to add and remove tables and procedures. After you change the conflict resolution file, stop and re-start the Post process.

Log information about resolved conflicts for PostgreSQL database

You can configure the Post process to log information about successful conflict resolution operations if you are using the SharePlex prepared routines. This feature is disabled by default. For more information, see Log information about resolved conflicts for PostgreSQL database on page 205.

User defined conflict resolution routines for PostgreSQL or PostgreSQL Database as a Service to Oracle

To create conflict resolution routines, you write PL/SQL procedures that direct the action of SharePlex when a conflict occurs. Business rules vary widely from company to company, so it is impossible to create a standard set of conflict resolution rules and syntax that apply in every situation. You will probably need to write your own routines. It is good practice to write more than one procedure, such as making site or system priority the primary routine and timestamp a secondary routine. SharePlex invokes one routine after another until one succeeds or there are no more procedures available.

SharePlex provides the following tools that can be used as a basis for your routines:

- A generic PL/SQL interface that you can use to write basic routines based on DML operation types. For more information, see How to write a routine using the SharePlex generic interface on page 196.
- Packaged routines that perform basic conflict resolution based on a key or column value, which can be used as a backup measure in case the custom routines fail. For more information, see User defined conflict resolution routines for PostgreSQL or PostgreSQL Database as a Service to Oracle on page 196.

IMPORTANT!

- This documentation provides guidelines, examples and templates to assist you, but do not use them as your own routines.
- Test your conflict resolution routines before you put them into production to make sure they work as intended, and to make sure that one routine does not counteract another one.
- By default, SharePlex does not stop for out-of-sync conditions. If failed attempts at conflict resolution are not resolved, the databases can become more and more out of synchronization. Check the Event Log frequently to monitor for out-of-sync warnings by using the **show log** command in **sp_ctrl**. See the SharePlex Reference Guide for more information about **show log** and other SharePlex commands.
- Updates are occasionally made to the conflict resolution logic, so refer to the Release Notes and documentation for your version of SharePlex for any additional information that augments or supersedes these instructions.

How to write a routine using the SharePlex generic interface

SharePlex provides a generic conflict resolution PL/SQL package that can be used to pass information to and from the procedural routines that you write.

Before you get started, understand that any table to be accessed through PL/SQL for conflict resolution requires implicitly granted privileges from the owner of the object to SharePlex. For additional guidelines for Oracle, see the How to write a routine for oracle section and for PostgreSQL, see the How to write a routine section.

NOTE: If you ran the SharePlex conflict resolution demonstration in the SharePlex Installation and Setup Guide, you can view a sample generic conflict resolution routine by viewing the **od_employee_gen** routine that was installed in the database used for the demonstration.

Procedure interface

For the procedure interface information of Oracle, see Procedure interface for Oracle.

For the procedure interface information of PostgreSQL, see Procedure interface for PostgreSQL.

List the routines in conflict_resolution.*sid*

After you create the conflict resolution procedure(s), construct the conflict resolution file. This file tells SharePlex which procedures to use for which objects and operation types, and in which order.

Where to find the conflict resolution file

A blank `conflict_resolution.oraA`, where `oraA` is a sid of OracleDB, was included in the **data** sub-directory of the SharePlex variable-data directory when SharePlex was installed. Use the file on the target system.

If this file does not exist, you can create one in ASCII format in an ASCII text editor. It must be named **conflict_resolution**.`oraA`, where `oraA` is a sid of OracleDB.

IMPORTANT! There can be only one **conflict_resolution**.sid file per active configuration.

How to make entries in the conflict resolution file

Use the following template to link a procedure to one or more objects and operation types.

SchemaName.tableName	IUD	*schema.procedure*

where:

- IUD is the type of operation that creates the conflict that is resolved with the specified procedure.
- *schema.procedure* is the schema and name of the conflict resolution procedure that will handle the specified object and operation type.

Syntax rules

- There must be at least one space between the object specification, the operation type specification, and the procedure specification.
- The order in which you list the procedures in the conflict resolution file determines their priority of use (in descending order). If you list a table-specific procedure, SharePlex uses it before procedures that are specified with a wildcarded object name.

Example conflict resolution file

scott.sal	IUD	scott.sal_cr
scott.cust	IUD	scott.cust_cr
!DEFAULT	IUD	scott.sal_cr5

How it works:

- The **scott.sal_cr routine** is used for the **scott.sal** table before the **scott.sal_cr5** procedure is used for that table.
- For **scott.cust**, a procedure **scott.cust_cr** is called for IUD before the default routines are used for all operations.

How to specify SharePlex prepared routines in the conflict resolution file

To use the SharePlex prepared routines for all tables in the replication configuration, use the **!DEFAULT** parameter instead of specifying an schema and object name.

A custom routine takes priority over a SharePlex prepared routine. A prepared routine is used only if the custom routine fails. This is true regardless of the order in which the **!DEFAULT**-associated routine appears in the file.

The **!DEFAULT** parameter is case-sensitive. It must be typed in all capital letters.

!DEFAULT	IUD	proc0
schema.table1	IUD	proc1
james.table1	IUD	proc2
james.table1	IUD	proc3

How to change the conflict resolution file while replication is active

You can change the conflict resolution file any time during replication to add and remove tables and procedures. After you change the conflict resolution file, stop and re-start the Post process.

Log information about resolved conflicts for Oracle database

You can configure the Post process to log information about successful conflict resolution operations if you are using the SharePlex prepared routines. This feature is disabled by default. For more information, see Log information about resolved conflicts for Oracle database on page 203.

SharePlex Prepared Routines

SharePlex provides optional prepared routines for use in conjunction with custom routines. These options can be used with basic and generic conflict resolution formats. There are no limitations on column types.

Supplemental logging of primary and unique keys must be enabled in the database.

Supported source and target database combinations

- Oracle to Oracle
- PostgreSQL to PostgreSQL
- PostgreSQL to Oracle
- PostgreSQL Database as a Service to PostgreSQL Database as a Service
- PostgreSQL Database as a Service to Oracle
- PostgreSQL Database as a Service to PostgreSQL

Considerations

Review the following considerations before implementing SharePlex prepared routines.

!HostPriority

This prepared conflict resolution routine works for INSERT, UPDATE, and DELETE operations. It provides host-based conflict resolution by assigning priority to the row change that originated on the trusted source system. To define the trusted source, set the SP_OPO_TRUSTED_SOURCE (Oracle source) or SP_OPX_TRUSTED_SOURCE (PostgreSQL source) parameter to the name of the source system.

Resolution logic

Operation	Resolution Action
INSERT	If the source is the one specified with SP_OPO_TRUSTED_SOURCE (Oracle source) or SP_OPX_TRUSTED_SOURCE (PostgreSQL source), convert the INSERT to an UPDATE and overwrite the existing row. Otherwise, discard the change record and do nothing to the target row.
UPDATE	If the source is the one specified with SP_OPO_TRUSTED_SOURCE (Oracle source) or SP_OPX_TRUSTED_SOURCE (PostgreSQL source), overwrite the existing row using an UPDATE and use only the key columns in the WHERE clause. Otherwise, discard the change record and do nothing to the target row.
DELETE	Ignore the out-of-sync error and do nothing to the target row.

Syntax in conflict resolution file

```
owner.table   {I | U | D}   !HostPriority
```

!LeastRecentRecord

This prepared routine works for INSERT, UPDATE, and DELETE operations. It provides time-based conflict resolution by assigning priority to the least recent row change, as determined by a timestamp.

To capture the timestamp, tables using this routine must have a non-NULL timestamp column that is updated with every INSERT and UPDATE on the table. If the timestamp column in the DML, or in the existing row, is NULL, this routine cannot resolve the conflict.

This routine requires the SP_OCT_REDUCED_KEY parameter for Oracle and SP_CAP_REDUCED_KEY parameter for PostgreSQL to be set to 0 on the source system, so that all of the pre-image values of UPDATES are available to the Post process.

Resolution logic

Operation	Resolution Action
INSERT and UPDATE	• If the value of the timestamp column of the replicated operation is **greater than or equal to** the timestamp column of the row in the target, discard the replicated operation and do nothing to the target row. • If the timestamp column of the replicated operation is **less than** the timestamp column of the row in the target, overwrite the existing row using an UPDATE and use only the key columns in the WHERE clause.
DELETE	Ignore the conflict (out-of-sync message).

Syntax in conflict resolution file

```
owner.table   {I | U | D}   !LeastRecentRecord(col_name)
```

Where *col_name* is the timestamp column to be used by the routine.

NOTES:

- The recommended data type for `col_name` in a PostgresSQL database is timestamp, as DATE data type does not store time value. As a result, if a date data type is used, a conflict will not be resolved if the same date is updated at both peers at the same time
- The case sensitivity of the `col_name` for the Oracle peer depends on the source database.
  - If the data is being replicated from Oracle to Oracle, the `col_name` should be in uppercase.
  - If the data is being replicated from PostgreSQL to Oracle, the `col_name` should be in lowercase.
  - If the data is being replicated from both Oracle and PostgreSQL sources to Oracle, there should be two different entries of conflict resolution routines for `col_name` – one in uppercase and the other in lowercase.

!MostRecentRecord

This prepared routine works for INSERT, UPDATE, and DELETE operations. It provides time-based conflict resolution by assigning priority to the most recent row change, as determined by a timestamp.

To capture the timestamp, tables using this routine must have a non-NULL timestamp column that is updated with every INSERT and UPDATE on the table. If the timestamp column in the DML, or in the existing row, is NULL, this routine cannot resolve the conflict.

This routine requires the SP_OCT_REDUCED_KEY parameter for Oracle and SP_CAP_REDUCED_KEY parameter for PostgreSQL to be set to 0 on the source system, so that all of the pre-image values of UPDATES are available to the Post process.

Resolution logic

Operation	Resolution Action
INSERT and UPDATE	• If the timestamp of the replicated operation is **greater than** the timestamp of the row in the target, overwrite the existing row using an UPDATE and use only the key columns in the WHERE clause. • If the timestamp of the replicated operation is **less than or equal to** the timestamp of the row in the target, discard the change record and do nothing to the target row.
DELETE	Ignore the conflict (out-of-sync message).

Syntax in conflict resolution file

```
owner.table   {I | U | D}   !MostRecentRecord(col_name)
```

Where *col_name* is the timestamp column to be used by the routine.

NOTES:

- The recommended data type for col_name in a PostgresSQL database is timestamp, as DATE data type does not store time value. As a result, if a date data type is used, a conflict will not be resolved if the same date is updated at both peers at the same time
- The case sensitivity of the `col_name` for the Oracle peer depends on the source database.
  - If the data is being replicated from Oracle to Oracle, the `col_name` should be in uppercase.
  - If the data is being replicated from PostgreSQL to Oracle, the `col_name` should be in lowercase.
  - If the data is being replicated from both Oracle and PostgreSQL sources to Oracle, there should be two different entries of conflict resolution routines for `col_name` – one in uppercase and the other in lowercase.

!UpdateUsingKeyOnly

This routine works for UPDATE operations. It provides conflict resolution that relies solely on the key value of the changed row. Normally, when SharePlex builds a SQL statement to post data, the WHERE clause uses both the key and the pre-image of the columns that changed to ensure synchronization. The **!UpdateUsingKeyOnly** routine directs SharePlex to post the data even though the pre-image values do not match, assuming the keys match.

If appropriate, this routine can be used as the sole routine for UPDATEs, but with the understanding that it does not include logic that assigns priority, such as system or time priority, in case of multiple concurrent UPDATEs. To avoid out-of-sync errors, Quest recommends using **!UpdateUsingKeyOnly** in conjunction with other, more specific routines, relying on **!UpdateUsingKeyOnly** as a final option if the custom routines fail.

IMPORTANT: **!UpdateUsingKeyOnly** must be the last entry in the list of routines, thus assigning it last priority.

In the following example, when there is a conflict for **owner.table1** during an UPDATE, SharePlex calls the two custom routines first (in order of priority) and then calls the **!UpdateUsingKeyOnly** routine.

owner.table1	u	owner.procedure_up_A
owner.table1	u	owner.procedure_up_B
owner.table1	u	!UpdateUsingKeyOnly

The !UpdateUsingKeyOnly name is case sensitive. It must be typed exactly as shown in these instructions, with no spaces between words. Do not list an owner name with this routine in the configuration file. For INSERT and DELETE operations, custom logic must be used.

Log information about resolved conflicts for Oracle database

You can configure the Post process to log information about successful conflict resolution operations if you are using the SharePlex prepared routines. This feature is disabled by default.

To enable the logging of conflict resolution:

1. Run **sp_ctrl** on the target system.
2. Issue the following command:

   `sp_ctrl>` set param SP_OPO_LOG_CONFLICT {1 | 2}

   - A setting of 1 enables the logging of conflict resolution to the **SHAREPLEX_CONF_LOG** table.

     NOTE: A setting of 1 will not update the columns EXISTING_TIMESTAMP and TARGET_ROWID (when existing data is not replaced) in the SHAREPLEX_CONF_LOG table.

   - A setting of 2 enables the logging of conflict resolution to the **SHAREPLEX_CONF_LOG** table with Post query for additional meta data.

     Using LeastRecentRecord or MostRecentRecord prepared routines Post will query the target database for the timestamp column of the existing record. The query result is logged into the EXISTING_TIMESTAMP column of the SHAREPLEX_CONF_LOG table.

     For any prepared routines, on rows that aren't replaced by the incoming record, Post will query the TARGET_ROWID of the existing row that could have been replaced. Otherwise the ROWID of the existing row will not be logged.

     NOTE: A setting of 2 may affect the performance of Post as a result of making the query.

3. Restart Post.

Post logs the information to a table named **SHAREPLEX_CONF_LOG**. The following describes this table.

Column	Column Definition	Information that is logged
CONFLICT_NO	NUMBER NOT NULL	The unique identifier of the resolved conflict. This value is generated from the **shareplex_conf_log_seq** sequence.
CONFLICT_ TIME	TIMESTAMP DEFAULT SYSTIMESTAMP	The timestamp of the conflict resolution
CONFLICT_ TABLE	VARCHAR2(100)	The name of the target table that was involved in the conflict
CONFLICT_ TYPE	VARCHAR2(1)	The type of conflict, either I for insert, U for update, or D for delete
CONFLICT_ RESOLVED	VARCHAR2(1) NOT NULL	Indicator of whether the conflict was resolved or not.

Column	Column Definition	Information that is logged
		Y = yes, the conflict was resolved N = no, the conflict was not resolved. Unresolved conflicts are logged to the *ID*_**errlog.sql** file, where *ID* is the source database identifier.
TIMESTAMP_ COLUMN	VARCHAR2(50)	The name of the column that contains the timestamp that Post compared to determine which record was most recent.
INCOMING_ TIMESTAMP	DATE	The timestamp that the row was inserted, updated, or deleted on the source system
EXISTING_ TIMESTAMP	DATE	The current timestamp of the row in the target database. This applies only if the SP_OPO_LOG_CONFLICT parameter is set to 2, which directs Post to query the target database to get this value.
PRIMARY_ KEYS	VARCHAR2(4000)	The names of the primary key columns

MESSAGE	VARCHAR2(400)	A message that states which row won in the conflict. The row that wins depends on which conflict resolution routine was used. For example, the following message is returned when the **!MostRecentRecord** routine is used and the most recent record is the source record:
		
		`Incoming timestamp > existing timestamp. Incoming wins, overwrite existing.`
		
		If the target record was the most recent one or has the same timestamp as the source record, then the message would be:
		
		`Incoming timestamp <= existing timestamp. Existing wins, discard incoming.`

SQL_ STATEMENT	LONG	The final SQL statement that got executed as a result of the conflict resolution
CONFLICT_ CHECKED	VARCHAR2(1)	Indicates whether or not someone reviewed the conflict. The default is **N** for No. The person who reviews the conflict can change this value to **Y**.

Log information about resolved conflicts for PostgreSQL database

You can configure the Post process to log information about successful conflict resolution operations if you are using the SharePlex prepared routines. This feature is disabled by default.

To enable the logging of conflict resolution:

1. Run **sp_ctrl** on the target system.
2. Issue the following command:

   `sp_ctrl>` set param SP_OPX_LOG_CONFLICT {1 | 2}

   - A setting of 1 enables the logging of conflict resolution to the **shareplex_conf_log** table.

     NOTE: A setting of 1 will not update the column `existing_timestamp` (when existing data is not replaced) in the `shareplex_conf_log` table.

   - A setting of 2 enables the logging of conflict resolution to the **shareplex_conf_log** table with Post query for additional meta data.

     Using **LeastRecentRecord** or **MostRecentRecord** prepared routines Post will query the target database for the timestamp column of the existing record. The query result is logged into the **existing_timestamp** column of the **shareplex_conf_log** table.

     NOTE: A setting of 2 may affect the performance of Post as a result of making the query.

3. Restart Post.

Post logs the information to a table named **shareplex_conf_log**. The following describes this table.

Column	Column Definition	Information that is logged
conflict_no	bigserial primary key	The unique identifier of the resolved conflict. This value is generated from the `shareplex_conf_log_conflict_no_seq` sequence.
conflict_ time	timestamp	The timestamp of the conflict resolution.
src_host	varchar(64)	The host name of source host.
curr_host	varchar(64)	The host name of current host.
trusted_ host	varchar(64)	The host name of trusted host. It will be used in case of **!HostPriority** prepared routine.
src_db	varchar (150)	The source database name.
source_ rowid	varchar(20)	The source table rowid. For PostgreSQL source, this column is not applicable. Its value will be **N/A**.

Column	Column Definition	Information that is logged
target_ rowid	varchar(20)	The target table rowid. For PostgreSQL target, this column is not applicable. Its value will be **N/A**.
conflict_ table	varchar (300)	The name of the target table that was involved in the conflict.
conflict_ type	char(1)	The type of conflict, either I for insert, U for update, or D for delete.
conflict_ resolved	char(1) not null	Indicator of whether the conflict was resolved or not. Y = yes, the conflict was resolved N = no, the conflict was not resolved. Unresolved conflicts are logged to the ID_ **errlog.sql** file, where ID is the source database identifier.
odbc_ error	varchar(20)	Indicates odbc error which caused the conflict. Its format is `<native error>:<error sqlstate>`

Configure Replication through an Intermediary System

These instructions show you how to set up *cascading replication*, also known as *multi-tiered* replication. This strategy replicates data from a source system to an intermediary system, and then from the intermediary system to one or more remote target systems.

Cascading replication can be used to support various replication objectives as a workaround in such conditions as the following:

- Your replication strategy exceeds the 1024 routes that are allowed directly from a given source system: You can send data to the intermediary system and then broadcast to the additional targets from there.
- The source has no direct connection to the ultimate target, because of firewall restrictions or other factors. You can cascade to a system that does allow remote connection from the source system.

To use a cascading strategy, the source machine must be able to resolve the final target machine name(s), although the ability to make a direct connection is not required.

Supported sources

Oracle and PostgreSQL

Supported targets

Oracle

Oracle and Open Target (final target)

Capabilities

This replication strategy supports the following:

- Replication to one or more target systems
- Identical or different source and target names
- Use of vertically partitioned replication
- Use of horizontally partitioned replication
- Use of named export and post queues

Requirements

- Prepare the system, install SharePlex, and configure database accounts according to the instructions in the SharePlex Installation Guide.

  IMPORTANT! Create the same SharePlex user on all systems if you will be using SharePlex to post to a database on the intermediary system.

- Disable triggers that perform DML on the target objects.
- No DML or DDL should be performed on the target tables except by SharePlex. Tables on the target system that are outside the replication configuration can have DML and DDL operations without affecting replication.
- If sequences are unnecessary on the target system, do not replicate them. It can slow down replication. Even if a sequence is used to generate keys in a source table, the sequence values are part of the key columns when the replicated rows are inserted on the target system. The sequence itself does not have to be replicated.

DDL Replication Support

DDL replication from source to target through an intermediary system is supported in accordance with the information found in the DDL that SharePlex Supports chapter of the Administration Guide, with the following exceptions:

- DDL initiated on the intermediary system, as opposed to the source, will cause inconsistencies leading to Post errors and should be avoided unless the DDL is synchronized across all systems.
- All systems must be monitored to ensure that latency or errors on the intermediary system do not cause inconsistencies.

IMPORTANT! These instructions assume you have a full understanding of SharePlex configuration files. They use abbreviated representations of important syntax elements. For more information, see Configure SharePlex to Replicate Data on page 60.

Conventions used in the syntax

In the configuration syntax in this topic, the placeholders represent the following:

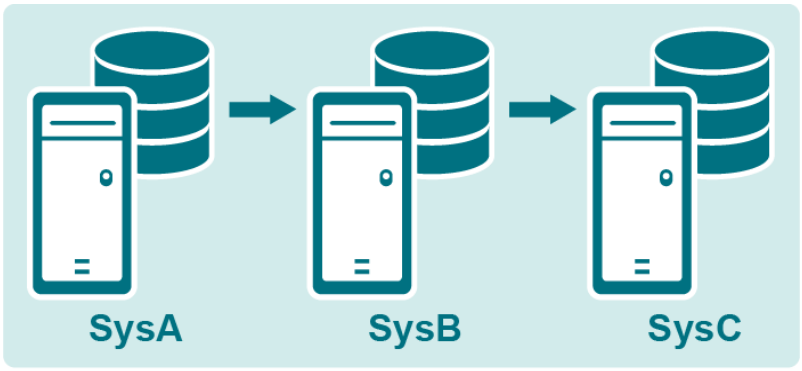
- *source_specification[n]* is the fully qualified name of a source object (owner.object) or a wildcarded specification.
- *target_specification[n]* is the fully qualified name of a target object or a wildcarded specification.
- *host* is the name of a system where SharePlex runs. Different systems are identified by appending a letter to the names, like *hostB*.
- *db* is a database specification. The database specification consists of either **o.** or **r.** prepended to the Oracle SID, TNS alias, or database name, as appropriate for the connection type. A database identifier is not required if the target is JMS, Kafka, or a file.

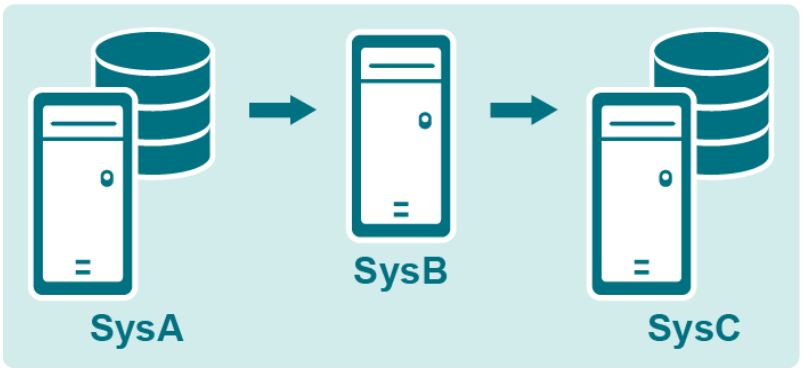
IMPORTANT! Configure SharePlex to Replicate Data on page 60.

Deployment options

To cascade data, you have the following options:

- If there is a database on the intermediary system, you can configure SharePlex to post to that database and then capture the data again to replicate it to one or more remote targets.
- If there is not a database on the intermediary system, you can configure SharePlex to import, queue, and then export the data to one or more remote targets. There is no Capture process on the system. This is known as a *pass-through* configuration. It passes the data directly from the source system to the target(s).





Cascade with posting on intermediate system

To use this configuration:

- SharePlex database accounts must exist on all systems and must be the same name on all systems. This account is usually created when SharePlex is installed. See the SharePlex Installation Guide for more information.
- Triggers must be disabled in the intermediary database, as well as on the target system.
- Oracle DDL replication is not supported from an Oracle database on the intermediary system to the target systems. It is supported only from the source system to the intermediary system.

- You create two configuration files: one on the source system, and one on the intermediary system.
- Enable archive logging on the source and intermediary systems in case the redo logs wrap before Capture is finished with them.

Configuration options on source system

This configuration replicates from the source system to the database on the intermediary system.

NOTE: In this template, *hostB* is the intermediary system.

datasource_specification		
source_specification1	*target_specification1*	*hostB*@o.*SID*
source_specification2	*target_specification2*	*hostB*@o.*SID*

Example on source system

Datasource:o.oraA		
hr.emp	**hr.emp2**	**hostB@o.oraB**
hr.sal	**hr.sal2**	**hostB@o.oraB**
cust.%	**cust.%**	**hostB@o.oraB**

NOTE: In this same configuration, you could route data from other source objects directly to other targets, without cascading through the intermediary system. Just specify the appropriate routing on a separate line.

Configuration options on intermediary system

This configuration captures the data from the database on the intermediary system, then replicates it to the target system(s). The tables that were the target tables in the source configuration are the source tables in this configuration. The target can be any supported SharePlex target.

datasource_specification		
source_specification1	*target_specification1*	*hostC*[@*db*][+...]
source_specification2	*target_specification2*	*hostD*[@*db*][+...]

Example on intermediary system

Datasource:o.oraB		
hr.emp	**hr.emp2**	**hostC@o.oraC**
hr.sal	**hr.sal2**	**hostD@o.oraD+hostE@r.mssE**
cust.%	**cust.%**	**!cust_partitions**

NOTE: The last entry in this example shows the use of horizontally partitioned replication to distribute different data from the **sales.accounts** table to different regional databases. For more information, see Configure Horizontally Partitioned Replication on page 118.

Required parameter setting on intermediary system

(Oracle intermediary database) Set the SP_OCT_REPLICATE_POSTER parameter to 1 if the intermediary database is Oracle. This instructs the Capture process on the intermediary system to capture the changes posted by SharePlex and replicate them to the target system. (The default is 0, meaning that Capture ignores Post activity on the same system.)

In **sp_ctrl**, issue the following command. The change takes effect the next time Capture starts.

```
set param SP_OCT_REPLICATE_POSTER 1
```

Cascade with pass-through on intermediary system

To use this configuration:

- Create an Oracle instance and an ORACLE_SID specification in the **oratab** file (Unix and Linux systems). The database can be empty.
- DDL replication is not supported.
- You create one configuration file, which is on the source system.

Configuration options on source system

NOTE: In this template, *hostB* is the intermediary system.

datasource_specification		
source_specification1	*target_specification1*	*hostB*hostC*[@*db*]
source_specification2	*target_specification2*	*hostB*hostD*[@*db*][+*hostB*hostE*[@*db*][+...]
source_specification3	*target_specification3*	*hostB*hostX*[@*db*]**+***hostY*[@*db*]

- The *hostB*host* syntax configures the pass-through behavior.
- If using a compound routing map where all data passes through the intermediary system first, make certain to use the *hostB** component in each target route.
- You can also use a compound routing map where data from a source object is replicated directly to one target, and also through the intermediary system to another target, as in the third line of this configuration file.

Example

Datasource:o.oraA		
hr.emp	**hr.emp2**	**hostB*hostC@o.oraC**
hr.emp	**hr."Emp_3"**	**hostB*hostC@r.mssB**
cust.%	**cust.%**	**hostB*hostD@o.oraD+hostE@o.oraE**

Configure Replication to Maintain High Availability

These instructions show you how to set up SharePlex for the purpose of *high availability*: replicating to a secondary Oracle database that is a mirror of the source database. This strategy uses bi-directional replication with two SharePlex configurations that are the reverse of each other. The configuration on the secondary (standby) machine remains in an activated state with the Export process on that system stopped in readiness for failover if the primary machine fails.

NOTE: For CrunchyData High Availability cluster environment setup information, see the **Install SharePlex on PostgreSQL High Availability Cluster** section in the SharePlex Installation and Setup Guide.

This strategy supports business requirements such as the following:

- Disaster recovery
- Continuous operation of business applications throughout maintenance cycles or mechanical failures

In this strategy, SharePlex operates as follows:

- Under normal conditions, SharePlex replicates changes from the primary database to the secondary database.
- When the primary system or database is offline and users are transferred to the secondary system, SharePlex captures their changes and queues the data on that system until the primary system is restored.
- When the primary system is restored, SharePlex updates it with those changes and then resumes capture and replication from the primary database.

Supported sources

Oracle and PostgreSQL

Supported targets

Oracle

Capabilities

This replication strategy supports the use of named export and post queues.

NOTE: Column mapping and partitioned replication is not appropriate in a high availability configuration. Source and target objects can have different names but this makes the management of a high-availability structure more complicated.

Requirements

- Prepare the system, install SharePlex, and configure database accounts according to the instructions in the SharePlex Installation Guide.
- All objects must exist in their entirety on both systems.
- The target objects must have the same structure and qualified names as their source objects.
- Enable archive logging on all systems.
- Create a script that denies INSERT, UPDATE and DELETE operations to all users except SharePlex.

For failover purposes, the following are required:

- Make the applications used on the primary system available on the secondary system.
- Copy non-replicated database objects and critical files outside the instance to the secondary system.
- Create a script that grants INSERT, UPDATE and DELETE privileges to all users, which can be run during a failover procedure.
- Create a script that enables constraints on the secondary system to be used during a failover procedure.
- Develop a failover procedure for relocating users to the secondary system.

NOTE: If you use an Oracle hot backup to create the secondary instance, keep the script. It can be modified to re-create the primary instance.

Conventions used in the syntax

In the configuration syntax in this topic, the placeholders represent the following:

- *hostA* is the primary system.
- *hostB* is the secondary system.
- *ownerA.object* is the fully qualified name of an object on *hostA* or a wildcarded specification.
- *ownerB.object* is the fully qualified name of an object on *hostB* or a wildcarded specification.
- oraA is the Oracle instance on *hostA*.
- *oraB* is the Oracle instance on *hostB*.

IMPORTANT! See Configure SharePlex to Replicate Data for more information about the components of a configuration file.

Configuration

A high availability configuration uses two configurations that are the reverse of each other. To replicate all objects in the database, you can use the **config.sql** script to simplify the configuration process. For more information, see the Configuration Scripts section in the SharePlex Reference Guide.

Configuration on the source system (primary system)

Datasource:o.*oraA*		
ownerA.object	*ownerB.object*	*hostB***@o.***oraB*

Configuration on the target system (secondary system)

Datasource:o.*oraB*		
ownerB.object	*ownerA.object*	*hostA***@o.***oraA*

Make the system ready for failover

1. On the secondary system (the one that will initially be the passive system) run **sp_ctrl** and then issue the following command to stop the Export process on the secondary system so that nothing accidentally happening on the secondary system (such as a scheduled job changing data) gets replicated back to the primary system. This is the required state of SharePlex on that system until there is a need for a role switch between systems.
2. Perform initial synchronization and startup. You will activate the source configuration during this procedure. For more information, see Start Replication on your Production Systems on page 254.
3. Making sure the Export process is stopped on the secondary system, activate the configuration on that system. The configuration on the secondary machine remains in an activated state, but the stopped Export process and lack of user activity ensure that the system remains static in readiness for failover.
4. Monitor the SharePlex instance that is linked to the secondary Oracle instance to make sure no non-SharePlex DDL or DML changes were performed. You can do this as follows: View the status of the export queue on the secondary system using the **qstatus** command in **sp_ctrl**. The queue should be empty, because the Capture process on a system ignores the Post process on that system. If there are any messages in the export queue, it means those transactions originated on the secondary system or the SP_OCT_REPLICATE_POSTER parameter was mistakenly enabled. See the SharePlex Reference Guide for more information about SharePlex commands and parameters.
5. Maintain backups of replication files.

Perform recovery procedures

If a system fails in your high-availability environment, you can move replication to a secondary system and then move it back to the primary system when it is restored. For more information, see Recover Replication after Oracle Failover on page 368.

12

Configure DDL Replication

This chapter contains the information that you need to know in order to replicate Oracle DDL operations that are supported by SharePlex.

Contents

DDL that SharePlex Supports

SharePlex supports DDL replication for Oracle databases only. SharePlex replicates certain Oracle DDL changes that are written to the redo logs. Changes that bypass the redo logs are not replicated.

For details on the DDL that SharePlex supports, see the SharePlex Release Notes.

In a cascading configuration, DDL replication is supported from source to target through an intermediary system. However, DDL initiated on the intermediary system could cause inconsistencies leading to Post errors and must be avoided. For more information, see Configure Replication through an Intermediary System on page 207.

DDL against objects owned by the SharePlex user or against system-owned objects is not replicated.

NOTE: Because the source and target databases can be of different versions in SharePlex replication, the source and target objects can be different. When DDL is applied to the target, it may fail if the operation is forbidden on the target but allowed on the source.

For a list of objects for which DDL is supported, see the SharePlex Release Notes.

Control Oracle DDL Replication

By default, SharePlex replicates some Oracle DDL for objects that are listed in the active configuration (explicitly or by wildcard), but you can expand this support with parameter settings.

IMPORTANT!

- For the most current information about supported DDL and requirements or limitations, see the SharePlex Release Notes provided with this release. That information may supercede what is documented here.
- DDL replication is supported only to Oracle targets, except for ALTER TABLE to ADD COLUMN or DROP COLUMN, which are supported for all SharePlex targets.

SharePlex provides default DDL support for objects in the configuration file. You can expand this support through parameter settings.

See the SharePlex Release Notes for detailed information about the DDL that is supported by SharePlex.

Default support for Oracle DDL

SharePlex provides some basic DDL support by default.

DDL for existing objects

By default SharePlex replicates:

- **TRUNCATE TABLE**
- **ALTER TABLE to:**

  ADD, MODIFY, DROP, SPLIT, COALESCE, MOVE, TRUNCATE, EXCHANGE PARTITION/SUBPARTITION

  ADD, MODIFY, or DROP columns

  when:

  - the affected object exists in the source and target at the time of activation and
  - its name is listed in the configuration file (explicitly or through wildcard).
- **DROP TABLE**

This functionality is controlled by the `SP_OCT_REPLICATE_DDL` parameter.

The valid values are as follows:

0 (disable replication of both ALTER TABLE and TRUNCATE)

1 (enable ALTER replication only)

2 (enable TRUNCATE replication only)

3 (enable replication of ALTER and TRUNCATE)

DDL for objects added after activation

By default, the SharePlex Auto-Add feature is also enabled to provide DDL support for **tables** and **indexes** that are created after activation. When SharePlex detects a CREATE statement for one of these objects and its name satisfies a wildcard in the configuration file, SharePlex does the following:

- replicates the CREATE to add the object to the target
- adds the object to replication
- maintains that object through future DDL and DML changes

The Auto-Add feature is controlled by the `SP_OCT_AUTOADD_ENABLE` parameter, which is set to 1 (enabled) by default.

See the SharePlex Reference Guide for details about this parameter.

Optional DDL on objects in replication

You can enable the replication of the following DDL when it is issued on objects that are in the configuration file. To enable the replication of a DDL command, set the associated parameter to 1.

DDL command	Set this parameter to 1
CREATE / DROP TRIGGER	SP_OCT_REPLICATE_TRIGGER
CREATE / DROP SYNONYM	SP_OCT_REPLICATE_SYNONYM
GRANT	SP_OCT_REPLICATE_GRANT

See the SharePlex Reference Guide for details about these parameters.

Optional Auto-Add support for Oracle DDL

You can expand Auto-Add support to include any of the object types listed in the following table. The object is added to replication if its name satisfies a wildcard specification in the active configuration file.

To enable auto-add of individual object types:

1. Make certain the `SP_OCT_AUTOADD_ENABLE` parameter is set to 1.
2. Set the appropriate parameter to 1, using the following table as your guide.

DDL to auto-add	Parameter to set to 1	Additional requirements
CREATE / DROP SEQUENCE	SP_OCT_AUTOADD_SEQ	Set the SP_SYS_TARGET_COMPATIBILITY parameter to at least 8.6.3.
CREATE /DROP MATERIALIZED VIEW*	SP_OCT_AUTOADD_MVIEW	Set the SP_SYS_TARGET_COMPATIBILITY parameter to at least 8.6.2.

* SharePlex does not replicate materialized views to materialized views. SharePlex converts a CREATE MATERIALIZED VIEW to a CREATE TABLE, applies the CREATE TABLE to the target, and then replicates the DML that populates the view. SharePlex replicates DROP MATERIALIZED VIEW, but not ALTER MATERIALIZED VIEW. See the SharePlex Reference Guide for details about these parameters.

Expanded DDL support for objects outside replication

You can configure SharePlex to replicate DDL for certain objects that are *not listed* in the configuration file. SharePlex replicates the DDL statements, but does not replicate any data change operations made to the objects because they are not part of active replication. Therefore, SharePlex does not maintain synchronization of these objects on the target. The objects must exist in the source and target prior to configuration activation.

NOTE: Expanded DDL replication supports not only tables and sequences but also a wide range of other objects such as procedures, functions, users, and views, which are not part of replication. Some of these objects may have underlying objects that *are in replication*. In those cases, Expanded DDL replication applies not only to the object that is outside the replication configuration, but also to the underlying objects that are in replication.

SharePlex does not support the Oracle Flashback Table feature. If the SP_REPLICATE_ALL_DDL parameter is enabled (value of 1), SharePlex may try to replicate the flashback DDL, which will return an error. To perform Flashback Table on a table that is in replication, use the following procedures to work around this issue:

1. Remove source objects from replication
2. Perform the flashback
3. Add or change objects in an active configuration

To replicate DDL for objects outside the replication configuration:

1. Set the SP_OCT_REPLICATE_ALL_DDL parameter to 1.
2. See the SharePlex Reference Guide for details about this parameter.
3. See the SharePlex Release Notes for more information about supported DDL for objects that are not in the configuration file.

Filter DDL Replication

You can filter the objects for which DDL is replicated when the SP_OCT_REPLICATE_ALL_DDL parameter is enabled. This is the DDL that is not in the replication configuration.

NOTE: DDL filtering is only allowed for objects outside the replication configuration. All DDL performed on objects inside the replication configuration must be replicated to keep the source and target metadata consistent so that DML succeeds.

DDL filtering is controlled in the SHAREPLEX_DDL_CONTROL table that is installed in the SharePlex schema.

```
Name              Type

-------------     ------------

DDL_PARAMETER     NUMBER

DDL_CODE          NUMBER

SCHEMA_FILTER     VARCHAR2(32)

OBJECT_FILTER     VARCHAR2(32)
```

Each row in the SHAREPLEX_DDL_CONTROL table defines a filter based on what you specify in each of the following columns:

- SCHEMA_FILTER filters DDL by a schema name.
- OBJECT_FILTER filters DDL by an object name.
- DDL_CODE filters by the code number of a DDL type. See **DDL codes**.

A null value in the DDL_CODE column means that the filter applies to *all* of the DDL types A null in the SCHEMA_FILTER or OBJECT_FILTER column means that the filter applies to *any* schema or object name.

NOTE: The DDL_PARAMETER column is not an active column as of this release of SharePlex.

To filter DDL:

Insert a row into the table with the desired values in the active columns.

Examples

The following filters out of replication the DDL for ALTER TABLE:

```
INSERT INTO SPLEX.SHAREPLEX_DDL_CONTROL (DDL_CODE, SCHEMA_FILTER, OBJECT_
FILTER) values (15,null,null);
```

The following filters out of replication all DDL for all objects with names that begin with TEST_ in any schema:

```
INSERT INTO SPLEX.SHAREPLEX_DDL_CONTROL (DDL_CODE, SCHEMA_FILTER, OBJECT_
FILTER) values (null,null,'TEST_%');
```

The following filters out of replication the DDL for CREATE TABLE for the "Sales" schema and objects with names that begin with "TEST_":

```
INSERT INTO SPLEX.SHAREPLEX_DDL_CONTROL (DDL_CODE, SCHEMA_FILTER, OBJECT_
FILTER) values (1,'Sales','TEST_%');
```

DDL codes

DDL Type	DDL_CODE
CREATE TABLE	1
ALTER TABLE	15
DROP TABLE	12
ASSOCIATE STATISTICS	168
DISASSOCIATE STATISTICS	169
COMMENT TABLE, COMMENT ON COLUMNS	29
TRUNCATE	85
CREATE INDEX	9
ALTER INDEX	11
DROP INDEX	10
CREATE SEQUENCE	13
ALTER SEQUENCE	14
DROP SEQUENCE	16
CREATE CLUSTER	4
DROP CLUSTER	8
CREATE USER	51
ALTER USER	43
DROP USER	53
CREATE_ROLE	52

DDL Type	DDL_CODE
ALTER_ROLE	79
DROP_ROLE	54
GRANT	17
REVOKE	18
CREATE SYNONYM	19
DROP SYNONYM	20
CREATE VIEW	21
ALTER VIEW	88
DROP VIEW	22
CREATE TYPE	77
ALTER TYPE	80
DROP TYPE	78
CREATE TYPE BODY	81
DROP TYPE BODY	83
CREATE FUNCTION	91
ALTER FUNCTION	92
DROP FUNCTION	93
CREATE PROCEDURE	24
ALTER PROCEDURE	25
DROP PROCEDURE	68
CREATE PACKAGE	94
ALTER PACKAGE	95
DROP PACKAGE	96
CREATE PACKAGE BODY	97
ALTER PACKAGE BODY	98

DDL Type	DDL_CODE
DROP PACKAGE BODY	99
CREATE DIRECTORY	157
DROP DIRECTORY	158

Best Practices for Alter Table DDL

The following are best practices for the replication of Oracle ALTER TABLE operations.

Tables with VARRAY or ABSTRACT types

Do not add a VARRAY column or abstract data type column if you plan to issue an ALTER TABLE to drop a column or set a column unused soon thereafter. SharePlex must query the database to obtain information about this data type. If the second DDL was performed before SharePlex was able to process the first DDL, the query will fail because the metadata is already changed.

Tables with system-specific metadata

If some metadata is system-specific, such as the storage parameters of database objects, there may be unexpected results when DDL on that metadata is replicated. For example, SharePlex replicates all of the storage parameters for a source Oracle object, even though only some of them were changed with the ALTER TABLE command. If the source and target objects were not created with the same storage parameters, one of two things can happen: either the target table will assume the storage of the source table or, if the DDL is not supported by the target, an error will be generated.

Example: consider a source table with MAXEXTENTS 525 and MINEXTENTS 20, and a target table with MAXEXTENTS 505 and MINEXTENTS 4. If the MAXEXTENTS of the source object is changed to unlimited, SharePlex will replicate both the MAXEXTENTS change and the non-changed MINEXTENTS of 20. This causes Oracle error 01570, because MINEXTENTS cannot be larger than the extents currently allocated. Alternatively, if the MINEXTENTS is changed to 1 on the source, but MAXEXTENTS is not changed, SharePlex replicates both, which results in target parameters of MAXEXTENTS 525 and MINEXTENTS 1.

Tables that are renamed

When ALTER TABLE RENAME is issued on a source table that is currently in replication, SharePlex changes the name of the table in the active configuration file by commenting out the old configuration line and adding a new line at the end of the configuration file. If the source and target table names are the same, both are changed to the new name. Otherwise, just the source name is changed. The following is an example:When ALTER TABLE RENAME is issued on a source table that is currently in replication, SharePlex changes the name of the table in the active configuration file by commenting out the old configuration line and adding a new line at the end of the configuration file. If the source and target table names are the same, both are changed to the new name. Otherwise, just the source name is changed.

The following is an example:

```
# Table scott.table1 renamed to scott.table2 August 5, 2003 10:14

scott.table2 scott.table2 sysA@o.ora555
```

Whether the Post process stops on RENAME operations or not depends on the setting of the `SP_OPO_STOP_ON_DDL_ERR` parameter.

Tables with system-generated interval partitions/subpartitions

Because the database generates the names of system-generated interval partitions/subpartitions, the names of those partitions on the source will not match the names of their corresponding partitions on the target. Set the SP_OCT_TRUNC_PARTITION_BY_ID parameter to 1 to ensure that SharePlex truncates the correct partition when it replicates an ALTER TABLE to TRUNCATE a system-generated interval partition. This setting directs SharePlex to identify the partition by using the partition ID, rather than by using the partition name that is specified in the original ALTER TABLE command. Post maps the partition ID to the correct partition name on the target table. For more information, see the SharePlex Reference Guide.

To support the replication of system-named interval partitions/subpartitions, both the source and target must be SharePlex version 8.6.4 or later.

SharePlex does not support TRUNCATE of a system-generated sub-partition if the sub-partition is empty.

ALTER TABLE...MOVE

ALTER TABLE DDL commands that change the rowid of a table can affect subsequent DML operations if the primary or unique keys of the tables in replication are not being logged. When the keys are not logged, SharePlex fetches their values based on the rowid. Any operation that changes the rowid, such as ALTER TABLE...MOVE, can cause the wrong key values to be used for subsequent DML operations.

DDL Logging and Error Handling

Both Capture and Post log the DDL that they process. SharePlex also prints replicated DDL to the SharePlex Event Log, but it may be truncated. Only the Post DDL log contains complete DDL statements. SharePlex stores the DDL logs in the **log** subdirectory of the variable-data directory on the source and target systems.

By default, Post stops on DDL errors. An error usually indicates that the database component for which the DDL was executed on the source system does not exist in the target database. The default setting of the SP_OPO_STOP_ON_DDL_ERR parameter stops Post on DDL errors, so that subsequent DML on that object does not fail. This enables you to correct the problem and keep the databases synchronized. For more information about this parameter, see the SharePlex Reference Guide.

Table 5: SharePlex DDL log naming conventions

DDL log type	Naming convention	Example
Capture	**o.***ORACLE_SID*_**ocap_ddl_***log_number*.**log**	**o.ora12_ocap_ddl_01.log**
Oracle target	**o.***ORACLE_SID_machine_name*_**opo_ddl_***log_number*.**log**	**o.ora12_server2_opo_ddl_01.log**
Open Target target	**r.***database_name_machine_name*_**xpst_ddl_***log_number*.**log**	**r.mssdb1_server3_xpst_ddl_01.log**

13

Configure Error Handling

This chapter contains an overview of the tools that SharePlex provides to handle errors that are returned by the Post process.

Contents

Continue to Post When there is a DML Error

SharePlex provides a way for Post to continue processing after it encounters a DML error, rather than stop.

Continue posting on Oracle and SharePlex errors

Valid for Oracle targets

When SharePlex posts to an Oracle target, you can configure Post to ignore specific Oracle DML errors and specific SharePlex error messages and continue processing. Post determines which messages to ignore based on the list in the **oramsglist** file. The file is installed with a small list of errors by default, but you can remove any of them as desired.

When Posts ignores an error, it writes the error to the SharePlex Event Log. Post also logs the error and the SQL statements that caused the error to the Error log. This log is named *SID*_**errlog.sql** log file and is stored in the **data** directory in the SharePlex variable-data directory. For more information, see View Events and Errors on page 286.

NOTE: There are certain errors that Post will not ignore, and it will stop for those messages even if they are listed in the **oramsglist** file.

IMPORTANT: Use caution when using this feature. It could result in hidden out-of-sync conditions. Enable this parameter only if your target users cannot tolerate replication lag and it is acceptable to have some out-of-sync data. Check the *SID*_**errlog.sql** log frequently to see if there were errors that could cause replication problems.

To configure Post to continue on errors:

1. On the **target system**, change directories to the **data** sub-directory of the SharePlex variable-data directory.
2. Find the **oramsglist** file.
3. If replication is not active, open the file in a text editor. If replication is active, make a copy of the file and then open the copy in the editor.
4. Increase the number on the *first* line by the number of errors that you are adding. This number must be equal to the total number of errors that are in the file. For example, in the following file there are 10 errors listed.

```
ora@sys1dad > vi oramsglist
10
604
900
902
908
909
910
911
932
960
1026
```

5. Starting at the *end* of the file, add the **number** of each Oracle or SharePlex error, one per line as shown in the preceding example. The messages need not be in numerical order.
6. Save and close the file.
7. Stop Post (if running).

```
sp_ctrl> stop post
```

8. If you edited a copy of the **oramsglist** file, save the copy to the original name of **oramsglist**.
9. Change the value of the SP_OPO_CONT_ON_ERR parameter to 1. Or change the value to 2 to also continue posting on table errors listed in the **oramsglist** file. See the SharePlex Reference Guide for a description of the SP_OPO_CONT_ON_ERR parameter.

```
sp_ctrl> set param SP_OPO_CONT_ON_ERR 1
```

10. Start Post.

```
sp_ctrl> start post
```

Continue posting on ODBC errors

Valid for Open Target

When SharePlex posts to an Open Target target, you can configure Post to ignore ODBC errors and continue processing. Post writes the error to the SharePlex Event Log. Post also logs the error and the SQL statements that caused the error to the Error log. This log is named *ID*_**errlog.sql** log file, where *ID* is the database identifier. The file is stored in the **data** directory in the SharePlex variable-data directory. For more information, see View Events and Errors on page 286.

IMPORTANT: Use caution when using this feature. It could result in hidden out-of-sync conditions. Enable this parameter only if your target users cannot tolerate replication lag and it is acceptable to have some out-of-sync data. Check the *SID*_**errlog.sql** log frequently to see if there were errors that could cause replication problems.

To configure Post to continue on errors:

1. On the **target system**, change directories to the **data** sub-directory of the SharePlex variable-data directory.
2. Look for one of the following files, depending on the database. These files are installed empty.

File name	Supported database
postgresmsglist	Postgres
sqlservermsglist	Microsoft SQL Server
mysqlmsglist	Oracle MySQL

NOTE: There are certain errors for which Post will stop, even if you list those errors in the message file.

3. If replication is not active, open the file in a text editor. If replication is active, make a copy of the file and then open the copy in the editor.
4. Starting at the *end* of the file, add the **number** of each error, one per line as shown in the example. The messages need not be in numerical order.

   Example:

   ```
   sqlservermsglist:

   8102

   8180

   544

   2627

   3621
   ```

5. Save and close the file.
6. Stop Post (if running).

   ```
   sp_ctrl> stop post
   ```

7. If you copied the original file, save it back to its original name.
8. Change the value of the SP_OPX_CONT_ON_ERR parameter to 1.

   ```
   sp_ctrl> set param SP_OPX_CONT_ON_ERR 1
   ```

9. Start Post.

   ```
   sp_ctrl> start post
   ```

Continue to Post When there is a DDL Error

By default, Post stops on DDL errors. An error usually indicates that the database component for which the DDL was executed on the source system does not exist in the target database. The default setting of the SP_OPO_STOP_ON_DDL_ERR parameter stops Post on DDL errors, so that subsequent DML on that object does not fail. This enables you to correct the problem and keep the databases synchronized. For more information about this parameter, see the SharePlex Reference Guide.

Increase the Number of Retries on Error

Post will retry certain failed operations when there is the possibility that they will succeed with another attempt. The main operations that Post will retry are TNS write failures, connection failures, or locks on tables when Post needs to apply a TRUNCATE.

To increase the likelihood that the failed operations are successful, you can increase the SP_OPO_RETRIES_MAX parameter so that Post tries the operation more times. At the same time, increase the SP_OPO_RETRY_DELAY_TIME parameter to increase the time interval between the attempts. That gives the lock or other blocking operation enough time to be resolved between attempts.

If the Post process is set to continue on error (SP_SYS_SUSPEND_ON_ERROR=0) or if the error message is listed in the **oramsglist** file, Post moves on to the next transaction in the queue. In all other cases, Post stops after it reaches the maximum allowed attempts.

IMPORTANT: Reducing this parameter can cause the data to accumulate in the queues, possibly causing them to exceed the available disk space.

For more information, see SP_OPO_RETRIES_MAX in the SharePlex Reference Guide.

Handle Transactions that Contain Out-of-sync Operations

You can configure Post to handle out-of-sync conditions. The following explains the default and alternate behaviors.

Default Post handling of out-of-sync errors

The default Post behavior when a transaction contains an out-of-sync operation is to continue processing other valid operations in the transaction to minimize latency and keep targets as current as possible. Latency is the amount of time between when a source transaction occurs and when it is applied to the target. Different factors affect the amount of latency in replication, such as unusually high transaction volumes or interruptions to network traffic.

Post logs the SQL statement and data for the out-of-sync operation to the `ID_errlog.sql` log file, where *ID* is the database identifier. This file is in the **log** sub-directory of the variable-data directory on the target system.

Stop on out-of-sync errors

You can configure Post to stop when it encounters an out-of-sync condition by setting the following parameter to 1:

- **Oracle targets:** `SP_OPO_OUT_OF_SYNC_SUSPEND`
- **Open Target targets:** `SP_OPX_OUT_OF_SYNC_SUSPEND`

If you use this feature, make certain to monitor replication frequently. If Post stops, latency increases and data accumulates in the queues. For more information, see the parameter documentation in the SharePlex Reference Guide.

Roll back the transaction if it generates out-of-sync errors

You can configure Post to roll back and discard a transaction if any operation in that transaction generates an out-of-sync error. The entire transaction is logged to a SQL file, but not applied to the target.You can edit the SQL file to fix the invalid DML and then run the SQL file to apply the transaction. This feature is enabled by setting the `SP_OPO_SAVE_OOS_TRANSACTION` to 1.

For more information, see the parameter documentation in the SharePlex Reference Guide.

14

Configure Data Transformation

This topic contains instructions for using the transformation feature of SharePlex. Transformation enables SharePlex to manipulate data before, or instead of, posting it to a target.

Contents

Overview of Transformation

Transformation directs the Post process to call a PL/SQL procedure (de fined as a transformation routine) instead of applying a SQL operation to the target database. Transformation enables replicated data to be manipulated before, or instead of, posting to a target.

For example, if a source table and its target table are dissimilar in construction — like when a person's first and last name are in one column in the source table but in separate columns in the target table — you can write a transformation routine to convert the data for those columns so that replication succeeds. You can use transformation routines to convert data types, units of measurement, or character sets. You can use them instead of database triggers to reduce I/O overhead, and for many other business requirements.

When you specify transformation for a table, Post takes no action on the replicated data. Instead, it passes data values to your transformation routine, enabling you to control both the form and destination of the data with the procedure. You can post to the target table, post to an alternate location, or both. Therefore, when writing your routine, is your responsibility to include in your procedure the necessary SQL operations for posting.

Supported sources

Oracle

Supported targets

Oracle

Supported replication strategies

Transformation is a convenient way to use SharePlex to transfer data from one table to another without concern for maintaining identical structure or data. This makes it practical for reporting, broadcast, and data mart and warehousing applications.

Transformation is not suitable for peer-to-peer or high-availability replication environments. High availability requires identical databases that are kept synchronized by replication. For peer-to-peer replication SharePlex must

be able to detect and resolve conflicts when there are concurrent changes to the same record. When data is transformed, SharePlex cannot compare before and after values to verify synchronization and detect conflicts.

Supported operations

Transformation supports only INSERT, UPDATE and DELETE operations. You can do the following when developing procedures:

- You can create one procedure for all three operation types, or you can create a procedure for each operation type.
- You can use one procedure for all tables, or use different procedures for different tables. SharePlex allows this through the use of wildcards to specify the tables.

If a transformation routine is specified for an individual table, and the table also is part of a group of tables for which another routine is specified, only the table-specific routine is used for that table when the associated DML operation occurs.

Considerations When using Transformation

Because transformation changes data and because SharePlex does not post the data, transformation changes the behavior of replication. It is a customization of SharePlex processing. Review the following considerations before implementing transformation to ensure that your transformation procedures succeed.

Privileges

Any table that will be accessed through PL/SQL for transformation requires implicitly granted privileges from the owner of the object to SharePlex.

Keys

A PRIMARY or UNIQUE key is required for all tables using transformation. SharePlex locates the target row for UPDATEs and DELETEs by using the key, which enables it to return values to your transformation routine from the target table for UPDATE operations. Do not allow keys to be changed on the target system, or SharePlex will not be able to locate the row to pass values to your routine.

Test your routines

Test your transformation routines before you put them into production to make sure they work as intended, and to make sure that one routine does not counteract another one. When data is transformed, SharePlex cannot compare before and after values to verify synchronization, which is a measure of whether the routines are performing correctly. The only way to confirm synchronization is to use the **compare** command with the **key** option. This option restricts the comparison to just the key values and is not a complete confirmation of synchronization. It only detects missing or extra rows. It does not (and cannot) indicate whether values in non-key columns are properly correlated to those in the source columns, because the target data was transformed.

For those reasons, the **repair** command cannot be used to resynchronize data. You must devise your own resynchronization procedures based on your company's business rules and the database environment.

Dates

The default date format for SharePlex transformation is MMDDYYYY HH24MISS. Tables with default dates must use that format, or transformation will return errors. Before creating a table with a default date, use the following command to change the date format in SQL*Plus.

```
ALTER SESSION SET nls_date_format = 'MMDDYYYYHH24MISS'
```

Other considerations

- Transformation does not support changes to LOB and LONG columns.
- The processing overhead for passing data to your procedure, combined with that of executing the procedure itself, degrades overall performance on the target system compared to normal replication and posting.
- The same PL/SQL package is used for both generic conflict resolution and transformation (its name is **sp_cr**). **Use either generic conflict resolution or transformation for a table, but not both.** Transformed tables cannot be compared by SharePlex and conflict resolution cannot succeed. If both are used, SharePlex only calls the transformation routine. If appropriate, you can use generic conflict resolution and transformation for different tables in the same configuration. For more information, see Configure Peer-to-Peer Replication on page 165.

Deploy Transformation

Deployment of transformation involves the following steps:

1. Create configuration entries for the source and target tables to be transformed. There are no special configuration procedures for tables that use transformation. Configure them as you would any other table. For more information, see Create a Configuration File on page 62.
2. Create transformation routines. For more information, see Create transformation routines on page 232.
3. Create a transformation file, which directs SharePlex to call the transformation routines. For more information, see Create the transformation file on page 238.

Create transformation routines

Write your transformation routines with dynamic PL/SQL procedural language. Use parameters and record and table structures defined in the public package named **sp_cr**. This package was installed in the SharePlex schema in the database. The package uses the following parameters.

Procedure interface

Follow this template to create your procedure.

(**table_info in out***splex*.**sp_cr.row_typ**, **col_values in***splex*.**sp_cr.col_def_tabtyp**)

where:

- *splex* is the SharePlex schema.
- **sp_cr** is the name of the package that contains the PL/SQL record and table structures.
- **row_typ** is the name of the PL/SQL record that passes in/out variables (see Package definition).
- **col_def_type** is the name of the PL/SQL table that stores column information (see col_def_type table).

Package definition

SharePlex defines PL/SQL record and table structures in a public package named **sp_cr** in the SharePlex database schema. The package uses the following parameters.

```
CREATE SCHEMA IF NOT EXISTS sp_cr;

CREATE TYPE sp_cr.row_typ AS

(src_host VARCHAR(32),

src_db VARCHAR(32),

src_time VARCHAR(20),

statement_type VARCHAR(6),

source_table VARCHAR(128),

target_table VARCHAR(128),

native_error INTEGER,

sql_state VARCHAR(10)

);

CREATE TYPE sp_cr.col_def_typ AS

(column_name VARCHAR,

datatype VARCHAR,

is_key BOOLEAN,

is_changed BOOLEAN,

old_value VARCHAR ,

new_value VARCHAR

);

CREATE SEQUENCE EXC_SEQ START WITH 1 INCREMENT BY 1 MINVALUE 1 CACHE 20 NO CYCLE ;
```

IN variables

For each row operation that causes a conflict, SharePlex passes this metadata information to your procedure.

Variable	Description
src_host	The name of the source system (where the operation occurred). It is case-sensitive and is passed using the same case as on the source system, for example **SysA**. If there are named post queues in use on the target system, this variable consists of the name of the post queue, for example **postq1**. **NOTE:** Maximum length is 32 characters. A host name longer than 32 will be truncated to 32 characters.
src_ora_sid	The ORACLE_SID of the source database. It is case-sensitive and is passed in the same case as in the **oratab** file or V$PARAMETER table.
src_ora_time	The timestamp of the change record in the source redo log.
source_rowid	The row ID of the source row. It is passed as a literal within single quotes, for example '123456'.
target_rowid	The row ID of the corresponding row in the target database. SharePlex obtains the row ID by querying the target database. It is passed as a literal within single quotes, for example '123456'. If the row cannot be found using the PRIMARY key, the value is NULL.
statement_type	A letter, either I, U or D, indicating whether the operation is an INSERT, UPDATE or DELETE statement.
source_table	The owner and name of the source table, expressed as *owner.table*. This value is case-sensitive and matches the way the table is named in the database. It is passed within double quotes, for example "scott"."emp."
target_table	The owner and name of the target table, expressed as *owner.table*. This value is case-sensitive and matches the way the table is named in the database. It is passed within double quotes, for example "scott"."emp."
oracle_err	This is different, depending on whether the procedure is being used for conflict resolution or transformation. **Transformation**: SharePlex passes a value of 0 for this variable. This variable is only used for conflict resolution. **Conflict resolution:** The Oracle error number that caused the conflict.

OUT variables

These variables direct the action of SharePlex based on whether the procedure succeeded or failed).

Variable	Description
status	Defines whether or not the procedure succeeded. You must specify a value for this parameter.

Variable	Description
	• A value of 0 implies successful execution. It acts differently, depending on whether the procedure is used for conflict resolution or transformation. **Transformation**: Post does not write any SQL. SharePlex does not write any error messages to the Event Log when transformation succeeds. It continues its processing by reading the next replicated operation in the post queue. • **Conflict resolution**: A value of 0 directs SharePlex to proceed with the SQL statement. SharePlex does not write any log entries to the Event Log when conflict resolution succeeds. • A value of 1 implies that the procedure was unsuccessful. In this case, the action SharePlex takes depends on what you specified as the action variable. • (**Transformation** only) A value of 7 implies unsuccessful execution and instructs the Post process to stop.
action	Defines the action that you want SharePlex to take. This is different, depending on whether the procedure is used for transformation or conflict resolution. **Transformation**: You must specify a value of 0 for this parameter, which directs SharePlex NOT to post the SQL statement. Your transformation routine is responsible for posting the results of the transformation either to the target table or another table. The outcome of this action depends on what you specify for the reporting variable **Conflict resolution**: Specifies the action to take as a result of an *unsuccessful* conflict resolution procedure. You must specify a value for this parameter. • A value of 0 directs SharePlex NOT to post the SQL statement. The outcome of this action depends on what you specify for the **reporting** variable. In addition, it directs SharePlex to try the next conflict resolution procedure that you listed in the conflict resolution file, if one exists. • The value of 1 is reserved for internal SharePlex use. Do not use it. • A value of 2 directs SharePlex to try the next conflict resolution procedure that you listed in the conflict resolution file, if one exists.
reporting	Determines how SharePlex reports unsuccessful procedural results. You must specify a value for this parameter. • A value of 0 directs SharePlex NOT to report an error or write the failed SQL statement to the *SID*_**errlog.sql** log. • Values 1 and 2 are reserved for internal SharePlex use. Do not use them.

Variable	Description
	• A value of 3 directs SharePlex to write the failed SQL statement to the *SID*_**errlog.sql** log and report an error to the Event Log.

col_def_type table

SharePlex creates a **col_def_tabtyp** PL/SQL table for each replicated operation. This table stores column information. It is different depending on whether the procedure is used for transformation or conflict resolution.

- **Transformation**: For each row operation, SharePlex writes column information to **col_def_type**.
- **Conflict resolution**: For each row operation that causes a conflict, SharePlex writes column information to **col_def_tabtyp**.

All fields are passed by SharePlex to your routine, although not all will have values if SharePlex cannot locate the row.

Following is the data type that is used to populate the **col_def_tabtyp** table.

```
CREATE SCHEMA IF NOT EXISTS sp_cr;

CREATE TYPE sp_cr.row_typ AS

(src_host VARCHAR(32),

src_db VARCHAR(32),

src_time VARCHAR(20),

statement_type VARCHAR(6),

source_table VARCHAR(128),

target_table VARCHAR(128),

native_error INTEGER,

sql_state VARCHAR(10)

);

CREATE TYPE sp_cr.col_def_typ AS

(column_name VARCHAR,

datatype VARCHAR,

is_key BOOLEAN,

is_changed BOOLEAN,

old_value VARCHAR ,

new_value VARCHAR

);

CREATE SEQUENCE EXC_SEQ START WITH 1 INCREMENT BY 1 MINVALUE 1 CACHE 20 NO CYCLE ;
```

Description of col_def_tabtyp

Column	Description
column_name	Tells your procedure the name of the column that was replicated from the source table, for example **emp_last_name**. This value is not case-sensitive.
data type	Tells your procedure the data type of the data in the replicated column, for example VARCHAR2. This value is always in capital letters.
is_key	Tells your procedure whether or not the column is a key column. If it is a key column, SharePlex passes a value of **TRUE**. If the column is not part of a key, SharePlex passes a value of **FALSE**.
is_changed	Tells your procedure whether or not the column value has changed. If it is changed, SharePlex passes a value of **TRUE**. If the column is not changed, SharePlex passes a value of **FALSE**. • For INSERTs, **is_changed** is **TRUE** for non-NULL values, because none of the columns existed in the database. If a **NULL** value is inserted, **is_changed** is **FALSE**. • For UPDATEs, **is_changed** is **TRUE** for non-key columns. For key columns, **is_changed** normally is **FALSE**, but SharePlex will pass a value for a changed key column. **Conflict resolution only**: If a key value also was changed on the target system, SharePlex will not be able to locate the correct row, and conflict resolution could fail. • For DELETEs, **is_changed** is always **FALSE**, because SharePlex replicates only the key values for a DELETE statement.
old_value	Tells your procedure the old value of the replicated column, before it was changed on the source system. This column is **NULL** for INSERTs, because the row did not exist in the target database before the INSERT. **Conflict resolution only**: This is the pre-image against which SharePlex compared the source and target columns as part of its synchronization check for UPDATEs and DELETEs. If the old value passed by SharePlex does not match the **current_value** value obtained from the target row, then there is a conflict.
new_value	Tells your procedure the new value of the replicated column, as changed on the source system.
current_value	Tells your procedure the current value of the column in the target table. If SharePlex cannot locate the target row, the value is **NULL**.

Example entries in col_def_tabtyp table per operation type

The following tables illustrate the possible outcomes of each type of operation.

INSERT operation

column_name	is_changed	old_value	new_value	current_value[1]	is_key
C1	TRUE	NULL	bind	NULL	FALSE
C2	TRUE	NULL	bind	NULL	TRUE
C3	FALSE	NULL	NULL	NULL	TRUE | FALSE

[1] When an INSERT fails, it is because a row with the same PRIMARY key already exists in the target database. SharePlex does not return the current value for INSERTs.

UPDATE operation

column_name	is_changed	old_value	new_value	current_value[1, 2]	is_key
C1	TRUE	bind	bind	NULL | *target_value*	FALSE
C2	FALSE	bind	NULL	NULL | *target_value*	TRUE
C3	TRUE	bind	bind	NULL | *target_value*	TRUE

[1] (Conflict resolution) When an UPDATE fails, it is because SharePlex cannot find the row by using the PRIMARY key and the pre-image. If the row cannot be found, SharePlex searches for the row by using only the PRIMARY key. If SharePlex finds the row, it returns the current value for the key column as well as the changed columns. If SharePlex cannot find the row by using just the PRIMARY key, then SharePlex returns a **NULL**.

[2] (Transformation) For an UPDATE, SharePlex cannot locate a row using the PRIMARY key and the pre-images, because the pre-images are different due to transformation. As an alternative, it searches for the row using just the PRIMARY key. If it finds it, SharePlex returns the current value for the key column as well as the changed columns. If it cannot locate the row using just the PRIMARY key, then current_value is NULL

DELETE operation

column_name	is_changed	old_value	new_value	current_value[1]	is_key
C1	FALSE	bind	NULL	NULL	TRUE

[1] When a DELETE fails, it is because SharePlex could not find the row by using the PRIMARY key. Therefore, SharePlex returns a NULL.

Create the transformation file

To direct SharePlex to call transformation routines instead of posting SQL operations, use the **transformation**.*SID* file, where *SID* is the ORACLE_SID of the target database. Before executing a SQL operation, Post reads this file to determine if there is a transformation routine that it must call.

Where to find this file

A blank **transformation**.*SID* file, where *SID* is the ORACLE_SID of the target instance, was included in the **data** sub-directory of the SharePlex variable-data directory when SharePlex was installed. Use the file on the target system.

If this file does not exist, you can create one in ASCII format in an ASCII text editor. It must be named **transformation**.*SID*, where *SID* is the ORACLE_SID of the target instance.

NOTE: The *SID* is case-sensitive.

IMPORTANT! There can be only one **transformation**.*SID* file per active configuration.

How to make entries in the file

Use the following template to link a procedure to one or more objects and operation types.

owner.object	{**i** | **u** | **d** | **iud**}	*owner.procedure*

where:

- *owner.object* is the owner and name of a *target* object, or a wildcarded entry. (See Syntax rules)
- **i**| **u** | **d** is the type of operation to be transformed by the specified procedure. You can specify any or all operation types, for example **id** or **iud**. Upper or lower case are both valid.
- *owner.procedure* is the owner and name of the procedure that will handle the specified object and operation type.

Syntax rules

- There must be at least one space between the object specification, the operation type specification, and the procedure specification.
- You can use the **LIKE** operator and a SQL wildcard (%) to specify multiple objects by using a search string. (See the Example.)
- You can use an underscore (_) to denote a single-character wildcard. For table names that contain an underscore character (for example emp_sal), SharePlex recognizes the backslash (\) as an escape character to denote the underscore as a literal and not a wildcard, for example: **like:scott.%_corp_emp**. If you are not using the LIKE operator, the backslash escape character is not required if an object name contains an underscore.
- You a comment line anywhere in the file. Start a comment line with the pound symbol (#).

Example transformation file

scott.sal	IUD	scott.sal_tr
like:scott.%_corp_emp	IUD	scott.emp_tr1
like:scott%	IUD	scott.emp_tr2
scott.cust	U	scott.sal_tr

How to change the file during replication

You can change the transformation file any time during replication to add and remove tables and procedures. After you change the file, stop and re-start the Post process.

15

Configure Security Features

SharePlex provides a number of security features that help protect replicated data on the local system and during transfer across a network. This chapter provides guidelines for the configuration and use of these features.

NOTE: When replicating data from PostgreSQL to PostgreSQL, SharePlex supports only Secure Data with SSL/TLS and Encrypt Data between Export and Import security features .

Contents

Secure Data with SSL/TLS

SharePlex provides a feature to enable SSL/TLS connections for all network traffic. This will encrypt data on the network between SharePlex instances and between SharePlex and the command line interface.

If SSL/TLS is enabled, SharePlex will only accept SSL/TLS connections. For all SharePlex instances that replicate to each other, either all must have SSL/TLS enabled or all must have SSL/TLS disabled. The SSL/TLS configuration includes a network password. This password must be the same across all of your SharePlex instances.

NOTE: For TLS connections, SharePlex supports TLS 1.2 (or later).

To change the SSL/TLS configuration:

1. Shutdown sp_cop on all nodes
2. Run “sp_security --setup” on all nodes
3. Start sp_cop on all nodes

Use the **sp_security** utility to enable, disable or view the SSL/TLS settings for SharePlex network communication.

Enable SSL/TLS

IMPORTANT! SSL/TLS must be either enabled with a common network password or disabled on all SharePlex installations.

To enable SSL/TLS:

Run `sp_security --setup`, select the **SSL/TLS** option, and then enter a network password.

```
% sp_security --setup

Security Setup Wizard
---------------------
This wizard will walk you through setting up the SharePlex network security.

Setup configuration for '/home/shareplex/var110/' and Port 2100 [N]: Y

Choose your network security model.  Please note the following:
         * Cop must be down when the security model is changed, or when the
network password is changed
         * The same model must be used among all SharePlex nodes replicating
to each other
         * For security model [1], the same network password must be set on
all SharePlex nodes replicating to each other

  [1] Use basic SSL/TLS connections
  [2] Use non-SSL/TLS connections (default prior to SharePlex 9.1.3)

Security model: 1
```

```
Please enter a network password that will be used for authentication
among the SharePlex nodes.  All SharePlex nodes that replicate data to each
other must have the same network password.

Network password:
Please re-enter the network password

Network password:

Security settings:

Configuration for '/home/shareplex/var110/' and Port 2100:

  Security model            : SSL/TLS
  Network password          : stored for unattended startup
  SSL key file password     : stored for unattended startup
  SSL key file              : key.pem
  SSL cert file             : cert.pem

Setup complete!
```

Disable SSL/TLS

IMPORTANT! SSL/TLS must be either enabled with a common network password or disabled on all SharePlex installations.

To disable SSL/TLS:

Run “sp_security --setup” and select non-SSL/TLS connections.

```
% sp_security --setup

Security Setup Wizard
---------------------
This wizard will walk you through setting up the SharePlex network security.

Setup configuration for '/home/shareplex/var110/' and Port 2100 [N]: Y

Choose your network security model.  Please note the following:
         * Cop must be down when the security model is changed, or when the
network password is changed
         * The same model must be used among all SharePlex nodes replicating
to each other
         * For security model [1], the same network password must be set on
all SharePlex nodes replicating to each other

  [1] Use basic SSL/TLS connections
  [2] Use non-SSL/TLS connections (default prior to SharePlex 9.1.3)
```

```
Security model: 2

Security settings:

Configuration for '/home/shareplex/var110/' and Port 2100:

  Security model            : Un-encrypted

Setup complete!
```

View current SSL/TLS configuration

To view the current SSL/TLS configuration:

Run “sp_security --show”.

```
% sp_security --show

Security settings:

Configuration for '/home/shareplex/var110/' and Port 210:

  Security model            : Un-encrypted
```

Host Authentication

SharePlex provides host authorization security that verifies whether or not SharePlex processes on specific remote systems are authorized to connect to the local system for service and command requests. To implement host authorization, you create an ASCII text file named **auth_hosts** in the **data** sub-directory of the SharePlex variable-data directory and then populate it with the names of systems being granted connection permission.

Requirements

- If used, the **auth_hosts** file must contain valid entries. If this file exists but is empty or contains invalid entries, SharePlex sends an error message similar to the following example to the Event Log: `unauthorized connection attempt.`
- If an **auth_hosts** file does not exist on a system, SharePlex accepts all requests from all systems that attempt to connect to **sp_cop**.
- The name of the local system must be the first non-commented line of this file, or host authorization will not function.
- All entries, including comments, must end with a return.

To configure the auth_hosts file:

NOTE: Begin comment lines with a pound character (#).

1. Run an ASCII text editor such as vi (Unix and Linux) to open a blank file. If you are using a Unix and Linux text editor, change directories to the **data** sub-directory of the SharePlex variable-data directory before you run the editor.
2. On the first non-commented line, enter the full machine name of the local system, for example:**Localhost.mycorp.com**.
3. On the next non-commented line, enter one of the following:

Value	Description
all	Grants connection authorization to processes on all remote systems.
hostname	Grants connection authorization to the specified host. Enter the fully qualified machine name, for example **remotehost.mycorp.com**. Specify as many host names as needed, each on its own line.

4. Save the file as **auth_hosts** in the **data** sub-directory of the SharePlex variable-data directory. If running multiple instances of **sp_cop**, make certain to save the file to the correct variable-data directory.

Example

Note the name of the local host is on the first non-commented line.

```
#Comment: first line is local host name.

Localhost.mycorp.com
```

```
#Comment: remaining lines are remote hosts.

remotehost.mycorp.com

remotehost2.mycorp.com

remotehost3.mycorp.com
```

Secure Data with SSH

SharePlex uses the SSH® Secure ShellTM utility to provide encryption for network services such as secure remote login and other services over an insecure network.

Requirements

- Purchase and install the SSH software. SSH is not included with SharePlex.
- Using SSH with SharePlex requires the use of local port forwarding (also known as tunneling) within the SSH configuration. Port forwarding allows you to establish a secure SSH session and then tunnel TCP connections through it.
- SharePlex can be configured to work with SSH software between a source system and one target system. If a source replicates to multiple targets, only one of the routes can be configured with SSH.
- This feature is supported on Unix and Linux.

To set up SharePlex to use SSH:

1. On the source and target systems, choose an available local port to be used as the tunnel port. For peer-to-peer and high availability replication, the port must be the same number on both systems. For other replication strategies, choose a different port on each system.
2. On the source system, issue the following command from the command prompt. This command connects to the target system to set up the tunnel.

   $ **ssh -L** *source_port*:*target_host*:*target_port userid*@*target_host* **-N -f**

   where:

   - **-L** specifies that the specified port on the local host (acting as the client) is to be forwarded to the remote host and port.
   - *source_port* is the port number on the source system.
   - *target_host* is the name of the target system.
   - *target_port* is the port on the target system.
   - *userid* is your Unix and Linux user ID. You will be prompted for the password.
   - **-N** specifies not to execute a remote command. This is used just to forward a port (protocol version 2 only).
   - **-f** forces the SSH shell to work in the background just before command execution. If this argument is omitted, the terminal window you are using must be kept open. SSH cannot be started with **nohup**.

   Refer to your SSH documentation for more information about these commands.

3. (If using multiple SharePlex instances) On the source system, export the correct variable-data directory for the instance of **sp_cop** for which you are setting up SSH.

   ksh shell:

   export SP_SYS_VARDIR=*/full_path_of_variable-data_directory*

   csh shell:

   setenv SP_SYS_VARDIR=*/full_path_of_variable-data_directory*

4. On the source system, start **sp_cop**.
5. On the source system, run **sp_ctrl** from the **bin** subdirectory of the product directory.
6. In **sp_ctrl**, set the SP_XPT_USE_LOCALHOST parameter in one of the following ways.

   - If there is only one target system, set the parameter with the following syntax:
     `sp_ctrl>` **set param SP_XPT_USE_LOCALHOST 1**
   - If there are multiple targets, use the following command to set up a tunnel to the target that will use SSH. Replication to the other target systems will connect directly in the normal fashion.
     `sp_ctrl>` **set param SP_XPT_USE_LOCALHOST to** *host* **1**

   where: *host* is the name of the target system that will use the tunnel.

7. In **sp_ctrl**, use the **list param** command with the **modified** option to verify the parameter setting. If the setting is correct, you can activate a configuration at this point.

   `sp_ctrl>` **list param modified**

8. If there is an active configuration, stop and then start **sp_cop** to make the new parameter setting active.

   To stop sp_cop:

   sp_ctrl**> shutdown** /*productdir*/**bin**/**sp_cop &**

   To start sp_cop:

   `$` **/***productdir***/bin/sp_cop &**

Encrypt Data between Export and Import

If you do not want to use SSL/TLS but still want to encrypt data between Export and Import, you might do so using this feature.

SharePlex can be configured to encrypt replicated data across the network. SharePlex uses Advanced Encryption Standard (AES) encryption.

Encryption guidelines

Encryption must be enabled on the source and target systems. You enable encryption and set the size of the key through the Export process. You configure the Import process to ensure that encryption is enabled on the source, so that no data is sent across the network unless it is encrypted.

When configuring encryption, follow these guidelines:

- Use one encryption key for all Export processes in the SharePlex instance.
- To use encryption, SharePlex must be version 9.1 or later.

Encryption procedure

On the source system:

1. Set the Export parameter **SP_XPT_ENABLE_AES** to 1. This enables encryption.

   ```
   sp_ctrl> set param sp_xpt_enable_aes 1
   ```

2. Run the **create encryption key** command to create the key.

   ```
   sp_ctrl> create encryption key
   ```

   The following is an example key:
   ```
   E5F5D4CBA329D2C86B5D7ABA096C18600595490129F55A1422AAB0248B28D0E4
   ```

3. (Optional) Set the SP_XPT_AES_KEY_LENGTH parameter to increase the key size.

   The **create encryption key** command returns a randomly generated, 256-bit AES key. By default, SharePlex uses 128 bits of that length to encrypt the data.

   To increase the key length that SharePlex uses, set the **SP_XPT_AES_KEY_LENGTH** parameter to 192 or 256 bits. When you increase the length, the key is harder to hack but requires more CPU power.

   ```
   sp_ctrl> set param sp_xpt_aes_key_length {192 | 256}
   ```

   Example: **set param sp_xpt_aes_key_length 256**

4. Run the **set encryption key** command. This adds the key to the Export configuration.

   ```
   sp_ctrl> set encryption key key_value
   ```

   Example: **set encryption key E5F5D4CBA329D2C86B5D7ABA096C18600595490129F55A1422AAB0248B28D0E4**

5. Restart Export to activate the settings.

   ```
   sp_ctrl> stop export
   sp_ctrl> start export
   ```

On the target system:

1. Set the **SP_IMP_ENABLE_AES** parameter to 1. This prevents SharePlex on the target from accepting data that is not encrypted.
2. Run the **set encryption key** command with the same key value that you set for Export. The key values on the source and target must match.

   ```
   sp_ctrl>set encryption key key_value
   ```

   Example: **set encryption key E5F5D4CBA329D2C86B5D7ABA096C18600595490129F55A1422AAB0248B28D0E4**

3. Restart Import to activate the settings.

   ```
   sp_ctrl> stop import
   sp_ctrl> start import
   ```

View the encryption key

Issue this command on the source and target systems to ensure that both key values match.

```
sp_ctrl> show encryption key
```

FIPS Compliance

SharePlex installations can be run on FIPS-enabled servers on the Linux platform. No FIPS-specific configurations are required to run SharePlex installations on FIPS-enabled servers.

Assign SharePlex Users to Security Groups

Contents

Overview

The SharePlex security groups provide access control to the SharePlex command and control system. Without proper configuration of these groups, anyone with permissions on the system can use the commands that view, configure, and control data replication.

About the SharePlex Security Groups

To monitor, control, or change SharePlex replication, a person must be assigned to one of the SharePlex security groups on the systems where he or she will be issuing commands. Each group corresponds to an authorization level, which determines which SharePlex commands a person can issue. To execute a command, a user must have that command's authorization level or higher. Use the **authlevel** command to determine your authorization level for issuing SharePlex commands on a system.

Description of the SharePlex security groups

Refer to the following table to determine the group and authorization level that you want to grant each SharePlex user.

User Authorization Levels and Roles

Auth level	User type	User group	User roles
1	Administration	spadmin*	You need at least one user with Administrator rights on each source and target system. Can issue all SharePlex commands. Commands that can *only* be issued by a SharePlex Administrator are: • **startup, shutdown** • all configuration commands relating to an active configuration • all parameter commands except **list param** • **start capture** • **stop capture** • **abort capture** • **truncate log** The SharePlex Administrator user must be in the Oracle **dba** group. For Oracle RAC and ASM 11*g*R2 and above, the user must also be in the Oracle Inventory group. For example: **$ useradd –g spadmin –G dba,oinstall**. The membership in Oracle Inventory group must be listed explicitly in the **etc/group** file. On Unix and Linux, unless you install SharePlex as a root user, the SharePlex Administrator user and the SharePlex **admin** group must exist prior to installation.
2	Operator	spopr	Can issue all SharePlex commands except those listed above.
3	Viewer	spview	Can view lists, status screens, and logs to monitor replication only.

NOTE: The default name for the SharePlex administrator group is **spadmin**, but you can designate any group or specify any name for that group during installation.

Create and Populate SharePlex Groups on Unix and Linux

Where and when to create the SharePlex groups on Unix and Linux depends on whether you install SharePlex as a root or non-root user.

- If you install as non-root, create the groups in the **/etc/group** file before you run the SharePlex installer. In a cluster, create them on all nodes.*
- If you install SharePlex as a root user, you can direct the installer to create the groups in the **/etc/group** file. If you install in a cluster, the installer creates the groups on the primary node, but you must create them yourself on the other nodes.

  * The groups must exist because the installer adds the SharePlex Administrator user to the **spadmin** group during the installation process. In a cluster, this user is only added to the primary node. You must add the SharePlex Administrator user to the other nodes.

To create the groups in /etc/group :

 # groupadd spadmin

 # groupadd spopr

 # groupadd spview

To assign a user to a group:

1. Open the **/etc/group** file.
2. Add the Unix or Linux user name to the appropriate group. To assign a list of user names to a group, use a comma-separated list (see the following example).

   ```
   spadmin:*:102:spadmin,root,jim,jane,joyce,jerry
   ```

   If the password field is null, no password is associated with the group. In the example, the asterisk (*) represents the password, “102” represents the numerical group ID, and **spadmin** is the group. The group ID must be unique.
3. Save the file.

Users can verify their authorization levels by issuing the **authlevel** command in **sp_ctrl**.

16

Start Replication on your Production Systems

This chapter contains instructions for the initial startup of replication from a source database to a target database on production systems.

Contents

What is Activation?

When you activate a configuration, through the activate config command in sp_ctrl, SharePlex does the following:

- Activates (read) the configuration file to build a series of internal structures that identify objects and routes. Only one configuration can be active for any given datasource at a time. Configurations for different datasources on a system can be active at the same time. For example, you can activate a configuration for each Oracle instance.
- Starts the processes that maintain the capture and replication of source transactions.
- (Oracle only) Reconciles replicated data with the copied data. SharePlex applies transactions that occurred after the copy was taken and discards transactions that occurred before the copy (and thus were applied by the copy and would be redundant if applied by replication). The reconcile process is only required for procedures that start replication while the source database is active.

The activation of a configuration generally proceeds as follows:

1. Assign an activation ID

SharePlex assigns an activation ID number to each configuration activation and its associated replication processes and queues. A configuration can be activated many times, and this ID keeps track of each one.

2. Create an object cache

SharePlex builds an object cache that records the standard metadata needed to support replication: the name, size, and type of columns, NOT-NULL constraints, and whether a column is part of a key. For tables using partitioned replication, additional information is stored.

3. Add a configuration change marker

SharePlex places a configuration-change marker in the data stream. This marker directs **sp_cop** to generate a new set of replication processes and queues. If another configuration is active for the same datasource, the marker deactivates it, causing the removal of the old processes and queues after the data they contain is posted.

4. Lock the tables, add the activation marker, unlock

(Oracle only) SharePlex locks the tables that are listed in the configuration file so that it can obtain information about them while they are in a read-consistent state. As many tables can be locked concurrently as there are locking threads available. When SharePlex locks a table, it places an activation marker in the data stream that tells the Capture process to start (or stop) replicating that table.

> **NOTE**: If an application uses NOWAIT locking on tables in the replication configuration, the NOWAIT could fail if it attempts to obtain a lock on an object that is already locked because it is being activated.

SharePlex locks the following:

- All tables added to replication (new and reactivated configurations)
- All tables removed from replication (reactivated configurations)
- All tables where routes changed (reactivated configurations)

Each table is locked for a very short time, just long enough to activate a table. Replication of each table begins as soon as its activation is complete. Should one or more table fail to activate, SharePlex continues with the activation of the other tables. Users can access the data in a source table when the activation lock is released.

Activation Commands

Use **sp_ctrl** commands to activate, deactivate and view information about a configuration activation, as well as to reconcile ongoing changes with a copy. For more information about these commands, see the SharePlex Reference Guide.

Purpose	Command
Activate a configuration file	**activate config** Starts the activation and retains control of the **sp_ctrl** command line until the activation is finished.
View configuration statistics	**show config** Shows statistics for active configurations. It displays the status of a configuration (active or inactive), the datasource, the date and time the configuration was activated, and the number of objects that are configured.
Deactivate a configuration file	**deactivate config** or **abort config** The purposes of these commands are different, but both deactivate a configuration file. • **deactivate config** gracefully terminates replication for an active configuration. It stops all Capture activity for the configuration, posts all data currently in the queues, and removes the associated SharePlex processes and queues. • **abort config** is a forceful deactivation. It stops all replication activity for the configured datasource on the source and target systems, whether or not data in the queues has been posted. Deactivating or aborting a configuration stops replication. If users continue making changes to the configured objects, the source and target data can go out of synchronization.
Reconcile replicated changes with the copy	**reconcile** (Valid for an Oracle source only) Coordinates the results of ongoing replication with a copy of the source data that is applied to the target system, so that changes that occurred before the copy are discarded.
View replication status	**status** Shows a summary of the status of replication to help you ensure that processes are running and to check for errors, warnings or notices.
View queue status	**qstatus** Shows statistics for the capture, post, and export queues.

Requirements for Activating a Configuration

Activation of a configuration is an event that requires a series of actions to be taken in a timely manner, uninterrupted. Therefore, it is important to have all of your preparations done ahead of time, leaving nothing in question. You can activate a configuration if you performed the minimal requirements outlined in this topic.

NOTE: These instructions apply to Oracle source databases, unless otherwise noted.

Required authorization level

Only a SharePlex Administrator can activate a configuration. Additional users should be assigned to monitor SharePlex and perform basic operational procedures. For more information, see Assign SharePlex Users to Security Groups on page 251.

Required setup

- Before you activate a configuration, make certain that the objects that you want to replicate exist in the source database.
- If a table will be partitioned, create those partitions before you activate a configuration to begin replication processing. Partitioning a table while it is actively replicating causes SharePlex to lose the identifying information it has compiled, and DML from that table partition will not be replicated. You can add a partition to a table already in replication, but you will need to reactivate the configuration to update that table in the SharePlex object cache. For additional information related to adding or dropping the partitioned table for PostgreSQL, see the **Activate Config for PostgreSQL** section in the SharePlex Reference Guide.

Prerequisites:

Make certain you satisfy the following prerequisites before you activate a configuration.

Requirement	Documentation to Read
Understand how to start and stop the **sp_cop** program.	Run SharePlex
Understand how to issue SharePlex commands.	Execute Commands in sp_ctrl
Understand the commands you will use during activation.	Activation Commands
Make certain your SharePlex configuration and setup are complete and any optional features are included in the configuration or setup.	Configure SharePlex to Replicate Data Configure Partitioned Replication Configure Named Queues Configure Security Features Configure Data Transformation Configure Error Handling

Requirement	Documentation to Read
Prepare the database to support replication.	See "Set up an Oracle environment for replication" in the *Installation and Setup Guide for an Oracle Source*
Plan and configure SharePlex to support your replication strategy	Configure a Replication Strategy
Start the source and target databases.	Database documentation
Unless your applications only generate SharePlex-supported DDL, prevent DDL operations, including TRUNCATE, during activation. Where permitted, DML changes are the only permissible changes during activation.	Database documentation

Test the Configuration before Activation

Before you activate a configuration in production, perform a test activation by using the **verify config** command. If there are syntax errors, misspellings or duplicate entries in the configuration file, the entire activation will fail.

This command will test the configuration to find and report the following conditions:

- Verify the syntax of the entries in the configuration file.
- Report an error if the source object is not supported for replication by SharePlex.
- Report if a host name specified in a route is unreachable.
- Report if there are duplicate specifications for a single object.
- Report if an object specification will be skipped and the reason why.

The **verify config** command does not verify how long the activation will take, nor will it verify the target objects or database connection (as represented by the database identifier listed in the routing map.)

For more information, see the **verify config** command in the SharePlex Reference Guide.

Frequently Asked Questions about Activation

What happens during activation?

During activation, SharePlex gets the information it needs to identify and understand the objects that are configured for replication and build routing maps. For more information, see What is Activation? on page 255.

How long does activation take?

The length of time that activation takes varies, depending on the size, number and structure of the configured objects.

Do users have to stop access to the data?

An Oracle database can remain available for transactions. Each source table is briefly locked so SharePlex can build its internal object information. Then the lock is released. This happens very quickly and may not even be noticeable by users. However, if a business application has NOWAIT locking, the SharePlex lock may cause the application to fail if it attempts to obtain a lock on an object that is being activated.

Can DDL be performed during activation?

No. The definition of objects cannot be changed during activation.

Can I activate multiple configurations for the same datasource?

Yes. You can create different configurations for the same datasource. You must use multiple instances of SharePlex (different **sp_cop** processes and variable-data directories, running on different ports). See Run Multiple Instances of SharePlex on page 48

Can I activate configurations for different datasources at the same time?

Yes. For more information, see How to Activate Multiple Configuration Files on page 261.

Can I test a configuration before I activate it?

Yes. This is highly recommended. Use the **verify config** command. For more information, see Requirements for Activating a Configuration on page 258.

Can I interrupt an activation to make changes?

No. Activation is meant to be an uninterrupted procedure that initiates replication while maintaining the same series of transactional events as those appearing in the redo log. You can use the **abort config** command to terminate an activation, but you may then need to clean out the queues and resynchronize the source and target data again. Ideally, you should have a tested configuration file ready to be activated and be prepared to issue commands to the database and to SharePlex.

Can I activate against a quiet database?

Yes. A quiet database is optional to activate Oracle capture. The Oracle database must be mounted and open, however.

Can I increase the number of activation threads?

Yes, but only for an Oracle source. The number of threads is controlled globally by the `SP_OCF_THREAD_COUNT` parameter, which must be set before you issue the **activate config** command. You can override this value for a particular activation by using the **threads** option when you issue the **activate config** command.

How to Activate Multiple Configuration Files

You can activate one configuration *per datasource* on a system. For example, if there are ConfigA, ConfigB and ConfigC for the same datasource, you can activate only one of them at a time. Activating another configuration for the same datasource automatically deactivates the first one.

However, if ConfigA replicates data from one datasource, while ConfigB replicates data from a different datasource, you can activate both of those configurations at the same time. Replication for those configurations can operate concurrently.

To activate multiple configuration files:

The activation process retains control of the **sp_ctrl** interface until the activation is finished. Because an activation can take a significant amount of time, you can activate different configurations in different sessions of **sp_ctrl**. Activate the first configuration, then open another session of **sp_ctrl** to activate the second one. Open as many sessions of **sp_ctrl** as you have configurations to activate.

The number of activation threads can be controlled for an Oracle source. To set the number of threads, use the `SP_OCF_THREAD_COUNT` parameter, which is a global setting for the instance of SharePlex. When activating more than one configuration concurrently, you can override this value for a particular activation by using the **threads** option when you issue the **activate config** command.

Activate Replication with an Oracle Hot Backup on an Active Database

Use this procedure to use an Oracle hot backup to establish a target Oracle instance and activate replication without quieting the source database. This procedure involves using the **reconcile** command to ensure that transactions which occurred after the point of backup are applied to the target, while eliminating redundant replicated transactions that were already captured by the backup.

Preliminary considerations

Read these points before you proceed.

Supported databases

Oracle source and Oracle target

Supported replication strategies

All replication strategies are supported with the following limitations:

Limitation applies to:	Description
Consolidated replication (many sources to one target)	To establish consolidated replication, the use of a hot backup from all source systems is not possible. A backup from one source will override the data that was applied by a backup from a different source. You can use a hot backup of one of the source instances to establish a target instance, and then use another copy method to apply the objects from the other source instances. Possible methods include: • Export/import. For more information, see Activate Replication with Cold Copy/transfer Methods on page 278. • Transportable tablespaces. For more information, see Activate Replication with Oracle Transportable Tablespaces on page 275.
Peer-to-peer	**To establish peer-to-peer replication, you must:** 1. Quiet all of the systems except the trusted source system for the duration of this procedure. 2. Move all users to the trusted source system, and then follow this procedure. Only after this procedure has been performed on all of the *secondary* systems may users may resume activity on them.

Requirements

- [Unix and Linux systems] Verify that the ORACLE_SID and ORACLE_HOME in the **oratab** file are correct for the instance you will be establishing with the hot backup. The SID must be the SID used in the routing map in the configuration file that you will be activating.
- Read the requirements before you start this procedure. For more information, see Requirements for Activating a Configuration on page 258.
- Make certain a SharePlex database account exists in the source database (only). This account usually is created when SharePlex first is installed. See the SharePlex Installation and Setup Guide for more information.
- Before you start, review this procedure and see the SharePlex Reference Guide for more information about the commands that are used.

Troubleshooting

If the configuration fails to activate, you can find information about the failure in these places:

- Use the **show log** command to view the **event_log**.
- View the activation process log, which is a file named ***SID_oconf##*.log** in the **log** sub-directory of the SharePlex variable data directory.

See also Solve database setup problems for Oracle on page 305.

Procedures

There are two procedures for activation with a hot backup, depending on your replication strategy.

Activation with hot backup: all strategies except cascading

Activation with hot backup: cascading replication

Activation with hot backup: all strategies except cascading

Use this procedure for all replication strategies except cascading replication where SharePlex will be posting to a database on the intermediary system.

Perform the following steps to activate with hot backup for all strategies except cascading:

1. On the source and target systems, go to the **bin** sub-directory of the SharePlex product directory, and start **sp_cop** and **sp_ctrl**. In a cluster, the source is the primary node where the cluster VIP is running.
2. On both systems, verify that the SharePlex processes are running.

   ```
   sp_ctrl> status
   ```

3. On the target system (primary node of a target cluster), stop the Post process. This allows replicated data to accumulate in the post queue until the database has been recovered and reconciled.

   ```
   sp_ctrl> stop post
   ```

4. On the source system, run the Oracle hot backup.
5. When the backup is finished, activate the configuration on the source system.

   ```
   sp_ctrl> activate config filename
   ```

6. On the source system, monitor activation status.

   NOTE: The command retains control of **sp_ctrl** until activation is finished.

7. When activation is complete, switch log files on the source system.

   On-premises database:

   ```
   svrmgr1> alter system switch logfile;
   ```

   Amazon RDS database:

   Run the Amazon RDS procedure **rdsadmin.rdsadmin_util.switch_logfile**.

8. **Do one of the following:**
   - To recover the database to a sequence number, make a note of the highest archive-log sequence number.
   - To recover the database to a Oracle System Change Number (SCN), pick an SCN to recover to on the target database.
9. **On the target system, do one of the following:**
   - If recovering to a sequence number, recover the database from the hot backup using the UNTIL CANCEL option in the RECOVER clause, and cancel the recovery after Oracle has fully applied the log from the previous step.
   - If recovering to a SCN, recover the database from the hot backup using the UNTIL CHANGE *scn* option in the RECOVER clause, and cancel the recovery after Oracle has applied the logs matching the SCN from the previous step.
10. On the target system, open the database with the RESETLOGS option.
11. On the target system, run Database Setup for Oracle on the database. When prompted for the SharePlex database user, enter **n** to choose the *existing* user and password (these were copied in the backup).

    ```
    Would you like to create a new SharePlex user [y]. n
    ```

    NOTES:
    - SharePlex can remain running during the setup process.
    - Database Setup Utilities in the SharePlex Reference Guide.
12. [Optional] If you are using named post queues and are unsure of the queue names, issue the **qstatus** command.

    ```
    sp_ctrl> qstatus
    ```

13. On the target system, issue the **reconcile** command as follows, depending on the recovery option you chose. If you are using named post queues, issue the command for each one.

    - If recovering to a sequence number, substitute the sequence number of the log that you noted previously.

      `sp_ctrl>` **reconcile queue** *queuename* **for** *datasource-datadest* **seq** *sequence_number*

      Example: **reconcile queue SysA for o.oraA-o.oraA seq 1234**

    - If recovering to a SCN, substitute the SCN that you noted previously.

      `sp_ctrl>` **reconcile queue** *queuename* **for** *datasource-datadest* **scn** *scn_number*

      Example: **reconcile queue SysA for o.oraA-o.oraA scn 0123456789**

    NOTE: The command retains control of **sp_ctrl** until the reconcile process is finished.

14. On the target system, run the **cleanup.sql** script to truncate the SharePlex internal tables. Instructions for running this script are in the SharePlex Reference Guide.
15. On the target system, disable triggers on the tables, or run the **sp_add_trigger.sql** utility script so that the triggers ignore the SharePlex user.
16. On the target system, disable check constraints and scheduled jobs that perform DML.
17. [Partitioned replication only] If you are using vertically partitioned or horizontally partitioned replication for any tables, delete the unneeded columns and rows from those tables.
18. [High availability only] On the target (secondary) system, stop **Export**.

    `sp_ctrl>` **stop export**

19. [High availability and peer-to-peer only] On the target (secondary) system, activate the configuration so that SharePlex is ready in the event of failover.

    `sp_ctrl>` **activate config** *filename*

20. On the target system, start the Post process. The two instances are now in synchronization, and SharePlex will continue replicating to maintain synchronization.

    `sp_ctrl>` **start post**

21. [Optional] If this was only a partial backup, drop the tablespaces that were not copied over during the hot backup.

Activation with hot backup: cascading replication

Use this procedure for cascading replication where SharePlex will be posting to a database on the intermediary system.

You will apply the backup to the intermediary system first (represented as “sysB”), and then to the target system (represented as “sysC”).

Perform the following steps to activate with hot backup for cascading replication:

1. On all systems, go to the **bin** sub-directory of the SharePlex product directory, and start **sp_cop** and **sp_ctrl**
2. On all systems, verify that the SharePlex processes are running.

```
sp_ctrl> status
```

3. On the intermediary and target systems, stop the Post process. This allows replicated data to accumulate in the post queue until the databases are recovered.
4.
```
sp_ctrl> stop post
```
5. On the source system, run the Oracle hot backup to the intermediary and target systems.
6. When the backup is finished, activate the configuration on the source system.

```
sp_ctrl> activate config filename
```

7. On the source system, view activation status.

> **NOTE:** The command retains control of **sp_ctrl** until activation is finished.

8. When activation is complete, switch log files on the source system.

   On-premises database:

```
svrmgr1> alter system switch logfile;
```

   Amazon RDS database:

   Run the Amazon RDS procedure **rdsadmin.rdsadmin_util.switch_logfile**.

9. Make a note of the highest archive-log sequence number.
10. On the intermediary system, recover the database from the hot backup using the UNTIL CANCEL option in the RECOVER clause, and cancel the recovery after Oracle has fully applied the log from the previous step.
11. On the intermediary system, open the database with the RESETLOGS option.
12. On the intermediary system, run Database Setup on the database. When prompted for the SharePlex database user, enter **n** to choose the *existing* user and password (these were copied in the backup).

```
Would you like to create a new SharePlex user [y]. n
```

> **NOTES:**
>
> - SharePlex can remain running during the setup process.
> - For more information, see Database Setup Utilities in the SharePlex Reference Guide.

13. [Optional] If you are using named post queues and are unsure of the queue names, issue the **qstatus** command.

```
sp_ctrl> qstatus
```

14. On the intermediary system, issue the **reconcile** command for each post queue. For **seq** *sequence_number*, substitute the sequence number of the log that you noted previously.

```
sp_ctrl> reconcile queue queuename for datasource-datadest seq sequence_number
```

    Example: **reconcile queue SysA for o.oraA-o.oraA seq 1234**

15. On the intermediary system, run the **cleanup.sql** script to truncate all of the SharePlex internal tables. Instructions for running this script are in the SharePlex Reference Guide.

16. On the intermediary system, disable triggers on the tables, or run the **sp_add_trigger.sql** utility script so that the triggers ignore the SharePlex user.
17. On the intermediary system, disable check constraints and scheduled jobs that perform DML.
18. On the intermediary system, set the SP_OCT_REPLICATE_POSTER parameter to 1. This directs SharePlex to capture posted changes on that system and replicate them to the target system.

    ```
    sp_ctrl> set param SP_OCT_REPLICATE_POSTER 1
    ```

19. [Partitioned replication only] If you are using vertically partitioned or horizontally partitioned replication for any tables, delete the unneeded columns and rows from those tables.

    IMPORTANT! Do not start any Post processes yet.

20. On the target system, recover the database from the hot backup using the UNTIL CANCEL option in the RECOVER clause, and cancel the recovery after Oracle has fully applied the log that you reconciled to in the previous steps taken on the intermediary system.
21. On the target system, open the database with the RESETLOGS option.
22. On the target system, run Database Setup on the database. When prompted for the SharePlex database user, enter **n** to choose the *existing* user and password (these were copied in the backup).

    ```
    Would you like to create a new SharePlex user [y]. n
    ```

    NOTE: SharePlex can remain running during the setup process. For more information about Database Setup, see Database Setup Utilities in the SharePlex Reference Guide.

23. On the target system, run the **cleanup.sql** script to truncate the SharePlex internal tables. Instructions for running this script are in the SharePlex Reference Guide.
24. On the target system, disable triggers on the tables, or run the **sp_add_trigger.sql** utility script so that the triggers ignore the SharePlex user.
25. On the target system, disable check constraints and scheduled jobs that perform DML.
26. [Partitioned replication only] If you are using vertically partitioned or horizontally partitioned replication for any tables, delete the unneeded columns and rows from those tables.
27. **On the intermediary system, activate the configuration file.**

    ```
    sp_ctrl> activate config filename
    ```

28. On the intermediary system, monitor activation status.

    NOTE: The command retains control of **sp_ctrl** until activation is finished.

29. On the intermediary and target systems, start the Post process. All instances are now in synchronization, and SharePlex will continue replicating to maintain synchronization.

    ```
    sp_ctrl> start post
    ```

30. [Optional] If this was only a partial backup, drop the tablespaces that were not copied over during the hot backup.

Activate replication with PostgreSQL hot backup on an active database

Use this procedure to use PostgreSQL hot backup to establish a target PostgreSQL instance and activate replication without quieting the source database. This procedure involves using the reconcile command to ensure that transactions which occurred after the point of backup are applied to the target, while eliminating redundant replicated transactions that were already captured by the backup.

Supported databases

PostgreSQL to PostgreSQL

Supported replication strategies

All replication strategies are supported with the following limitations

Limitation applies to	Description
Peer-to-peer	To establish peer-to-peer replication, you must: 1. Quiet all the systems except the trusted source system for the duration of this procedure. 2. Move all users to the trusted source system, and then follow this procedure. Only after this procedure has been performed on all the secondary systems, users may resume activity on them.

Requirements

- Read the requirements for activating a configuration file as mentioned in the user guide.
- Make certain a SharePlex database account exists in the source database (only). This account is usually created when SharePlex is installed for the first time. See the SharePlex Installation and Setup Guide for more information.
- Before you start, review this procedure and see the User Guide for more information about the commands that are used.

Troubleshooting

- If the configuration fails to activate, you can find information about the failure in these places:
- Use the show log command to view the event_log.

- View the activation process log, which is a file named dbname_oconf##.log in the log sub-directory of the SharePlex variable data directory.

Procedure

SharePlex uses the "*Activation with hot backup: all strategies except cascading*" procedure for activation with a hot backup.

Activation with hot backup: all strategies except cascading

Use this procedure for all replication strategies except cascading replication, where SharePlex will be posting to a database on the intermediary system.

Perform the following steps to activate with hot backup for all strategies except cascading:

1. On the source system, navigate to the bin sub-directory and run the '`pg_setup`' utility. On the target system, the database will be in the **stopped** state as backup has not been applied yet; hence, `pg_setup` cannot be executed at this point.
2. On both the systems, navigate to the bin sub-directory of the SharePlex product directory, and start `sp_cop` and `sp_ctrl`.
3. On the source system, activate the configuration.

   `sp_ctrl>` **activate config filename**
4. On the target system, the Post process will be in the **idle** or **stopped due to error** (if `pg_setup` was executed before) state as the database is not running, hence stop the Post process if it is in the **idle** state. This allows replicated data to accumulate in the Post queue until the target database has been recovered and reconciled.

   `sp_ctrl>`**stop post**
5. On the target system, execute the `pg_basebackup` utility, which will take a hot backup from the source system and prepare a data directory for the target PostgreSQL database. Once it is completed, a database can be started on the target system.

Follow the below steps to start the database at the target system:

a. Add an entry for the target system in the `pg_hba.conf` file at the source system.

# TYPE	DATABASE	USER	ADDRESS	METHOD
host	replication	all	`<ip_target>/32`	trust

Following is the description of field used:

ip_target : IP address of the target system where the backup needs to be restored

b. Take a backup of the existing database and create an empty PostgreSQL data directory.
```
mv <pg_data_dir_target> <pg_data_dir_target>_old
mkdir <pg_data_dir_target>
```

c. Execute the `pg_basebackup` utility from the bin directory of PostgreSQL installation at the target system.
```
./pg_basebackup -h <source_ip> -p <source_port> -U <user_id> -X f -D <pg_data_dir_target>
```

Description of the fields used:

- **source_ip :** IP address of the source system where the PostgreSQL database is running
- **source_port :** Port of the PostgreSQL database at the source system
- **user_id :** Database user ID with replication role (SharePlex user has replication role, so it can be used here)
- **pg_data_dir_target :** PostgreSQL data directory path at target system

d. Once the backup is done on the target system, a **backup_manifest** file will be created in `pg_data_dir_target`. Open this file, navigate to the end, and find `End-LSN` under the **WAL-Ranges** tag. Keep note of the `End-LSN` value.

6. On the target system, run the `pg_setup` utility. When prompted for the SharePlex database user, enter **n** to choose the existing user and password (these were copied in the backup).

   - `Would you like to create a new SharePlex user [y]. n`

NOTE: SharePlex can remain running during the setup process.

7. [Optional] If you are using named post queues and are unsure of the queue names, issue the qstatus command.

   `sp_ctrl>`**qstatus**

8. On the target system, issue the reconcile command as follows. If you are using named post queues, issue the command for each one. The LSN number is the one which we have noted down in step 5, which is `End-LSN`.

   `sp_ctrl>`**reconcile queue queuename for datasource-datadest pglsn lsn_number**
   Example:`reconcile queue SysA for r.dbA-r.dbB pglsn 0/C7000100`

NOTE: The command retains control of sp_ctrl until the reconcile process is finished.

9. On the target system, run the `cleanup_pg.sql` script to truncate the SharePlex internal tables if setup is configured as a source during '`pg_setup`'. It is present in bin sub-directory of the SharePlex product directory. To execute this, login to `psql` using the `\i <path_to_script>` command.
10. [High availability and peer-to-peer only] On the target (secondary) system, activate the configuration.

    `sp_ctrl>`**activate config filename**
11. On the target system, start the Post process. The two instances are now in synchronization, and SharePlex will continue replicating to maintain synchronization.

    `sp_ctrl>`**start post**

Activate Replication with an Oracle Hot Backup on a Quiet Database

Use this procedure to use an Oracle hot backup to establish a target Oracle instance and activate replication if user activity can be stopped while the procedure is performed. This procedure can be used for all replication configurations.

Preliminary considerations

Read these points before you proceed.

Supported databases

Oracle source and Oracle target

Supported replication strategies

All but high-availability. This procedure is not appropriate for a high-availability strategy because it requires the source database to be quieted while the backup is taken and the configuration file is being activated.

Certain limitations apply:

Limitation applies to:	Description
Consolidated replication (many sources to one target)	To establish consolidated replication, the use of a hot backup from all source systems is not possible. A backup from one source will override the data that was applied by a backup from a different source. You can use a hot backup of one of the source instances to establish a target instance, and then use another copy method to apply the objects from the other source instances. Possible methods include: • Export/import. For more information, see Activate Replication with Cold Copy/transfer Methods on page 278. • Transportable tablespaces. For more information, see Activate Replication with Oracle Transportable Tablespaces on page 275.

Requirements

- [Unix and Linux systems] Verify that the ORACLE_SID and ORACLE_HOME in the **oratab** file are correct for the instance you will be establishing with the hot backup. The SID must be the SID used in the routing map in the configuration file that you will be activating.
- Read the requirements before you start this procedure. For more information, see Requirements for Activating a Configuration on page 258.

- Users must stop accessing the production database while the hot backup and configuration activation take place.
- Make certain a SharePlex database account exists in the source database (only). This account usually is created during SharePlex installation. See the SharePlex Installation and Setup Guide for more information.
- Before you start, review this procedure and see the SharePlex Reference Guide for more information about the commands that are used.

Procedure

NOTE: If you are not using cascading replication, ignore all references to an *intermediary* system. For more information, see Configure Replication through an Intermediary System on page 207.

1. On the source system, complete the Oracle hot backup.
2. On the source system, stop user access to the source database by shutting it down and opening it in restricted mode.
3. On the source system, switch the redo logs.

   On-premises database:

   ```
   svrmgr1>  alter system switch logfile;
   ```

   Amazon RDS database:

   Run the Amazon RDS procedure **rdsadmin.rdsadmin_util.switch_logfile**.
4. Keep a record of the sequence number of the current log.
5. On all systems, start **sp_cop** and **sp_ctrl** from the **bin** sub-directory of the SharePlex product directory.
6. On all systems, verify that **sp_cop** and **sp_ctrl** are running.

   ```
   sp_ctrl>  status
   ```
7. On the intermediary and target systems, stop Post. Stopping Post allows replicated data to accumulate in the post queue until the databases have been recovered.

   ```
   sp_ctrl>  stop post
   ```
8. On the source system, activate the configuration file.

   `sp_ctrl>` **activate config** *filename*
9. On the source system, view activation status.

   NOTE: The command retains control of **sp_ctrl** until activation is finished.
10. When the activation is finished, allow users to resume access to the source database.
11. List the archive logs on the intermediary and target systems. Delete any logs made after the one for which you made a record.
12. On the intermediary and target systems, recover the database to the log number that you recorded. Make sure a full recovery is performed.
13. On the intermediary and target systems, open the database.

14. On the intermediary and target systems, run the Database Setup utility for the target instance. When prompted for the SharePlex database user, enter **n** to choose the *existing* user and password (these were copied in the backup).

```
Would you like to create a new SharePlex user [y].n
```

NOTES:

- SharePlex can remain running during the setup process.
- For more information about Database Setup, see Database Setup Utilities in the SharePlex Reference Guide.

15. On the intermediary and target systems, run the **cleanup.sql** script to truncate the SharePlex internal tables. Instructions for running this script are in the SharePlex Reference Guide.
16. On the intermediary and target systems, disable triggers on the tables, or run the **sp_add_trigger.sql** utility script so that the triggers ignore the SharePlex user.
17. On the intermediary and target systems, disable check constraints and scheduled jobs that perform DML.
18. [Partitioned replication only] If you are using vertically partitioned or horizontally partitioned replication for any tables, delete the unneeded columns and rows from those tables on the intermediary and target systems.
19. [Intermediary system only] On the intermediary system, set the SP_OCT_REPLICATE_POSTER parameter to 1. This directs SharePlex to capture posted changes on that system and replicate them to the target system.

    ```
    sp_ctrl> set param SP_OCT_REPLICATE_POSTER 1
    ```

20. On the intermediary system, activate the configuration file.

    ```
    sp_ctrl> activate config filename
    ```

21. On the intermediary system, monitor activation status.

    NOTE: The command retains control of **sp_ctrl** until activation is finished.

22. When activation is finished, start the Post process on the intermediary and target systems. All instances are now in synchronization, and SharePlex will continue replicating to maintain synchronization.

    ```
    sp_ctrl> start post
    ```

23. [Optional] If this was only a partial backup, drop the tablespaces that were not copied over during the hot backup.

Activate Replication with Oracle Transportable Tablespaces

Use this procedure to use the Oracle transportable tablespaces feature to establish a target Oracle instance and activate replication. It enables you to synchronize and resynchronize numerous objects quickly and with minimal downtime. It allows you to export just the metadata (data dictionary) and then copy the data files. This method also moves indexes so that there is no need to rebuild them in the target database, and you can move multiple tablespaces at one time.

NOTE: This document does not provide instructions for how to use transportable tablespaces. This procedure should be performed by someone who has a solid understanding of database copy methods.

Preliminary considerations

Read these points before you proceed.

Supported databases

Oracle source and Oracle target

Supported replication strategies

All replication strategies. This procedure may not appropriate for a high-availability strategy if the source database cannot be quieted even briefly.

Requirements

- Read the requirements before you start this procedure. For more information, see Requirements for Activating a Configuration on page 258.
- Make certain a SharePlex database account exists in the source database (only). This account usually is created when SharePlex is first installed. See the SharePlex Installation and Setup Guide for more information.
- Before you start, review this procedure and see the SharePlex Reference Guide for more information about the commands that are used.

Naming conventions used

In this procedure, the "source" system is one of the following:

- The source system of a single-direction replication configuration, including cascading replication.
- All source systems of a consolidated replication configuration.

- The trusted source system in a peer-to-peer replication configuration.
- The primary node of a cluster (where the cluster VIP is running).

In this procedure, the "**intermediary**" system only needs to be part of this procedure if SharePlex will be posting to, and capturing from, an intermediary system in a cascading configuration.

In this procedure, the "target" system is one of the following:

- The target system of a single-direction replication configuration, including cascading and consolidated replication.
- The secondary systems in a peer-to-peer replication configuration.
- The primary node (where the cluster VIP is running) of the target cluster.

In this procedure, the SharePlex commands in the procedure apply to all **sp_cop** instances that apply to the replication strategy you are using (for example, all **sp_cop** processes on a target in consolidated replication).

Procedure

1. On the source system, set the source tablespaces that you want to copy to READ ONLY.

   `svrmgr1>` **alter tablespace** *name* **read only;**

2. On the source system, activate the configuration file.

   `sp_ctrl>` **activate config** *filename*

3. On the source system, start **sp_cop** and **sp_ctrl** from the **bin** sub-directory of the SharePlex product directory.
4. On the source system, verify that **sp_cop** and **sp_ctrl** are running.

   `sp_ctrl>` **status**

5. On the intermediary and target systems, stop Post. Stopping Post allows replicated data to accumulate in the post queue until the databases have been recovered.

   `sp_ctrl>` **stop post**

6. On the source system, export the metadata to an export file.
7. When the export is finished, copy the datafiles to another location on the source system. This minimizes the impact on the source database of copying the files to the target system.
8. Set the source tablespaces back to read/write mode.

   `svrmgr1>` **alter Tablespace** *name* **read write;**

9. If any of the copied datafiles and tablespaces exist in the intermediary or target database, drop them so that the copied files can be applied.
10. Copy the files from the new location on the source system to the intermediary and target systems.
11. On the intermediary and target systems, use the Oracle import utility to import the metadata and the tablespace definitions.
12. On the intermediary and target systems, set the tablespace(s) to read/write mode.
13. On the intermediary and target systems, open the Oracle instances.

14. On the intermediary and target systems, disable triggers on the tables, or run the **sp_add_trigger.sql** utility script so that the triggers ignore the SharePlex user.
15. On the intermediary and target systems, disable check constraints and scheduled jobs that perform DML.
16. [Partitioned replication only] If you are using vertically partitioned or horizontally partitioned replication for any tables, delete the unneeded columns and rows from those tables on the intermediary and target systems.
17. [Intermediary system only] Set the SP_OCT_REPLICATE_POSTER parameter to 1. This directs SharePlex to capture posted changes on that system and replicate them to the target system.

    ```
    sp_ctrl> set param SP_OCT_REPLICATE_POSTER 1
    ```

18. [Intermediary system only] Activate the configuration file.

    ```
    sp_ctrl> activate config filename
    ```

19. [High availability] On the target system, stop the Export process.

    ```
    sp_ctrl> stop export
    ```

20. [High availability and peer-to-peer replication] Activate the configuration on the target system(s).

    ```
    sp_ctrl> activate config filename
    ```

21. Start Post on the intermediary and target systems. SharePlex begins executing the SQL statements that have been collecting in the post queue, keeping the source and target data in sync.

    ```
    sp_ctrl> start post
    ```

22. [Peer-to-peer replication] Allow users to access the databases on all systems.

Activate Replication with Cold Copy/transfer Methods

Use this procedure to synchronize the source and target data with the following utilities:

- Import/Export/Data Pump
- Store/Restore from tape
- FTP

NOTE: This document does not provide instructions for how to perform the chosen copy method. This procedure should be performed by someone who has a solid understanding of database copy methods.

Preliminary considerations

Read these points before you proceed.

Supported databases

Oracle source and Oracle target

Supported replication strategies

All but high-availability. This procedure is not appropriate for a high-availability strategy because it requires the source database to be quieted while the configuration file is being activated.

Requirements

- [Unix and Linux systems] Verify that the ORACLE_SID and ORACLE_HOME in the **oratab** file are correct for the instance you will be establishing with the hot backup. The SID must be the SID used in the routing map in the configuration file that you will be activating.
- Read the requirements before you start this procedure. For more information, see Requirements for Activating a Configuration on page 258.
- Users must stop accessing the production database while the copy and configuration activation take place.
- The target instance must exist.
- Make certain SharePlex database accounts exist in the source and target databases. This account usually is created during installation. See the SharePlex Installation and Setup Guide for more information.
- Before you start, review this procedure and see the SharePlex Reference Guide for more information about the commands that are used.

Naming conventions used

In this procedure, the "source" system is one of the following:

- The source system of a single-direction replication configuration, including cascading replication.
- All source systems of a consolidated replication configuration.
- The trusted source system in a peer-to-peer replication configuration.
- The primary node of a cluster (where the cluster VIP is running).

In this procedure, the "**intermediary**" system only needs to be part of this procedure if SharePlex will be posting to, and capturing from, an intermediary system in a cascading configuration.

In this procedure, the "target" system is one of the following:

- The target system of a single-direction replication configuration, including cascading and consolidated replication.
- The secondary systems in a peer-to-peer replication configuration.
- The primary node (where the cluster VIP is running) of the target cluster.

In this procedure, the SharePlex commands in the procedure apply to all **sp_cop** instances that apply to the replication strategy you are using (for example, all **sp_cop** processes on a target in consolidated replication).

Procedure

1. On the source system, stop user access to the objects that are in the replication configuration.
    - If deploying consolidated replication, you can either stop access to all of the source systems at once and make the copies at the same time, or you can synchronize each source system one at a time using these instructions.
    - If deploying peer-to-peer replication, stop access to all databases in the peer group, including the trusted source.
2. Copy the files from the source system to the intermediary and target systems.
3. On the source system, start **sp_cop** and **sp_ctrl**.
4. On the source system, activate the configuration file (all files if using consolidated replication).

    ```
    sp_ctrl> activate config filename
    ```

5. On the intermediary and target systems, start **sp_cop** and **sp_ctrl**.
6. On the intermediary and target systems, stop Post. Stopping Post allows any data that gets replicated before the target data is established to collect in the post queue.

    ```
    sp_ctrl> stop post
    ```

7. On the source system, allow users to resume access to the source database.
8. On the source system, verify that the **sp_cop**, Capture, and Read processes are running.

    ```
    sp_ctrl> status
    ```

9. Start and mount the intermediary and target databases, but do not allow users access.
10. On the intermediary and target systems, apply the copy to the database.

11. On the intermediary and target systems, disable triggers on the tables, or run the **sp_add_trigger.sql** utility script so that the triggers ignore the SharePlex user.
12. On the intermediary and target systems, disable check constraints and scheduled jobs that perform DML.
13. [Partitioned replication only] If you are using vertically partitioned or horizontally partitioned replication for any tables, delete the unneeded columns and rows from those tables on the intermediary and target systems.
14. [Intermediary system only] Set the SP_OCT_REPLICATE_POSTER parameter to 1. This directs SharePlex to capture posted changes on that system and replicate them to the target system.

    ```
    sp_ctrl> set param SP_OCT_REPLICATE_POSTER 1
    ```

15. [Intermediary system only] Activate the configuration file.

    ```
    sp_ctrl> activate config filename
    ```

16. [Peer-to-peer] Activate the configuration file on the target systems.
17. Start Post on:
    - The intermediary system
    - The trusted source and all other targets in a peer group
    - All other targets

    NOTE: SharePlex will start executing SQL statements that accumulated in the post queue.
18. [Peer-to-peer] On the target systems in the peer group, allow users to resume access to the database.

Activate Replication from Oracle to Open Target

Use this procedure to synchronize an Oracle source database with an Open Target database. SharePlex replicates the Oracle data changes and maintains them in the Post queue until the target is established with the copy. When the target is ready, you run the SharePlex **reconcile** feature, which ensures that Post only applies the operations that occurred after the copy and discards operations that were committed to the source before the copy.

Preliminary considerations

Read these points before you proceed.

Supported databases

Oracle source and any supported target

Supported replication strategies

All

Requirements

- You can use your Oracle RMAN backup system to take a hot backup of your primary instance and recover to an SCN or sequence number in a staging instance.
- This document does not provide instructions for how to perform the chosen copy method. Someone with expertise in database copy methods should perform this procedure. You can use Toad Data point or a third-party tool to extract data from the staging instance into the Open Target database.
- Read the requirements for activating a configuration file. For more information, see Requirements for Activating a Configuration on page 258.
- Before you start, review this procedure and see the SharePlex Reference Guide for more information about the commands that are used.

Procedure

1. On the source and target systems, start **sp_cop** and **sp_ctrl** from the **bin** sub-directory of the SharePlex product directory.
2. On source and target systems, verify that the SharePlex processes are running.

   ```
   sp_ctrl> status
   ```

3. On the target system, stop the Post process. This allows replicated data to accumulate in the post queue until the target database is instantiated and reconciled.

   ```
   sp_ctrl> stop post
   ```

4. Activate the configuration on the source system.

   ```
   sp_ctrl> activate config filename
   ```

5. On the source system, monitor activation status.

   NOTE: The command retains control of **sp_ctrl** until activation is finished.

6. When activation is complete, start the hot backup to the staging instance.
7. When the hot backup is finished, switch log files on the primary source system twice.

   On-premises database:

   ```
   svrmgr1> alter system switch logfile;
   svrmgr1> alter system switch logfile;
   ```

   Amazon RDS database:

   Run the Amazon RDS procedure **rdsadmin.rdsadmin_util.switch_logfile** twice.

8. Copy the archive logs that were generated by the log switch from the primary instance to the staging instance.
9. Do one of the following:
   a. If the source is RAC, recover the database on the staging server to the latest SCN of the last archive log that was copied to the staging server.
   b. If the source is not RAC, recover to the sequence number of the last archive log that was copied to the staging server.

      NOTE: The next steps apply the replicated changes that occurred after the backup point.

10. **Do one of the following:**
    - If the source is RAC, make a note of the SCN that you recovered to on the staging server.
    - If source is non-RAC, make a note of the log sequence number that you recovered to on the staging server.
11. Using the copy method of your choice, make a copy of the Oracle data from the staging server to the Open Target database. Wait until the copy is finished before proceeding to the next step.

12. [Optional] If you are using named post queues and are unsure of the queue names, issue the **qstatus** command and make a note of them.

    ```
    sp_ctrl> qstatus
    ```

13. On the target system, disable triggers on the target tables.
14. On the target system, disable check constraints and scheduled jobs that perform DML.
15. On the target system, run **sp_ctrl**, then issue one of the following **reconcile** commands. If you are using named post queues, issue the command for each one.

    - If the source is non-RAC, reconcile to the log sequence number of the log that you noted previously.

      ```
      sp_ctrl> reconcile queue queuename for datasource-datadest seq sequence_number
      ```

      Example: **reconcile queue SysA for o.prod1-r.rep1 seq 1234**

    - If the source is RAC, reconcile to the SCN that you noted previously.

      ```
      sp_ctrl> reconcile queue queuename for datasource-datadest scn scn_number
      ```

      Example: **reconcile queue SysA for o.prod1-r.rep1 scn 0123456789**

    NOTE: The command retains control of **sp_ctrl** until the reconcile process is finished.

16. On the target system, start the Post process. The two instances are now in synchronization, and SharePlex will continue replicating to maintain synchronization.

    ```
    sp_ctrl> start post
    ```

17

Monitor SharePlex

This chapter contains an overview of the tools that SharePlex provides to detect errors and monitor the replication processes. Like any mission-critical software, SharePlex should be monitored regularly for situations or events that could interfere with processing, especially those that could result in loss of data synchronization.

Contents

View and Terminate SharePlex Processes

These instructions show you how to forcefully terminate SharePlex processes in cases where replication must be shut down immediately.

View and Terminate Processes on Unix and Linux

On Unix and Linux systems, you can use the ps -ef | grep sp_ command to view the SharePlex processes that are running.

- The **sp_cop** process is the root process.
- **The following child processes are spawned by sp_cop on a source system:**
    - Command and Control process (**sp_cnc**)
    - Capture (**sp_ocap**)
    - Read (**sp_ordr**)
    - Export (**sp_xport**)
- **The following child processes are spawned by sp_cop on a target system:**
    1. Command and Control process (**sp_cnc**)
    2. Import (**sp_mport**)
    3. Post (**sp_opst_mt** if the database is Oracle or **sp_xpst** if the database is Open Target)

Each child process has the same **-u***identifier* as its parent **sp_cop** process. This makes it easier to identify related processes when multiple session of **sp_cop** are running.

To terminate a SharePlex process on Unix and Linux:

$ **kill***PID*

Or...

$ **kill -9***PID*

View Events and Errors

SharePlex reports errors and other abnormal conditions in the following ways.

Event Log

SharePlex reports operational errors, notices and warning conditions to the Event Log. This log provides a perpetual step-by-step record of replication activities, errors, and events. The Event Log can help you replay the sequence of events that led up to a problem.

Examples of replication events include:

- Start or stop of **sp_cop** or a replication process
- Execution of a command in **sp_ctrl**. User-issued commands are recorded for every SharePlex command that is issued.

  NOTE: A user-issued command appears in the Event Log as a notice, as in the following example:

  ```
  Notice 08-07-02 16:13:24.641582 23696 1 User command: rjones activate
  config 1route (from mycomp14)
  ```

- Database errors
- Failure of a network connection or SharePlex process
- Start or stop of a utility or script
- Login or logout of a user

Each entry in the Event Log includes:

- The date and time of the event.
- A description of the event and any related messages (error or non-error).
- The event's process ID number, if it is associated with a SharePlex process.

To view the Event Log:

Use the **show log** command in **sp_ctrl** or open the file named **event_log** in the **log** sub-directory of the SharePlex variable-data directory.

SharePlex provides a script for unattended monitoring of this log. For more information, see Monitor events with sp_eventmon on page 295.

NOTE: To control the number of out-of-sync messages that Post logs when a target table is very out-of-sync, use the `SP_OPO_SYNC_LOG_FREQUENCY` parameter. For more information, see the SharePlex Reference Guide.

Status Database

The Status Database contains a summary of the conditions reported in the Event Log, including events that did not generate an error message or warning at the **sp_ctrl** user interface. This information alerts you to potential problems and helps you resolve existing ones. The Status Database may refer you to the Event Log for a more detailed explanation of a warning, notice or event.

To view the Status Database:

Use the **show statusdb** command in **sp_ctrl** or open the file in the **data** sub-directory of the SharePlex variable-data directory.

Error Log

When the Post process detects that source and target tables are out of synchronization, it logs the first 100 SQL statements and data for the out-of-sync transactions to an error file on the target system. You can use this log to determine the extent of the out-of-sync condition, and you can use the SQL statements to repair target tables if the condition is not too severe, after first correcting the cause of the problem.

To view the Error Log:

Open the *ID*_**errlog.sql** file in the **log** sub-directory of the SharePlex variable-data directory (where *ID* is the identifier of the SharePlex target, for example a target database).

Process logs

When a SharePlex process cannot process a record, the process not only logs the record to the Event Log, but also to its process log file. The process logs are primarily for use in debugging.

The name of a process log consists of the *datasource identifier* (such as the ORACLE_SID), the *short name* of the process (such as **ocap**, **ord**, **opo**, **rcl**), the *file number*, and the *file extension* (**.log**).

Examples:

Capture: **ora10_ocap02.log**

Read: **ora10_ord01.log**

Post: **ora10_opo03.log**

Reconcile: **ora10_rcl01.log**

The aging of old log files is performed in a circular pattern. The numbering begins with 01 and ends with 03. Up to three logs can exist at any time, including the current one. When all three logs are full (50 MB), the process starts overwriting them, beginning with the oldest one.

To view a process log

Open the file in the **log** sub-directory of the SharePlex variable-data directory.

Activation log

When you activate a configuration, it generates a log.

To view the activation log

Open the file named ***SID_oconf##*.log** in the **log** sub-directory of the SharePlex variable data directory.

Compare/repair log

The **compare** and **repair** commands log errors, messages and warnings to a log. For more information about these logs, see the **compare** commands in the SharePlex Reference Guide.

Monitor with sp_ctrl Commands

The information commands in sp_ctrl help you monitor different aspects of replication. Issue them frequently to:

- Monitor for out-of-sync tables.
- Verify that replication processes are running.
- View the number of replicated messages in the queues.
- View the Event Log to view warnings, errors and other notifications.
- View process statistics that are helpful for tuning and problem solving.
- Detect tables or operations that are slowing down the replication process.

List of information commands

Command	Auth. level	Description
append status	3	Displays status and results of the **append using** and **append** commands.
copy status	3	Displays status and results of the **copy using** and **copy** commands.
compare status	3	Displays the status and results of the **compare using** and **compare** commands.
lstatus	3	Displays detailed information about the state of SharePlex replication.
job status	3	Displays current status and history for **append**, **compare**, **copy** and **repair** commands.
orainfo	3	Displays the Oracle database information.
qstatus	3	Displays the state of the capture, export and post queues.
repair status	2	Disaplys the status and results of the **repair** and **repair using** commands.
report	3	Displays **append**, **compare**, **copy** and/or **repair** history for a table.
show	3	Displays the source and destination of the data being processed by each replication process on a system, and displays the status of each process.
show capture	3	Displays brief or detailed statistics for the Capture process for use in tuning and problem solving.
show config	3	Displays properties of the active configuration.
show export	3	Displays the number of messages sent to the target system(s).
show import	3	Displays the number of messages received from the source system(s).
show log	3	Displays the Even Log, Command Log, Verify Log, Trace Log, or a process log.

Command	Auth. level	Description
show post	3	Displays brief or detailed statistics for the Post process for use in tuning and problem solving.
show read	3	Displays brief or detailed statistics for the Read process for use in tuning and problem solving.
show sql	3	Displays the current or last SQL statement processed by the Post process.
show statusdb	3	Displays the Status Database, which contains records of important replication events.
show sync	3	Displays information about out-of-sync conditions.
status	3	Displays an overview of the state of SharePlex replication.

See the SharePlex Reference Guide for details about these commads.

Run Monitor Scripts on Unix or Linux

The SharePlex monitoring scripts notify you of events and conditions that can adversely affect replication on Unix or Linux systems. These scripts provide a monitoring mechanism without the need for frequent status checks through **sp_ctrl**. They can be run independently or through scheduled jobs.

SharePlex provides the following scripts:

- **sp_eventmon** monitors the SharePlex Event Log and reports errors that you specify in a special file.
- **sp_logmon** monitors how well Capture is keeping pace with the changes entering the redo logs. If Capture loses pace by a specified number of logs, **sp_logmon** alerts you before the logs wrap so that you can take corrective action. (applicable only for Oracle source)
- **sp_ps monitors** the SharePlex processes and notifies you if one or more are stopped so that you can correct the problem before the logs wrap or the queues exceed available disk space.
- **sp_qstatmon** monitors the status of the SharePlex queues and sends a warning if the backlog exceeds a threshold (limit) that you define. This enables you to take corrective action before the queues exceed available disk space and replication is adversely affected.

IMPORTANT!

- These scripts run on Unix or Linux systems only.
- The monitoring scripts are overwritten with new scripts during patches and upgrades of SharePlex. Before you install the patch or upgrade, rename your existing scripts so that your customizing is retained. After applying the patches, update the new scripts with your customizing. Do not rename the existing scripts to replace the updated scripts, or you could lose important improvements or fixes.

Requirements for using the monitoring scripts

- These scripts must be run in the **ksh** shell.
- All monitoring scripts must remain in the directory where they were installed. All but **sp_ps** are in the **.app-modules** directory of the SharePlex installation directory. The **sp_ps** script is in the **util** directory of the installation directory.
- The scripts must be customized to reflect your environment, such as the type of e-mail or the paging available.
- To use the monitoring scripts, start **sp_cop** with the **-uname** *name* option, where *name* can be an identifier of your choice. Suggestions are:
    - the SharePlex port number
    - the database name of the instance for which replication is being monitored
    - the SharePlex Administrator's name
- SharePlex must be running prior to executing a monitoring script.
- Verify the ORACLE_HOME (the path to the ORACLE_HOME directory) for each Oracle instance being monitored. (only for Oracle source)
- The monitoring scripts make use of **sp_ctrl** commands. Before you use the scripts, make a link in the **util** sub-directory to the **sp_ctrl** binary in the **bin** sub-directory of the SharePlex product directory. Do not copy the binary itself, because that makes it difficult to maintain patches to **sp_ctrl**.

- Users of the monitoring utilities must have the following rights:
    - Local access to **sp_ctrl** and permission to run the script on the system on which the sp_cop to be monitored is running.
    - Korn (ksh) shell access and **ps** permission from the Unix or Linux command line.
    - Read, write and execute permissions to the directory where the scripts reside.
    - The permission on the **iwgrep** utility must be 755.
- The monitoring utilities use the **mailx** program to send e-mail notifications. Before using a script, make sure **mailx** is configured to send e-mail on all systems where the monitoring scripts will be deployed.
- Paging requires that your service provider supports receiving e-mails on your paging device.
- To kill any of the processes generated by these scripts, use the **kill -9** command. The **kill** command alone does not kill all of the processes.

Monitor Oracle capture with sp_logmon

The **sp_logmon** monitoring script helps prevent situations where you must resynchronize your data because the Oracle redo logs wrapped before Capture was finished reading them. It monitors the redo log group to which Oracle currently is writing and determines which log SharePlex is reading.

If Capture loses pace by a specified number of logs, **sp_logmon** generates a warning in the **logmon.log** file and in an e-mail message, if that option is enabled. This gives you time to correct the cause of the delay and restore the archive logs, if necessary.

Prepare to run sp_logmon

Before running the script, perform the following tasks.

Satisfy requirements

See Requirements for using the monitoring scripts on page 291 before you use this script.

NOTE: The script must be run in the **ksh** shell.

Define email addresses

To use the e-mail notification feature, define the e-mail address(es) in the script before running it.

1. Open the script in the **app-modules** directory of the SharePlex product directory.
2. Add any number of address strings after the **MailUserName=** variable. Use the full e-mail and/or pager address. Separate multiple entries with a comma, as shown in the following example:

   ```
   MailUserName=scott@company.com,12345678910@pageservice.com
   ```

IMPORTANT! If the person modifying the script is someone other than a SharePlex user, he or she needs to have these Oracle privileges:

- CONNECT privileges
- SELECT privileges for the V$LOG table
- SELECT privileges for the SharePlex internal tables

Run sp_logmon

Run the script from the **util** sub-directory of the SharePlex product directory, not from **app-modules**. When you run it from the **util** directory, you actually make a soft link that runs a utility which first sets up the correct environment before running the script itself.

Syntax:

nohup sp_logmon -p *port* **-t** *interval* **-l** *integer* [**-m**] [**/dev/null**] **&**

Table 6: Required arguments

Argument	Description
nohup sp_logmon	Directs the script to continue running in the background if the user logs out. This ensures continuous monitoring. The **sp_logmon** component runs the script.
-p *port*	Sets the port number for the instance of **sp_cop** that you are monitoring. You can monitor different SharePlex instances by running **sp_logmon** for each one, using different values for this argument.
-t *interval*	Sets the time interval between scans in seconds. The value can be any positive integer.
-l *integer*	Sets the maximum permissible number of redo logs between where Oracle is writing and where Capture is reading. This value triggers the warning generated by **sp_logmon**. Valid values are positive integers from 1 to the number of redo logs in the group.
&	Runs the script in the background.

Table 7: Optional arguments

Argument	Description
/dev/null	Redirects the notification output to the **/dev/null** device on the local system so that the monitoring process continues to run in the background and generate output. To have the output appear on screen, omit this argument.
-m	Enables the e-mail/paging option. Without this parameter, **sp_logmon** only logs errors to the log file.

Monitor events with sp_eventmon

The **sp_eventmon** monitoring script monitors the SharePlex Event Log (**event_log**) at set intervals for entries relating to key replication events. You can define the scan interval and the error messages you want the script to detect. Each scan starts where the previous one stopped to keep the impact on the system minimal and prevent duplicate warnings.

The sp_eventmon script takes the following actions after each scan of the Event Log:

- When **sp_eventmon** detects an error that you defined, it prints a notification to the **error.splex** log file and an e-mail message, if that option is enabled.
- It logs each error, the error Event Log line number of the error, the **sp_cop** instance name (typically the port number), and the time and date of the error.

The script relies on the **iwgrep** program, the **error_list** file (described later), and a marker file named *username***.mrk** (where *username* is derived from the string that you enter with the **-s** argument when you run **sp_eventmon**). These three components must be kept in the same directory as the script, or it will not function.

NOTE: The *username***.mrk** file prevents duplicate warning messages from being sent to the log and to your e-mail or pager. Without this file, the script starts scanning the Event Log from the beginning every time it starts. Warnings that were previously generated are sent again.

Prepare to run sp_eventmon

Before running the script, perform the following tasks.

Satisfy requirements

See Requirements for using the monitoring scripts on page 291 before using this script.

NOTE: The script must be run in the **ksh** shell.

Define error messages

The **sp_eventmon** script scans for events listed in the **error_list** file, located in the **util** sub-directory of the SharePlex product directory. View that file for more information about the supported errors. You can add custom error strings to the **error_list** file by editing it in any ASCII text editor. Open the file and place each error string on a separate line.

Set IW_HOME

The IW_HOME variable in the script must be set to the correct value on each machine. This variable must point to the directory in which the monitoring scripts and **iwgrep** reside.

If the path is not correct:

1. Open the script in the **app-modules** directory of the SharePlex product directory.
2. Set the path as shown in the following example:

```
IW_HOME=/export/home/splex/monscripts
```

Define e-mail addresses

To use the e-mail notification feature, define the e-mail address(es) in the script before running it.

1. Open the script in the **app-modules** directory of the SharePlex product directory.
2. Add any number of address strings after the **MailUserName=** variable. Use the full e-mail and/or pager address. Separate multiple entries with a comma, as shown in the following example:

   ```
   MailUserName=scott@company.com,12345678910@pageservice.com
   ```

Run sp_eventmon

NOTES:

- If you are running multiple instances of **sp_eventmon**, each instance must be run under the name of a different operating system user. Each *username*.**mrk** file will have a different *username*.
- Use the **truncate log** command in **sp_ctrl** to truncate the Event Log frequently when you are running the **sp_eventmon** script. If the log grows too large, the **iwgrep** program cannot grep from it properly. When you issue the **truncate log** command, remove the *username*.**mrk** file. The next time you run **sp_eventmon** it will create a new file. See the SharePlex Reference Guide for more information about the **truncate log** command.
- When there is an existing Event Log with errors in it and the script is running, issue the **truncate log** command and then delete the *sp_cop_name*.**mrk** file, where *sp_cop_name* is the value used in the **-s** argument when the script was run. This file is in the **util** sub-directory of the SharePlex product directory.

To run sp_eventmon

Run the script from the **util** sub-directory of the SharePlex product directory, not from **app-modules**. When you run it from the **util** directory, you actually make a soft link that runs a utility which first sets up the correct environment before running the script itself.

Syntax:

nohup sp_eventmon -s '*sp_copname*' **-t** *interval* **-p** *path* [**-n** *name*] [**-m**] **/dev/null &**

Table 8: Required arguments

Component	Description
nohup sp_eventmon	Directs the script to continue running in the background if the user logs out. This ensures continuous monitoring. The **sp_eventmon** component runs the script.
-s 'sp_*copname*'	Sets the name of **sp_cop** that was used when **sp_cop** was started with the **-u** option. The name of **sp_cop** must be enclosed within single quote marks. You can use this parameter more than once to monitor multiple **sp_cop** instances on a system. Without this parameter, **sp_eventmon** will not start.
&	Runs the script in the background.
-t *interval*	Sets the time interval between scans in seconds. The value can be any positive integer.

Table 9: Optional Components

Component	Description
-p *path*	Sets the path to the SharePlex variable-data directory. Without this variable, **sp_eventmon** assumes the default path.
/dev/null	Redirects the notification output to the **/dev/null** device on the local system so that the monitoring process continues to run in the background and generate output. To have the output appear on screen, omit this argument.
-n *name*	Sets the name of the Event Log if it is something other than the default name "event_log."
-m	Enables the e-mail/paging option. Without this option, **sp_eventmon** only logs errors to the log file.

Monitor processes with sp_ps

The **sp_ps** monitoring utility monitors all SharePlex processes, including child processes, associated with a specified **sp_cop** instance. It scans the processes at regular intervals and reports abnormal conditions to one or more log files. It can monitor multiple installations of SharePlex on one or more systems, and it supports uni-directional and bi-directional (peer-to-peer) configurations.

Prepare to run sp_ps

Before running the script, perform the following tasks.

Satisfy requirements

See Requirements for using the monitoring scripts on page 291 before using this script.

NOTE: The script must be run in the **ksh** shell.

Set the scan interval

The scan interval specifies how long the **sp_ps** program waits between checks. The default is 2,000 seconds. To specify a different scan interval, follow these steps.

1. Open the **sp_ps** file in the **app-modules** directory of the SharePlex product directory.
2. Set the `interval=` parameter to the required scan interval. Use any positive integer, for example:

   ```
   interval=1500
   ```

Define email addresses

To use the e-mail notification feature, define the e-mail address(es) in the script before running it.

1. Open the script in the **util** directory of the SharePlex product directory.
2. Add any number of address strings after the **MailUserName=** variable. Use the full e-mail and/or pager address. Separate multiple entries with a comma, as shown in the following example:

   ```
   MailUserName=scott@company.com,12345678910@pageservice.com
   ```

NOTE: The e-mail/paging option is enabled by default for **sp_ps**, but confirm that it was not changed. In the script, `MAILOPTION=TRUE` enables e-mail notifications and `MAILOPTION=FALSE` disables them.

Run sp_ps

Run the script from the **util** sub-directory of the SharePlex product directory.

Syntax:

nohup sp_ps ['**sp_cop -u** *name*'] *CONFIGURATION* [**> /dev/null**] [**&**]

Table 10: Required arguments

Argument	Description
nohup sp_ps	Directs the script to continue running in the background if the user logs out. This ensures continuous monitoring. The **sp_ps** component runs the script.
'sp_cop -u *name'*	Use this parameter if you are running more than one **sp_cop** process. Use it to specify each one of those processes that you want to monitor. This argument must reflect exactly the same name that was used when **sp_cop** was started with the **-u** option. It must be enclosed within single quote marks.Without the **-u***name* option, **sp_ps** assumes you want to monitor the **sp_cop** that uses the default SharePlex port of 2100.
CONFIGURATION	Specifies the type of configuration of the SharePlex instance being monitored. This value must be entered in CAPITAL letters. Valid values: `SOURCE` — Use for uni-directional replication to monitor the Capture, Read and Export processes on the source system. `TARGET` — Use for uni-directional replication to monitor the Import and Post processes on the target system. `MULTI-SOURCE` — Use for peer-to-peer replication. It directs the script to monitor the Capture, Read, Export, Import and Post processes on each system. **NOTE:** If replicating between source and target tables on the same system, there are no Export or Import processes.
> /dev/null	Redirects the notification output to the **/dev/null** device on the local system so that the monitoring process can continue to run in the background and generate output. To have the output appear on screen, omit this argument
&	(Ampersand) Runs the script in the background.

Monitor queues with sp_qstatmon

The **sp_qstatmon** script monitors the status of the capture and post queues for message backlogs. You can configure the script to alert you if the number of messages in a queue exceeds a defined threshold (limit), indicating that there is a potential data, system or network problem. This gives you time to correct the problem before the queues exceed their allocated space on the filesystem.

After each analysis of the queues, the **sp_qstatmon** script prints a notice in the **capstat.log** file for the capture queue or the **poststat.log** file for the post queue, as well as an e-mail message if that option is enabled.

Prepare to run sp_qstatmon

Before running the script, perform the following tasks.

Satisfy requirements

See Requirements for using the monitoring scripts on page 291 before using this script.

NOTE: The script must be run in the **ksh** shell.

Assign permission to create temporary files

The script creates some temporary files in the **util** sub-directory of the SharePlex product directory. Assign write permission to that directory to the **sp_qstatmon** module.

Define email addresses

To execute **sp_qstatmon** with e-mail notification, you must first must define the e-mail address(es) in the script. Notification messages are sent to all addresses coded in the script. Unless email notification is enabled, **sp_qstatmon** only logs errors to the log file.

You can specify as many addresses as you want using the following procedure:

1. Open the **sp_qstatmon** script in any ASCII text editor. The script is in the **.app-modules** directory in the SharePlex installation directory.
2. Add the address strings after the **MailUserName=** variable. Use the full e-mail and/or pager address. Separate multiple entries with a comma, as shown in the following example.

   ```
   scott@company.com, 12345678910@pageservice.com
   ```

3. Save and close the file.

Run sp_qstatmon

Run the script from the **util** sub-directory of the SharePlex product directory, not from **app-modules**. When you run it from the **util** directory, you actually make a soft link that runs a utility which first sets up the correct environment before running the script itself.

Syntax:

nohup sp_qstatmon -v *path* **-t** *n* **-p** *port_number* [**-c** *integer*] [**-d** *integer*] [**-m**] > **/dev/null &**

Table 11: Required arguments

Argument	Description
nohup sp_qstatmon	Directs the script to continue running in the background if the user logs out. This ensures continuous monitoring. The **sp_qstatmon** component runs the script.
-v *path*	Sets the path to the SharePlex variable data directory for the instance of **sp_cop** that you want to monitor. Without this variable, **sp_qstatmon** fails and prints an error message requesting a valid path.
-t *n*	Sets the time interval between scans in seconds. This value can be any positive integer.
-p *port*	Sets the port number for the instance of **sp_cop** that you are monitoring. You can monitor different SharePlex instances by running **sp_qstatmon** for each one, using different values for this argument.
&	Runs the script in the background.

Table 12: Optional arguments

Argument	Description
/dev/null	Redirects the notification output to the **/dev/null** device on the local system so that the monitoring process continues to run in the background and generate output. To have the output appear on screen, omit this argument.
-c *integer*	Sets the number of messages in the capture queue at which the script issues a warning message. This value can be any positive integer. Without this parameter, **sp_qstatmon** defaults to 100 messages.
-d *integer*	Sets the number of messages in the post queue at which the script issues a warning message. This value can be any positive integer. Without this parameter, **sp_qstatmon** defaults to 100 messages.
-m	Enables the e-mail/paging option. Without this parameter, **sp_qstatmon** only logs errors to the log file.

Monitor Replication with SNMP

SharePlex provides agent support for Simple Network Management Protocol (SNMP) on all Unix and Linux platforms supported by SharePlex replication.

NOTE: SharePlex provides only **agent** support for SNMP. It only sends SNMP traps. SharePlex *does not* provide an SNMP signal daemon (SNMP manager) to intercept the traps. Use the SharePlex SNMP feature only if you have a Network Management Station (NMS) to manage SNMP signals. The SharePlex SNMP agent is named **snmptrap** and is installed with SharePlex in the **bin** sub-directory of the SharePlex product directory. Do not run this program.

Enable SNMP

To enable SNMP monitoring of SharePlex replication, set the SP_SLG_SNMP_ACTIVE parameter to 1. By default, the parameter is set to 0 (disabled).

Configure the SNMP agent

The following parameters configure the SNMP agent to communicate with the NMS. Each parameter must have a value if the SP_SLG_SNMP_ACTIVE parameter is enabled.

Parameter	Value
SP_SLG_SNMP_HOST	The name of the system (host) to which the traps will be sent
SP_SLG_SNMP_COMMUNITY	The community security string
SP_SLG_SNMP_MJR_ERRNUM	The major error number to be used by the traps
SP_SLG_SNMP_MNR_ERRNUM	The minor error number to be used by the traps

Custom MIB parameters

The following parameters specify required information for a custom MIB.

Parameter	Value
SP_SLG_SNMP_ENTERPRISE_OID	The enterprise object identifier to send with the trap. The default is 1.3.6.1.4.1.3.1.1 .
SP_SLG_SNMP_TRAP_OID	A custom object identifier to bind to the trap. The default is 1.3.6.1.2.1.1.1.0.
SP_SLG_SNMP_TRAP_PROGRAM	The name of the trap program. The default is iwsnmptrap.

Configure the SNMP traps

The following parameters configure the SNMP agent to send traps for specific replication events. The message or error text for the event is included in the trap and is the same error that appears in the Event Log.

To enable an SNMP trap for an event, set the corresponding parameter to a value of 1. By default all traps are disabled (parameter value of 0).

Parameter	SharePlex Event
SP_SLG_SNMP_INT_ERROR	SharePlex logic errors and errors that cause processes to exit
SP_SLG_SNMP_SYS_ERROR	System-related errors encountered by SharePlex
SP_SLG_SNMP_ERROR	Other SharePlex errors
SP_SLG_SNMP_OUT_OF_SYNC	Replication is out of synchronization
SP_SLG_SNMP_STARTUP	SharePlex starts up
SP_SLG_SNMP_SHUTDOWN	SharePlex shuts down
SP_SLG_SNMP_LAUNCH	A SharePlex process starts
SP_SLG_SNMP_EXIT	A SharePlex process stops

18

Prevent and Solve Replication Problems

This section contains solutions to many of the common questions and problems that can arise during replication, and it also suggests preventive measures for avoiding problems.

Contents

Find the Solution in the SharePlex Knowledge Base

If the issue you are experiencing is not listed in this documentation, search the SharePlex Knowledge Base at:

https://support.quest.com.

The Knowledge Base provides filtering options and links to other resources that can help you use and troubleshoot SharePlex.

Solve database setup problems for Oracle

This section helps you diagnose problems that are associated with the SharePlex database account and connection information that was created with the Database Setup utility when SharePlex was installed on the system.

NOTE: For more information about Database Setup, see Database Setup Utilities in the SharePlex Reference Guide.

If the issue you are experiencing is not listed in this documentation, search the SharePlex Knowledge Base at:

https://support.quest.com.

The Knowledge Base provides filtering options and links to other resources that can help you use and troubleshoot SharePlex.

Oracle setup issues

Problem	Description	Solution
Incorrect ORACLE_SID and/or ORACLE_HOME	If SharePlex cannot interact with Oracle, it might be using the wrong ORACLE_SID and/or ORACLE_HOME.	1. See How to find the ORACLE_SID and ORACLE_HOME on page 352 to determine the Oracle values. 2. Rerun the Database Setup utility. For more information, see Database Setup Utilities in the SharePlex Installation and Setup Guide.
Insufficient database privileges	If the Database Setup utility fails, the person who runs it may not have the correct privileges	For more information, see Database Setup Utilities in the SharePlex Installation and Setup Guide.
Asterisk as the ORACLE_SID entry	Sometimes, the **oratab** file has an * (asterisk) symbol instead of a value for the ORACLE_SID.	Ensure that a valid ORACLE_SID is in the **oratab** file, and then try running the database setup again.
More than one **oratab** file (Sun Solaris)	On Solaris systems, the **oratab** file is typically located in the **/var/opt/oracle** directory, but because other platforms store the **oratab** file in the **/etc** directory, there could be a second **oratab** in the **/etc** directory.	Either move, rename or delete the secondary **oratab** file, and then try running the database setup again.
Oracle not running	Oracle must be running and the instance must be open while you run the Database Setup utility. The utility accesses Oracle to establish SharePlex as a user and install its internal tables.	Start Oracle and open the instance.

Problem	Description	Solution
sp_cop is running	The SharePlex **sp_cop** process cannot be running while you are running the Database Setup utility.	If it is running, shut it down using the **shutdown** command in **sp_ctrl.** Run **sp_ctrl** from the bin sub-directory in the SharePlex product directory.
Oracle library location not correct	On Unix and Linux systems, SharePlex expects the Oracle library to be in the **$ORACLE_ HOME/lib or $ORACLE_HOME/lib32** directory. In some environments, the Oracle library has a different name than what SharePlex expects it to be, or it is installed in a different location than expected (or both). In that case, you will see an error message when you attempt to run the Database Setup utility.	Install the appropriate library from Oracle and then re-start SharePlex (if it is stopped). SharePlex will link to the correct library from that point forward.
ld.so.1: sqlplus: fatal: libsunmath.so.1: can't open file: errno=2" error	On Unix and Linux systems, this error indicates that SharePlex cannot find the **libsunmath** and **libshareplex** libraries, even though the link exists in the proper place.	**You can use either of these solutions:** • Create a softlink for **$ORACLE_ HOME/lib/libsunmath.so.1** in the **/usr/lib** directory. or... • In the **ECXpert/config/bdg.ini** file in the [**DB_ENV**] section add the following line: LD_LIBRARYPATH=*full oracle home path*/lib
Wrong user-id	To run Database Setup on Unix and Linux systems, the **set-user-id** for the Oracle software need to be **-rwsr-s--x.** Those permissions allow non-Oracle users to log into SQL*Plus.	Set the correct values for **set-user-id**.

Solve Configuration File Problems

This section reviews problems and solutions associated with the management of configuration files. See also Solve Activation Problems on page 308.

If the issue you are experiencing is not listed in this documentation, search the SharePlex Knowledge Base at:

https://support.quest.com.

The Knowledge Base provides filtering options and links to other resources that can help you use and troubleshoot SharePlex.

A configuration file was accidentally deleted

You might be able to recover an accidentally deleted configuration file if that configuration was previously active and you did not run **db_cleansp** since it was activated.

Solution: View the Event Log to determine the activation ID for that configuration file, then look in the **save** sub-directory of the SharePlex variable-data directory for a .**conf**.*actid* file, where **actid** is the activation ID you got from the Event Log.

Solve configuration file errors

Error message	Description	Solution
`The parameter for 'create config' must be a new file name. or... Destination file exists - file must not exist prior to operation.`	The name you gave this configuration already exists for another configuration file.	Use a different name. To see a list of configurations on a system, use the **list config** command.
`Couldn't fork editor. or… Editor execution failed.`	SharePlex failed to open the default text editor.	Make sure the editor still exists on the system. The default editors is vi on Unix and Linux . To change the default text editor, see Set a Default Editor for sp_ctrl on page 55.
`Destination file exists - file must not exist prior to operation.`	The name you specified when copying the configuration file already exists in this SharePlex instance.	Use a different name. To see a list of configurations on a system, use the **list config** command.
`Problems in reading or writing file used in edit -- command aborted.`	The **edit config** command could not open the specified file.	Confirm the name of the file you are trying to edit, including the case, and check to see if the file is corrupted.
`Destination file exists - file must not exist prior to operation.`	The new name you are giving this configuration already exists for another file.	Use a different name. To see a list of configurations on a system, use the **list config** command.
`Invalid file name passed to command. or… File does not exist.`	You could have misspelled the configuration name or used the wrong case.	Verify the name and spelling, including the case, then enter it again. To see a list of configurations on a system, use the **list config** command.
`File access denied - check file permissions.`	You are not authorized to issue the command.	View your authorization level with the **authlevel** command in **sp_ctrl**, then determine its minimum authorization level. The SharePlex Administrator assigns authorization levels. For more information, see About the SharePlex Security Groups on page 252.

Solve Activation Problems

This section reviews problems and solutions that may be associated with the activation of a configuration file.

If the issue you are experiencing is not listed in this documentation, search the SharePlex Knowledge Base at:

https://support.quest.com.

The Knowledge Base provides filtering options and links to other resources that can help you use and troubleshoot SharePlex.

NOTE: Many configuration problems can be prevented by using the **verify config** command in **sp_ctrl** before you activate a configuration or reactivate one containing new or changed objects. The **verify config** command reviews the configuration to help ensure that basic requirements for activation and replication have been satisfied. See the SharePlex Reference Guide for more information.

SharePlex cannot locate the configuration file

If SharePlex cannot locate the configuration file that you want to activate, verify its location. Configuration files must reside in the **config** sub-directory of the SharePlex variable-data directory for activation to succeed. The **create config** command in **sp_ctrl** automatically puts configuration files in that directory. It is possible someone moved it.

Problems also occur if the configuration file was created without using the **create config** command. This can happen if the file was created directly through the operating system or if the **config.sql** or **build_config.sql** script was used to create it. If the working directory at the time was not the **config** sub-directory, or if the file was saved to a different directory, the activation will not find it.

Solution: Move the configuration file to the **config** sub-directory of the SharePlex variable-data directory.

Some objects failed to activate

If SharePlex cannot activate one or more objects listed in the configuration file, it will continue to activate the other ones and display the names of the ones that failed in the ***ID_oconf##*.log** file.

The following are reasons why individual objects fail to activate.

Problem	Description	Solution
Invalid objects	You may be trying to replicate objects that are not supported by SharePlex.	To understand the objects and operations that SharePlex supports, see the SharePlex Release Notes that accompany this release.
Invalid target systems	SharePlex could not get routing information.	Verify the names of the targets, and fix any syntax errors in the routing map. For some databases you specify an instance name, while for others you specify the actual database name. For routing instructions, see: • Database Specifications in a Configuration File on page 67

Problem	Description	Solution
		• Routing Specifications in a Configuration File on page 69 After your fix the routing syntax, activate the affected configuration again.
Syntax errors and misspelled words	Misspelled names, object names specified without an owner name, and other improper syntax in the configuration specification can cause activation of an object to fail.	Run the **verify config** command to view the errors. For help with configuration syntax, see Configure SharePlex to Replicate Data on page 60.
(Oracle) SharePlex cannot lock tables	If activation cannot lock the tables in the configuration file, activation of that table will fail.	Assign the SharePlex database user the privilege to lock tables.

Activation failed completely

There are several things can cause the entire activation of a configuration to fail. Common error messages for configuration activation problems are:

```
Bad configuration file

The Oracle sid SID specified in the config file is invalid.
```

The following are causes and solutions for activation failures.

Problem	Description	Solution
Password problems	If SharePlex is having trouble connecting to the source database and you know the SharePlex account exists, find out if someone changed the password.	Update the connection information. For more information, see Change the SharePlex Database Account on page 398.
Deactivation followed too closely by activation	In rare cases, if you activate a configuration too soon after a deactivation, the activation fails.	Before you activate a new configuration after deactivating one, wait until you see the following message in the Event Log: `Notice: sp_ordr (for o.ora10 queue o.ora10) Deactivated.`
No datasource specification	The datasource specification is wrong or incomplete. An error similar to the following is returned: `The` ***datasource*** `specified in the config file is invalid.`	Specify as: **Datasource:o.***SID*
(Oracle) Incorrect ORACLE_SID	The wrong ORACLE_SID is specified on the **Datasource**:**o**.SID line.	Edit the configuration file to specify the correct ORACLE_SID. How to find the ORACLE_SID and ORACLE_HOME on page 352.
(Oracle)Insufficient	If Oracle Error 20 (`ORA-00020 maximum`	Increase the PROCESSES parameter in

Problem	Description	Solution
PROCESSES setting	`number of processes (string) exceeded`) is the cause of an activation failure, it is because Oracle ran out of resources on the source system to allow one or more threads to log on.	Oracle or decrease the number of activation threads that you are using. The number of threads is controlled by the **threads** option of the **activate config** command.
(Oracle) ORA-00942: table or view does not exist.	SharePlex cannot access the Data Dictionary.	Make certain that the O7_ DICTIONARY_ACCESSIBILITY parameter in the **init.ora** file is set to **TRUE**. (This is the default.) The database must be restarted if the parameter is changed.

The reconcile command is slow to complete

This applies to activation and reconcile of an Oracle source. If you issue the **reconcile** command when source database activity is low, the command process can, in some circumstances, seem to stall. This happens because the **reconcile** command is dependent on data continuing to arrive from the source system. If there is no replicated activity on the source system after the point of the hot backup or copy, the reconcile process waits until source activity resumes. This is normal.

Common activation errors

The following are common error messages that you might encounter when activating a configuration.

ACTIVATE CONFIG error messages	Cause	Solution
`line n, source object name (T_HFL_1) not of form OWNER.TABLE`	The owner name may be missing from one or more objects listed in the configuration file.	Specify object names as *ownername.objectname*.
`syntax error in line n.`	There is a syntax error on the specified line of the configuration file.	Fix the incorrect syntax. For more information, see Configure SharePlex to Replicate Data on page 60.
`line n, bad routing spec (o.ora10)`	There is a syntax error in the routing map.	Make sure the routing map is written correctly. For more information, see Routing Specifications in a Configuration File on page 69.
`File does not exist.`	SharePlex cannot find the configuration file.	Issue the **list config** command. If the file you want to activate is not listed, it may not be in the **config** sub-directory of the variable-data directory. Find the file and move it to that directory, then issue the **activate config** command again.
`Attempt to run sp_conf when sp_conf is already active`	The configuration is already in the process of being activated.	None required.
`Login parameters not set...`	A SharePlex account and	Run the Database Setup utility. For

ACTIVATE CONFIG error messages	Cause	Solution
	internal tables do not exist in the source database.	more information, see Database Setup Utilities in the SharePlex Reference Guide.
`WARNING, not all objects activated successfully. Check activation log.`	One or more tables failed to activate.	For more information, see Some objects failed to activate on page 308.
`Deactivate/flush a nonactive datasource`	You are attempting to deactivate a configuration that is not active.	No action is necessary if this is the configuration that you wanted to deactivate. To see a list of configurations, use the **list config** command.
`(Oracle) Currently involved in transaction.`	Objects in the configuration file are locked.	SharePlex cannot lock Oracle tables to analyze them if another process has them locked. If the locks are from source transaction activity, try activating at a time when the database is less busy.

Solve Replication Problems

This section reviews common problems that you could encounter while data is being replicated.

If the issue you are experiencing is not listed in this documentation, search the SharePlex Knowledge Base at:

https://support.quest.com.

The Knowledge Base provides filtering options and links to other resources that can help you use and troubleshoot SharePlex.

General issues

Problem	Description	Solution
Network problems	Network factors can cause data to transfer to a target system too slowly, or not at all.	Make certain that all Export and Import processes are running, and ask the network administrator to check the network for causes of slowdowns.
Parameter settings	Some SharePlex parameters, when changed from their default settings, can inhibit replication performance.	To determine if any parameter settings are responsible for reduced performance, issue the **list param modified** command to view parameters that have non-default settings. Review the documentation for those parameters in the SharePlex Reference Guide to determine whether a particular setting is affecting replication speed.

Oracle Capture-related issues

If Capture is stopped, issue the **status** command in **sp_ctrl** to verify whether or not it was stopped intentionally by an authorized SharePlex user. If it was, find out why and make certain that it is not stopped for too long.

The following are other reasons why Capture could slow down or stop unexpectedly.

Problem	Description	Solution
Excessive chaining	Excessive chaining reduces the performance of the source database because Oracle must read multiple blocks for a single row. Excessive chaining reduces Post performance on the target system because block fragmentation in the database decreases the speed at which Post can apply SQL statements.	Reorganize chained tables.
Ineffective redo log configuration	Capture reads archive logs if the redo logs wrap before Capture is finished with them, but this can slow down replication and consume disk space. In most cases, the redo logs should be configured so that SharePlex	For more information about how to set up the logs, see the SharePlex Installation and Setup Guide.

Problem	Description	Solution
	can avoid using the archive logs.	
SharePlex has low process priority	If Capture, Read, or both slow down during hot backups or other Oracle-intensive operations, view the process-priority settings on the system to determine if SharePlex can be assigned more resources.	Ideally, the SharePlex priority should match Oracle's priority.
Keys are not being logged	If PK/UK logging is not enabled for the tables in replication, SharePlex must query the database to get the key values for update and delete operations.	Enable PK/UK logging.
Temporary tables and FND tables	Temp tables receive numerous changes that cause replication overhead and performance degradation.	Remove temporary tables from replication.
archive logs not available	When Capture detects a log wrap and the archive logs are not available, Capture stops and returns a `"Log wrap detected"` error. It waits a certain amount of time and then starts again, continuing this process until the logs are restored.	Restore the archive logs from the one that Capture last processed or point SharePlex to their location with the SP_OCT_ARCH_LOC parameter. To determine the log that Capture needs, use the **show capture** command with the **detail** option. For more information, see How to Restore Oracle Archive Logs on page 347.
Compressed archive logs	Capture cannot read compressed archive logs.	Uncompress the current log that Capture needs and all those after it so that Capture can process them. To determine the log that Capture needs, use the **show capture** command with the **detail** option.
archive logs in unexpected location	Capture stops if it cannot locate the archive logs it needs. When the redo logs wrap, Capture looks for the archives in Oracle's archive-log list, and if the logs are not there, Capture looks in the location specified by the SP_OCT_ARCH_LOC parameter.	If you are storing the archive logs in a location other than the normal Oracle location, make sure this parameter is set to the full path name of the directory containing the archive logs.
No access to the redo logs	Capture stops when it cannot read or find the logs, then it tries to read the log again.	Find out if someone changed permissions such that SharePlex does not have permission to read the redo logs.
No access to the variable-data directory	Capture stops if it is unable to write to the SharePlex logs because of restrictive permissions on the **state** sub-directory in the variable-data directory, or if there are space restrictions in that directory.	Verify the permissions and space for the variable-data directory.

Oracle Post-related issues

There are a number of things that cause the Post process to slow down. SharePlex generates the `operation taking too long` message when Post exceeds the internally controlled allowable wait time to apply a SQL statement to the target instance. Often it only takes one table to cause a bottleneck. Use the **show sql** or **show post** command to find out which table Post is processing, and then check for the following.

Problem	Description	Solution
Full table scans	If a target table does not have a key, Post must perform a full table scan to find the correct row. This slows the Post process.	**Do one of the following:** • Add an index or key if possible, and use a hints file. See For more information, see Use Oracle INDEX Hints on page 360. • If you cannot add a key, create a key definition.For more information, see Define a Unique Key on page 92. • Set the Oracle DB_FILE_MULTIBLOCK_READ_COUNT parameter to the maximum number of blocks that can be read in one I/O request. This is defined by the system setting MAX_IO_SIZE/DB_BLOCK_SIZE. You can increase the DB_BLOCK_BUFFERS parameter as well.
Bitmap indexes on target tables	Bitmap indexes are beneficial for queries, but they slow down the DML operations applied by Post.	Avoid using bitmap indexes on target tables in replication, or only use them on tables that do not have frequent DML activity.
Disk I/O bottleneck	Disk I/O bottlenecks on the target system are a common cause of slow Post performance. Post can spend a high percentage of its time waiting for Oracle to commit data. The effect is worse when there is a Capture process reading from the same log device in peer-to-peer replication.	Disk I/O is the nature of the database environment, but you can reduce the bottleneck by placing the redo logs on faster hard drives or on a solid-state drive.
High number of buffer gets	Review any table with SQL statements that cause a high number of buffer gets. There should only be two to four buffer gets on an index, depending on its size and whether or not it is a unique index.	If there are more buffer gets than four, the index probably needs to be rebuilt. You can rebuild an index without the need to reactivate the configuration file.
Oracle write-rate bottlenecks	By default, one buffer writer writes all dirty Oracle blocks to disk. Whenever that buffer writer process wakes up to	If this is a constant problem, you can consider increasing the number of writers. See the database documentation.

Problem	Description	Solution
	write, it locks portions of shared memory and, in effect, blocks the processes that are either modifying or reading data blocks — including the Post process.	
Small transaction size	Normally, Post performs an internal read/release after it receives each COMMIT, which means it purges that data from the queue as part of the checkpoint recovery process. For smaller transactions, this can cause excessive I/O on the target system and hinder the Post process.	If most of your transactions are small, try changing the value of the Post SP_OPO_READRELEASE_INTERVAL parameter. See the SharePlex Reference Guide for more information about this parameter.
Sequences not cached	If the sequences in replication are not cached, they add unnecessarily to the replication volume.	Cache sequences. If replicated sequences are part of a key, replicate the tables that contain those keys, and remove the sequences from the replication configuration. You should see significant performance improvement.
Low ulimit	An error similar to the following means that the system file descriptors setting on the target system needs to be increased: `Error 07-24-08 12:11:40.360226 8693 12345 Poster error: /var/quest/vardir/log/event_log: Too many open files (posting from ora102, queue prodsys, to ora10b)`	**Solution**: Set the ulimit as close to the optimal value of 1024 as possible. The ulimit can be set either as a system *hard limit* or a session-based *soft limit*, as follows: • Set a hard limit: (Recommended) A root user and system restart are required to change the hard limit, but the value remains fixed at the correct level to support SharePlex. Consult your System Administrator for assistance. • Set a soft limit: A soft limit setting stays in effect only for the duration of the **sp_cop** session for which it was set, and then it reverts back to a default value that may be lower than the hard limit and too low for SharePlex.
Compare process locks	The **repair** command locks all rows that need repair, or the entire table if there are more than 1000 out-of-sync rows, throughout the repair process. This could block Post if Post tries to apply data to the table being repaired.	If you do not want Post to wait for a repair process to finish, you can kill the compare process. To avoid issues with locks caused by repair processes, consider running the repairs during non-peak hours.

Problem	Description	Solution
Difference in Capture and Post speed	SharePlex reads and processes records from the redo logs faster than it can post those operations to the target database with standard SQL statements.	For more information, see Tune the Post Process on page 360.
Full archive log directory	If Post appears stalled and will not shut down normally, but the only error in the Event Log is `sp_opst_mt - operation taking too long`, it could be that the archived log directory on the target system is full. If so, then Oracle cannot create new logs, and it suspends processing. Post stalls because it is waiting for Oracle.	Move some of the old archive logs to another device, or delete them to make room for new ones.

Commit reduction issues

If commit reduction is not working, it may be because records are not being dispatched quickly enough for there to be a valid message available after the commit. If a valid message is found after the commit message, then Post can skip the commit. Otherwise it issues the commit.

One way to find out if this is the issue is to stop the Post process completely, run a large amount of small transactions through replication, then start the Post process. If you do not see any commit reduction, then this would rule out this issue as the cause.

Post stopped

If Post stops, issue the **status** command in **sp_ctrl** to find out why.

• An *idle* status means there is no data in the post queue to post.

• A *stopped by user* status means that a SharePlex user stopped the Post process. To find out which user is responsible, view the user issued commands in the Event Log.

• A *stopped due to error* status means a replication or database error caused Post to stop.

The following are some potential causes for Post to stop unexpectedly.

Problem	Description	Solution
Correctable database errors	Post stops for database errors that can be corrected, so that you can fix the problem without risk of the data going out of synchronization. The errors and the faulty SQL statement are logged to the *databaseID*_**errlog.sql** file in the **log** sub-directory of the variable-data directory on the target system.	Use the information in the log file to correct the problem, then start Post again. Posting resumes from the point where it stopped.

Problem	Description	Solution
Non-correctable database errors	Some database errors, such as out-of-sync conditions, cannot be corrected. In that case, SharePlex reports the error to the Event Log, writes the error and SQL statement to the *databaseID*_**errlog.sql** file, and continues processing. Sometimes, errors that cannot be corrected cause Post to stop.	If the errors cannot be corrected but you want Post to continue processing, list the errors in the in the *database***msglist** file and then set the SP_OPO_CONT_ON_ERR or SP_OPX_CONT_ON_ERR parameter to 1, which directs SharePlex to ignore those errors and continue posting. For more information, see Continue to Post When there is a DML Error on page 225.
Locks on target tables	If a target table is locked, the Post process cannot apply a SQL statement and generates an error message. This message could mean that a user, application, or job is accessing the table and might have caused an out-of-sync condition. Or, for an Oracle target, it could mean that a **repair** command has locked the table.	Find out why the table was locked and resolve the access issue (unless due to the **repair** command). You might have to resynchronize the data if **DML** was performed on the table. For more information, see How to Resynchronize Source and Target Tables on page 341.
Configuration deactivation when there are named export queues	When you are using named export queues, and you deactivate the configuration, Post could stop with the following error instead of posting the remaining data from the queues: `sp_opst_mt (for o.qa920-o.qa920 queue q5) 15007 - Can't open poster queue que_NOEXIST: Queue does not exist.`	To start Post and finish replication, shut down SharePlex, then start it again.
Queue name is too long	Post stops if a queue name is too long.	Make certain the name assigned to named queues in the configuration file are no longer than 15 characters.
(Oracle) New tables are not added	For an Oracle source, tables created after activation are automatically added to replication.	The auto-add feature is controlled by parameters. Make certain the correct ones are set to achieve your goals. For more information, see Control Oracle DDL Replication on page 215.
(Oracle) No more open cursors	If you see the following error, Post has exceeded the available open cursors. `Warning: sp_opst_mt (for o.oracle-o.oracle queue oracle) Post has opened number cursors. No more available cursors! Exiting` Post requires a certain number of open cursors to the target database.	View the current database OPEN_CURSORS value using the following SQL statement: **select value from V$PARAMETER where name = 'open_cursors' ;** To determine an adequate OPEN_CURSORS value for SharePlex, see Adjust Open Cursors on page 363.

Other problems and solutions

Problem	Description	Solution
SharePlex cannot resolve a machine name	Sometimes machine names cannot be resolved between Unix or Linux system.	Add the IP addresses and names of all servers (Unix and Linux) in the replication network to the **/etc/hosts** file on all Unix and Linux machines.
sp_cop is using too much CPU time	SharePlex may be performing its overhead activities too frequently.	Increase the idle time of **sp_cop** with the SP_COP_IDLETIME parameter. See the parameter documentation in the SharePlex Reference Guide.
Corrupted source table	If a source table is corrupted or there is another reason that you do not want the replicated data to be posted to the target database, you can prevent posting for that table without removing it from the active configuration or affecting posting for other objects.	To disable posting for one or more tables, use the SP_OPO_DISABLE_OBJECT_NUM or SP_OPX_DISABLE_OBJECT_NUM parameter, which can be set to disable posting of both DML and DDL operations for specified object IDs. This parameter is disabled by default. When you are ready to begin posting to the target table again, disable the parameter again by resetting it to 0. For more information, see the parameter documentation in the SharePlex Reference Guide.
Only some columns were replicated	If you have a table that is configured for vertically partitioned replication but that table name also satisfies a wildcard, the specific listing (with the vertical partitioning) takes precedence over the wildcarded listing. Vertical partitioning cannot be used with full-table replication for the same table.	No action is needed if replication of the specified columns is desired. For more information, see Configure Vertically Partitioned Replication on page 141.
Conflict resolution generates compile errors	If you encounter compile problems with your conflict resolution routines, check whether any tables have the same names as their owners. A known issue in PL/SQL prevents the SharePlex conflict resolution logic from compiling the PL/SQL for tables whose names are the same as their owners. Oracle has stated that the issue will not be fixed. See Oracle TAR 2577886.996 for more information.	This issue does not affect replication; SharePlex replicates data for tables with identical owner and table names.

Common replication errors

The following table explains many of the common error messages that you might experience during replication.

sp_cop error messages	Cause	Solution
`sp_cop cannot setup; memory segment n in use.`	Processes that access the queues were still running when you last shut down **sp_cop**.	Kill those processes. SharePlex processes start with **sp_**. When all have been killed, **sp_cop** should start.
`Error: sp_cop can't setup shared memory statistics capability - exiting Error: sp_cop(shs) Cannot delete previous memory segment 303. Please check to see if any SharePlex processes are running (ps -ef | grep sp_). If there are some processes running then kill them and restart sp_cop.`	There are already one or more SharePlex**sp_cop** processes pointing to the same variable-data directory.	To run multiple sessions of SharePlex, you need to use separate variable-data directories for each one. For more information, see Run Multiple Instances of SharePlex on page 48..
Capture error messages		
`Capture time limit (300 sec) exceeded.`	Capture is not processing records, which could indicate a problem with the redo log. If, after a specific number of seconds, Capture cannot process a record, it stops, logs the record, and returns this message.	See Oracle Capture-related issues on page 312 for possible causes. If you cannot determine the cause of the problem, call Quest Support before a log wrap occurs.
`(Oracle) Log wrap detected`	The redo logs wrapped and Capture cannot locate the archive logs.	If archive logs are available, uncompress and restore them to the archive log directory. SharePlex looks for the archive logs first in the Oracle archive log list, then in the location specified by the SP_OCT_ARCH_LOC parameter. This parameter should always be set to the correct archived log directory. If you use compression for the archive logs, do not compress them until SharePlex is finished processing them. To determine the current log for SharePlex, issue the **show capture** command with the [**detail**] option in **sp_ctrl** on the source system. You can compress any logs that were generated before the current one. Note that this error also occurs when the archived log is corrupted.
Post error messages		
`operation taking too long.`	It is taking more than the internally allowed time to	For more information, see Oracle Post-related issues on page 314.

sp_cop error messages	Cause	Solution
	apply a SQL statement to the target instance.	
`Rowid not found`	SharePlex cannot locate the correct row to update in the target database.	Check for triggers, processes, or users that may have deleted the row on the target. For more information, see Solve Synchronization Problems on page 327.
`Database not available.`	Post cannot log into the target database.	Verify that the database is running, and determine whether someone changed the password of the SharePlex database account.
Oracle-related error messages		
`Can't access OBJ$Table`	SharePlex cannot access the Data Dictionary, which it must be able to do in order to replicate.	Check the O7_DICTIONARY_ ACCESSIBILITY Oracle tuning parameter and make sure it is set to TRUE (the default).
`Forward and backward operation counts do not match...`	Messages may be out of sequence.	Find out if Oracle was shut down before SharePlex. This can cause SharePlex to return errors, and in rare cases, corrupt the queues. The proper procedure is to shut down SharePlex and then shut down Oracle. For help resolving this problem, contact Quest Support.
`Error: sp_opst_mt (for o.blues920-o.ora9 queue bluesky) 15033 - Failed to execute SQL on table: QA.T_ DEST_1: ORA-00001: unique constraint (.) violated.`	A unique constraint was violated on the source system. The change entered the redo log, but Oracle rolled it back. The rollback also entered the redo log. SharePlex detects the constraint violation on the target.	Ignore this message. The tables are still in synchronization because Oracle rolled back the offending operation. This is an unavoidable error because of the way Oracle handles the violation.
SQL Cache error messages		
`Warning: Too many concurrent transactions. Will disable the SQL Cache capability.`	The SQL Cache size is set to 1, and more cursors are still needed. SharePlex disables the SQL Cache feature in this case.	No action is required if this is your intended configuration. For more information, see Tune SQL Caching on page 361.

`Warning: Running out of cursors. Number of cursors opened so far is number. Will attempt to decrease SQL Cache size.`
`or...`
`Notice: Shrinking SQL Cache`
	Post detected that it will exceed its maximum number of cursors and is trying to decrease the SQL Cache size.	No action is required unless the value gets to 1 and there are still not enough cursors. For more information, see Tune SQL Caching on page 361.

sp_cop error messages	Cause	Solution
`size to number per session.`		
`SQL Cache disabled.`	The SQL Cache feature is disabled.	No action is required if this is your intended configuration. For more information, see Tune SQL Caching on page 361.
Hints file error messages (Oracle only)		
`15050 – hint file not found`	SharePlex looks for the **hints**.*SID* file whether you use it or not. The location of this file is the **data** sub-directory of the SharePlex variable-data directory. If this file gets moved or deleted, SharePlex returns this error message.	To prevent this message, create a blank hints file in the **data** sub-directory of the variable-data directory. Name it **hints**.*SID*.

`15051 – missing column in the hint file (either table of index name)`
`15052 – syntax error for` *`tablename`*
`15053 – syntax error for indexname`
	The hints file is not configured correctly.	For more information, see Use Oracle INDEX Hints on page 360.

`15054 – source table's object_id not found in object cache`	The hints file includes a source table that is not in the active configuration. All tables in the hints file must be listed in the active configuration.	If this table is in the configuration, make certain that the owner name and table name are spelled in the hints file the same way as they are in the configuration.
`15055 – more than n valid entries were entered into the hints file`	The hints file permits only as many table-index combinations as the value set by the SP_OPO_HINTS_LIMIT parameter.	Either remove some of the table-index combinations or increase the value of SP_OPO_HINTS_LIMIT. For more information about this parameter, see the Post parameters documentation in the SharePlex Reference Guide.
`15056 – error allocation memory for hints`	This indicates a system-level memory problem; the hints file itself does not demand a significant amount of memory.	If you believe the system memory is sufficient, stop the Post process and start it again. If you are not using the hints file, you can ignore this error.
`17000 – error opening hint file`	SharePlex cannot open the hints file.	Check the file for corruption. If the file is valid, make certain there is sufficient read permission for the Post process. For more information, see Use Oracle INDEX Hints on page

sp_cop error messages	Cause	Solution
		360.
Environment-related error messages		
`sp_opst_mt: pid=num date/time src host/ sid=db01:N2PB /var/quest/vardir/log/ event_ log: Too many open files`	The system file descriptors setting is not 1024.	Set the ulimit to 1024. The ulimit can be set either as a system *hard limit* or a session-based *soft limit*, as follows: • Set a hard limit: (Recommended) A root user and system restart are required to change the hard limit, but the value remains fixed at the correct level to support SharePlex. Consult your System Administrator for assistance. • Set a soft limit: A soft limit setting stays in effect only for the duration of the **sp_cop** session for which it was set, and then it reverts back to a default value that may be lower than the hard limit and too low for SharePlex.
`06/29/00 08:05 System call error: sp_ocap(que) (for o.QA11 queue o.QA11) No space left on device devname 06/29/00 08:05 Internal error: sp_ocap (for o.QA11 queue o.QA11) 10705 - writecommit failed que_BUFWRTERR: Error writing buffer to file 06/29/00 08:05 Process exited sp_ocap (for o.QA11 queue o.QA11) [pid = 8692] -exit(1)`	SharePlex ran out of space for the queues on the disk.	For more information, see How to Resolve Disk Space Shortage on page 350.
`gethostbyname name failed - exiting`	The local hosts file is not configured properly.	The host name is not specified correctly in the hosts file (**/etc/hosts** on Unix and Linux). If this system is not part of a cluster, make corrections to the name in the file. If this system is part of a cluster, the virtual IP address must be mapped to a host alias in the hosts file. For instructions on configuring SharePlex in a cluster, see the SharePlex Installation and Setup Guide.

sp_cop error messages	Cause	Solution
Other error messages		
`Snapshot too old`	The read-consistent view required by SharePlex is no longer available.	Increase the size of the rollback segment.
`Parameter paramname does not exist in the paramdefaults file. Using hard coded default value. Please make sure that your param-defaults file is the correct version.`	SharePlex cannot find a parameter that it needs to reference. You have an older version of the **param-defaults** file than the version of SharePlex that you are running. Someone might have updated the SharePlex binaries using a downloaded patch, and may not have installed the latest **param-default** file.	Always check for an updated **param-defaults** file when you manually update SharePlex.
`Invalid DATE format detected in record with rowid=rowid, on obj object_id. See capture log for detail.`	A user or application entered an invalid date value into the Oracle database that bypassed the database's validity check.	Set the SP_OCT_DEF_ parameters to enable SharePlex to fix the format of dates and times if they are not caught by the database checks. See the SP_OCT_DEF parameters documentation in the SharePlex Reference Guide.
`shs_SHMERR: an error occurred with shared memory.`	You ran the **qview** utility without shutting down SharePlex (**sp_cop**).	Shut down SharePlex and re-run **qview**.

Solve Oracle DDL Replication Problems

This section reviews many common problems and solutions encountered when replication of Oracle DDL is active. For more information about SharePlex DDL support, see Configure DDL Replication on page 215.

If the issue you are experiencing is not listed in this documentation, search the SharePlex Knowledge Base at:

https://support.quest.com.

The Knowledge Base provides filtering options and links to other resources that can help you use and troubleshoot SharePlex.

DDL is not replicating

Only some Oracle DDL is enabled by default. Other DDL support must be explicitly enabled with parameter settings.

The SP_OPO_STOP_ON_DDL_ERR parameter is defaulted to direct the Post process to stop if there is an error applying the DDL, to enable you to correct the problem to keep the databases synchronized. This parameter should be set to 1 (on). When enabled, messages such as the following notify you that DDL was skipped.

```
Skipping generic 9i DDL operation, schema (bob) could not be set.

FAILED DDL Replication for "create user bob."
```

Solution: Make certain the following parameters are set to appropriately. See Control Oracle DDL Replication on page 215 for more information about how to use these parameters to configure DDL replication.

- SP_OCT_REPLICATE_DDL
- SP_OCT_AUTOADD_ENABLE
- SP_OCT_AUTOADD_MVIEW
- SP_OCT_AUTOADD_SEQ
- SP_OCT_REPLICATE_ALL_DDL

Replicated DDL is not completely displayed in the Event Log

SharePlex prints replicated DDL to the Event Log, but it truncates statements longer than 2,000 characters. Only the first 2,000 characters are recorded in the log.

Solution: None required.

Solve Queue Problems

This topic helps you solve replication problems that may be related to the SharePlex queues.

If the issue you are experiencing is not listed in this documentation, search the SharePlex Knowledge Base at:

https://support.quest.com.

The Knowledge Base provides filtering options and links to other resources that can help you use and troubleshoot SharePlex.

SharePlex is running out of disk space

Data accumulating in the queues can cause SharePlex to run out of disk space. The following are some possible causes and solutions.

Problem	Description	Solution
Stopped replication processes	When replication processes stop, data accumulates in the queues.	If a process was stopped by a user, find out why and then start it as soon as possible so that SharePlex can process the accumulated data. If a process stopped because of an error, view the Event log with the **show log** command to find out what happened, then resolve the problem so that processing can continue and the queue backlog can be reduced.
Large operations	The Post queue may become large when storing large transactions for which there has not yet been a COMMIT. To allow for rollbacks and data recovery, SharePlex retains data in the post queue until it receives the COMMIT.	View the post queue with the **qstatus** command. If the value in the **Backlog (messages)** field remains constant or is shrinking, while the value in the **Number of messages** field is increasing, Post is waiting for a COMMIT before releasing the data. Use the **show post detail** command to verify that Post is processing transactions normally. If possible, set the COMMIT point for your application to 500 to generate smaller SQL statements for SharePlex to process. Also, consider creating a configuration that uses named post queues which isolate tables that are known for long transactions. For more information, see Configure Named Post Queues on page 113. If the accumulation of messages in the Post queue is threatening to cause disk space issues, you can stop Import until Post can clear out some of the operations from other transactions. For more information, see How to Resolve Disk Space Shortage on page 350.
(Oracle) Capture is processing archive logs	If Capture is processing archive logs, the capture queue consumes disk space while the archived records are being processed.	If the location of Capture in the logs is far behind that of the database, the latency between the source and target data may be too large to be acceptable to the target users. Rather than add more disk space to the variable-data directory, it might be more practical to run **ora_cleansp** to clean up the replication environment and queues, resynchronize the data, and reactivate the configuration.

Problem	Description	Solution
		If SharePlex continues to run out of disk space because Capture is processing the archive logs, the SharePlex Support team can help you tune the performance of Capture and adjust the redo configuration.
Unplanned increase in source transaction activity	An unplanned increase in activity on the source system can cause data to accumulate in the Post queue and approach the maximum amount of allocated disk space.	For more information, see How to Resolve Disk Space Shortage on page 350.

There is a “failure to write and open queue” error

If there is a sequence of messages in the Event Log similar to the following, stop and start **sp_cop**.

```
1 08-06-12 13:20:17.089485 2384 1 sp_ordr(rim) (for o.user queue o.user) Error
opening output queue rv=9 que_open(-,writeuser+ X,0x0a02d364+PR+o.user+sp_
ordr+o.user)

Notice 08-06-12 13:20:17.089485 2384 1 sp_ordr (for o.user queue o.user) data
route to a02d364.48.7e failed err=100

Error 08-06-12 13:20:17.089485 2384 1 sp_ordr (for o.user queue o.user) 11004
- sp_ordr failure writing to queue(s)

Notice 08-06-12 13:20:17.089485 2384 1 sp_ordr (for o.user queue o.user) Exit
sp_ordr to retry rim-write.

Info 08-06-12 13:20:17.089485 2384 1 Process exited sp_ordr (for o.user queue
o.user) [pid = 8183] - exit(1)
```

The queues are corrupted

If the system that hosts SharePlex fails, one or more SharePlex queues can become corrupted, and there will be errors reading from and writing to them. In that case, the **purge config** and **abort config** commands cannot be used, because they rely on the queues to operate.

Solution: Contact SharePlex Support to resolve queue corruption issues.

The post queue seems too large

If the size of the post queue seems too large relative to the number of messages it contains, that is not unusual. The SharePlex post queue actually consists of a number of sub-queues, each approximately corresponding to a user session on the source system. A sub-queue can have one or more files associated with it, each with a default size of 8 MB. If the entire 8 MB of size is not used, the file remains on the system even after the data has been posted and read-released.

Solution: The size for the post queue in a status display is the actual disk space used by all of the queue files. You can eliminate obsolete files that have been read-released by using the **trim** command in the SharePlex **qview** utility. Thecommand preserves files containing data not yet posted and committed to the target database. See the SharePlex Reference Guide for more information about the **qview** utility.

Solve Synchronization Problems

This section reviews the causes and solutions for common synchronization problems. If you try these solutions and are still having problems, contact Quest Support.

For more information, see Understand the Concept of Synchronization on page 35.

If the issue you are experiencing is not listed in this documentation, search the SharePlex Knowledge Base at:

https://support.quest.com.

The Knowledge Base provides filtering options and links to other resources that can help you use and troubleshoot SharePlex.

How SharePlex reports out-of-sync conditions

For all objects **except those involved in transformation**, SharePlex verifies that the source and target data in a given operation are synchronized before posting the replicated data to the target. SharePlex does not verify synchronization if transformation is being used because:

- Transformation changes the target data, so before and after images cannot be compared.
- The transformation routine posts the data, not SharePlex.

When SharePlex determines that source and target data are different, it generates error conditions **but continues to post other data** from the post queue. To direct Post to stop processing altogether when it detects an out-of-sync condition, change the SP_OPO_OUT_OF_SYNC_SUSPEND (Oracle) or SP_OPX_OUT_OF_SYNC_SUSPEND (Open Target) parameter. See the Post parameter documentation in the SharePlex Reference Guide.

When an out-of-sync condition occurs, the Post process logs a message in the Status Database and also to the Event Log. To view these files in **sp_ctrl**:

- Status Database: Use the **show statusdb** or **show sync** command.
- Event Log: Use the **show log** command.

Use these commands frequently to monitor for out-of-sync errors.

The following is an example of how SharePlex reports an out-of-sync condition.

```
sp_ctrl (irvspxu14:8567)> show sync

Out Of Sync Status Database irvspxu14

Count Details

----- --------------------

3 Table "SCOTT"."TG_TEST1" out of sync for queue irvspxu14 since 16-Jun-

08 17:06:33

3 Table "SCOTT"."TG_TEST2" out of sync for queue irvspxu14 since 17-Jun-

08 15:47:58

1 Table "SCOTT"."TG_TEST3" out of sync for queue irvspxu14 since 17-Jun-

08 15:52:03
```

When data goes out of synchronization, SharePlex logs the failed SQL statements to the *database_ID*_**errlog.sql** file in the **data** sub-directory of the SharePlex variable-data directory.

IMPORTANT: If you see an out-of-sync message in the Status Database and in the Event Log, but there is no record in the *database_ID*_**errlog.sql** file for the transaction, **do not ignore** those messages. They could be associated with a ROLLBACK. Regardless of whether or not a transaction is rolled back, SharePlex still compares the pre-images of the source and target rows. If they are different, that indicates that the data is out of synchronization. Only when a transaction is committed on the source but fails on the target does SharePlex log it to the *database_ID*_**errlog.sql file**, to give you a record of the statement that should have been applied as a tool for problem solving and for manually applying the statement if appropriate. Rolled back statements are canceled operations, and therefore not logged on the target.

Detect false out-of-sync conditions

Sometimes an out-of-sync message can be false, and the data is not out-of-sync. You can compare the source and target data by using the **compare** command, or you can perform the following comparison manually.

To compare with the compare command:

See the **compare** commands in the SharePlex Reference Guide.

To verify that data is out-of-sync:

1. Get the rowid of the affected row from the *database_ID*_**errlog.sql** file in the variable-data directory on the target system.
2. Using the rowid in the WHERE clause, run a SELECT query on the source table to get the row data for this rowid.
3. Run a SELECT query on the target table, using the row data that you got from the source query in the WHERE clause.
4. Compare the query results. If they are different, the rows are out of synchronization. If they are the same, the rows are synchronized.

NOTE: If none of the columns are unique in a table, Compare might show incorrect results.

Common out-of-sync conditions and solutions

The following are common ways in which data goes out of synchronization. In most cases, SharePlex detects out-of-sync conditions and returns an error message, but there are some situations where an out-of-sync condition is hidden, and SharePlex will not return an error.

Out of sync condition	Description	Solution
Incorrect cleanup procedure	If one of the *database*_**cleansp** cleanup utilities was run on some, but not all, systems associated with an active configuration, SharePlex perceives an out-of-sync condition.	Determine whether or not the cleanup utility was run on all systems by viewing the Event Log. The log tells you if and when it was run on each system. It also tells you if and when the configuration was activated. You can compare the times for those events to determine what happened. If the cleanup was not completed on all replication systems for this configuration, run the cleanup utility on all systems. Because the cleanup removes replication queues and processes and deactivates the configuration, you must perform initial synchronization again.
DDL changes	Some DDL-related causes of out-of-sync conditions are: • Non-replicated DDL changes are made to source objects, but the configuration was not reactivated so that the objects can be re-analyzed. • DDL that SharePlex replicates also gets performed manually on the target.	For a list of supported DDL, refer to the SharePlex Release Notes. To undo duplicate DDL changes made manually and also by SharePlex:

Out of sync condition	Description	Solution
		1. Stop the Post process (it might already be stopped). `sp_ctrl (sysB)>` **stop post** 2. Alter the target table to undo the DDL changes. 3. Start Post and let SharePlex post the replicated DDL (which is still in the post queue). `sp_ctrl (sysB)>` **start post**
DML changes made directly to the target	If applications or users make DML changes to the tables on the target, the results of those changes will cause hidden out-of-sync conditions until Post attempts to apply a replicated change to the affected rows. When the change is applied, SharePlex returns an out-of-sync error.	Prevent all DML access to the target tables that are in replication. You can use the **compare** and **repair** commands to compare tables for out-of-sync rows and then repair those rows. For more information, see the command documentation in the SharePlex Reference Guide.
Triggers on target objects	Triggers must be disabled on target objects. The triggers fire on the source system and SharePlex replicates their effects to the target.	If triggers have fired on the target system, the objects changed by the triggers are out of synchronization and must be resynchronized.

Out of sync condition	Description	Solution
		For more information, see How to Resynchronize Source and Target Tables on page 341. **To disable the effects of triggers on target Oracle objects after the data is resynchronized, you can either of the following:** 1. Run the **sp_add_trigger.sql** script, which directs triggers to ignore the SharePlex Oracle user. See the utilities documentation in the SharePlex Reference Guide for more information about the trigger scripts. 2. Disable the triggers if they are not needed.
Unnecessary constraints	The only constraints that are necessary on target tables in a one-way replication configuration are primary and unique key constraints. CHECK constraints are not necessary on the target because they are satisfied on the source. FOREIGN KEY and ON DELETE CASCADE constraints are also satisfied on the source, and SharePlex replicates the child operations to the correct tables on the target. For an Oracle target, you can leave ON DELETE CASCADE constraints enabled if you configure SharePlex to ignore them.	See Set up Oracle database objects for replication on page 1for how to handle constraints on replicated objects.
Duplicate entries in the configuration file.	Duplicate entries, where the source, target, and routing map are identical, cause double posting on the target.	Copy the configuration file to a new file. Find and remove any duplicates in the new file.

Out of sync condition	Description	Solution
		Perform a resynchronization and reactivation. For more information, see How to Resynchronize Source and Target Tables on page 341.
Lack of disk space	Data goes out of synchronization if user transactions continue when SharePlex does not have enough room to accommodate them in the queues. This can happen if: • The network or target system was unavailable and too much data accumulated in the export queue. • The target was unavailable and too much data accumulated in the Post queue. • Capture lost pace with the logging of source transactions. In this case, data accumulates in the capture queue. • A SharePlex process was stopped, but not restarted. • The **flush** command was issued, but Post was not started again.	For more information, see How to Resolve Disk Space Shortage on page 350.
Changes to column conditions in horizontally partitioned replication	Out-of-sync conditions can result from the use of horizontally partitioned replication in the following cases: • A column condition value is updated and the new value no longer satisfies the row selection criteria. • A row that does not satisfy the column condition is updated to satisfy the condition.	Create column conditions so that partition shift does not occur. For more information, see Configure Horizontally Partitioned Replication on page 118.
Not reactivating after a configuration change.	If a table was added to the configuration, but the configuration was not reactivated, operations on that table are not being replicated. **NOTE**: For Oracle source tables, the auto-add feature is enabled by default, and any new table whose name satisfies a wildcard in the configuration file gets automatically added to replication. For more information, see Control Oracle DDL Replication on page 215.	Resynchronize the affected tables, then reactivate the configuration so that SharePlex can update its object cache. For more information, see How to Resynchronize Source and Target Tables on page 341.
Queue corruption	If the SharePlex queues are corrupted, such as if there is a system failure, the data in them can be lost. This requires a resynchronization.	For more information, see How to Resynchronize Source and Target Tables on

Out of sync condition	Description	Solution
		page 341. (Oracle) To avoid queue corruption during system failure, you can use the parameter SP_QUE_SYNC. See the Queue parameters documentation in the SharePlex Reference Guide.

Oracle-related out-of-sync conditions and solutions

The following are common synchronization issues that relate specifically to replication between Oracle databases. In most cases, SharePlex detects out-of-sync conditions and returns an error message, but there are some situations where an out-of-sync condition is hidden, and SharePlex will not return an error.

Out of sync condition	Description	Solution
DDL changes	Some DDL-related causes of out-of-sync conditions are: • Non-replicated DDL changes are made to source objects, but the configuration was not reactivated so that the objects can be re-analyzed. • DDL that SharePlex replicates also gets performed manually on the target.	For a list of supported DDL, refer to the SharePlex Release Notes. To undo duplicate DDL changes made manually and also by SharePlex: 1. Stop the Post process (it might already be stopped). `sp_ctrl(sysB)` > **stop post** 2. Alter the target table to undo the DDL changes. 3. Start Post and let SharePlex post the replicated DDL (which is still in the post queue). `sp_ctrl(sysB)` > **start post**
Inadequate or non-existing conflict resolution routines	Conflict resolution procedures are required in a peer to peer (active-active) configuration. SharePlex uses the conflict resolution procedures to determine which operation to post when	Revise and test the conflict resolution procedures, then resynchronize the data. For more information, see How to Resynchronize Source and Target Tables on page 341.

Out of sync condition	Description	Solution
	the same data change is received from different systems.	
Log wrap	If the redo logs wrap before Capture can process the data it needs, the data can go out of synchronization if archived logging is not enabled or if the archive logs needed by Capture were removed. (Normally, Capture would access the archive logs and continue replicating.)	• See Correct the problem first on page 336 • If archiving is not enabled, there are no archive logs for SharePlex to read. Data lost after the log wrap cannot be recovered. Enable archived logging, and resynchronize the data. For more information, see How to Resynchronize Source and Target Tables on page 341. • If SharePlex is many logs behind Oracle, consider resynchronizing the data instead of restoring the logs. It might take less time than Capture would take to process the missing records from the archive logs. In addition, it eliminates the possibility that the capture queue could exceed free disk space while processing the archive logs. You can base your decision on the size of the redo logs and the number of tables being replicated, both of which determine how much information Capture must process. Also take into consideration how much latency the users of the target data can tolerate. • If archive logs are available, copy the appropriate logs back to the archived-log directory on the source system, or use the SP_OCT_ARCH_LOC parameter to point SharePlex to their location.
LONG columns	If a table has no primary or unique key, SharePlex builds a simulated key based on all of the columns *except the LONG and LOB columns*. If the LONG columns in the target rows are the only columns that contain unique values, multiple rows could meet the criteria for the simulated key. SharePlex could apply the UPDATE or DELETE to the wrong row without the error being detected, causing the table to go out of synchronization without an error message.	If you can create a key from columns that ensure uniqueness on the target table(s), you can avoid this kind of out-of-sync condition. After you create the key, resynchronize and re-activate those objects so that SharePlex can update its object cache. If adding a primary or unique key is not possible (such as when packaged applications are in use), uniqueness of rows on the target system cannot be ensured.

Out of sync condition	Description	Solution
Changes to keys	If tables use automatically generated numerical sequences as keys, and a key value changes, this may cause duplicity on the target system. If the new value already exists as a key in another row on the target system, SharePlex returns a unique-key constraint violation and out-of-sync error. This can happen when you update values using an **x** +*n* formula, where *n* is an incremental increase. It is possible that one of the **x** +*n* values will equal an existing value.	Create the sequences so that the updates cannot result in a duplication of keys on the target system.
DBMS_ SCHEDULER procedures running on source and/or target system	The effects of these procedures, such as objects being created, manipulated and dropped outside replication, may not be visible to replication or may not be supported, resulting in data changes that cause out-of-sync conditions.	Exclude source and target objects from the jobs.
Virtual private database	If replicating data that is configured as a virtual private database, the SharePlex database user may not have the access rights to capture the data. Any changes to that data will not be reflected on the target.	If you do not need that data replicated to the target, you can filter it out of the SharePlex configuration by means of partitioned replication. If you want the data to be replicated, assign the SharePlex user the correct access rights.
PK/UK logging not enabled	Certain replication problems can be prevented by logging key values. SharePlex fetches key values based on the rowid. Any operation that changes the rowid, such as ALTER TABLE...MOVE, can cause the wrong key values to be used for subsequent DML operations.	SharePlex recommends that both primary key and unique key supplemental logging be set, or that a supplemental log group on unique columns be defined for every table in replication.

Correct the problem first

If data is out-of-synchronization, do the following before you resynchronize the data:

1. Determine why it happened before you resynchronize the data. Otherwise, the problem can repeat itself and result in more data going out of synchronization.
2. Stop the Post process to prevent further errors. If the accumulation of messages in the Post queue is threatening to cause disk space issues, and if there is enough disk space available on the source system, you can stop Import until Post can clear out some of the operations from other transactions. For more information, see How to Resolve Disk Space Shortage on page 350.
3. View the Status Database and the Event Log to determine the cause of the problem.
4. Resolve the problem.

Resynchronize data

For more information, see How to Resynchronize Source and Target Tables on page 341.

Solve Compare Command Errors

If you are having trouble with any of the compare or repair commands, refer to the following for assistance.

If the issue you are experiencing is not listed in this documentation, search the SharePlex Knowledge Base at:

https://support.quest.com.

The Knowledge Base provides filtering options and links to other resources that can help you use and troubleshoot SharePlex.

Problem	Description	Solution
Oracle error 904	`Error 904 calling oexec in de_select_prepare_to_fetch.` A comparison failed because the source and target tables being compared are structurally different. The **compare** and **repair** commands do not detect and repair out-of-sync conditions caused by unsynchronized DDL operations or tables that are not structurally identical.	Do a DESCRIBE on both tables. It probably will show that the tables do not have an equal number of columns or the data types are different. After you correct the DDL problem, you can run a **repair** to resynchronize the values in the rows.
"Too many users" error	`Can not add DataEquator queue reader que_TOOMANYUSERS: User table is full.` The maximum number of processes reading from and writing to the SharePlex queues was exceeded. No more than 20 processes can read from and write to the post queue at the same time, including the replication processes and the **compare** and **repair** processes. An error is most likely to occur when a repair option is used, because a repair accesses the queue much longer than a comparison without a repair.	There is no workaround or way to adjust the limit, nor is there a way to determine how many compare processes can run concurrently without exceeding the limit. **TIP:** To compare multiple tables at the same time, without being restricted by process limitations, use the **compare using** command. To limit the tables being compared, create a new configuration containing only the ones that you want to compare, and use it for the comparison. (Do not activate that configuration.) All of the tables are compared with one compare using process.
A client process fails to die	When a **sp_desvr** server process dies, the associated **sp_declt** client processes usually die. If a process does not die, you can kill it.	For more information, see Kill compare processes on page 338.
A server process fails to die	When you kill a **sp_declt** client process or it dies on its own — or if the **sp_desvr** server process has not communicated with the client — the **sp_desvr** server process usually exits after a certain amount of time, which is controlled by the SP_DEQ_TIMEOUT parameter.	For more information, see Kill compare processes on page 338.

Kill compare processes

To kill a client process:

If you need to kill a **sp_declt** client process, and there are multiple compares running, you can determine which one to kill in one of the following ways:

- By viewing the **sp_declt** log file — In the file, look at the Session IDs of the **sp_declt** processes and find the one that matches the PID of the **sp_desvr** process that died. That is the **sp_declt** process to kill. The **sp_declt** Session ID is the same as the PID of the associated **sp_desvr** process.
- By viewing the Event Log — The Event Log records the startup of each **sp_declt** client process and its PID. A subsequent entry in the log records the compare log file to which the process is writing. Within the compare log file's name in that entry is the PID of the server process. For example, in the following sample entry, the **sp_declt** process PID is 2450. The process writes to compare log ../o734v32a_declt- 1228-01.log. The 1228 is the PID of the server process.

  ```
  05/04/01 17:01 Process launched: sp_declt (for o.o734v32a-o.o734v32a- 87056 queue all) [pid = 2450]
  ```

  ```
  05/04/01 17:01 Notice: sp_declt(deq) (for o.o734v32a-o.o734v32a-87056 queue all) Opened DataEquator session log file /u10/julia30014/var7/ log/o734v32a_ declt-1228-01.log
  ```

  You can search the log file names for the server process that died, and look for the client process associated with that log file to determine the correct PID to kill.

To kill a server process:

If you need to kill a **sp_desvr** server process when a **sp_declt** client process dies, look in the Event Log to find out which log the **sp_declt** client process was writing to. The Event Log records the startup of each client process and its PID. A subsequent entry in the log records the compare log file to which the process is writing. Within the compare log file's name in that entry is the PID of the server process. For example, in the following sample entry, the **sp_declt** process PID is 2450. The process writes to log ../o734v32a_declt-1228-01.log. The 1228 is the PID of the server process, and that is the process to kill.

```
05/04/01 17:01 Process launched: sp_declt (for o.o734v32a-o.o734v32a- 87056 queue all) [pid = 2450]
```

```
05/04/01 17:01 Notice: sp_declt(deq) (for o.o734v32a-o.o734v32a-87056 queue all) Opened DataEquator session log file /u10/julia30014/var7/ log/o734v32a_declt-1228-01.log
```

Solve other Replication Problems

This section reviews solutions to other replication problems.

If the issue you are experiencing is not listed in this documentation, search the SharePlex Knowledge Base at:

https://support.quest.com.

The Knowledge Base provides filtering options and links to other resources that can help you use and troubleshoot SharePlex.

Common connection errors

The following are solutions to common errors when starting **sp_ctrl**, or with forming a connection with the **host**, **port** or [**on** *host*] commands in **sp_ctrl**.

Explanation of connection error messages

Error	Cause	Solution
`Host unknown: cannot form connection`	Appears when either the **host** command or [**on** *host*] option is issued.	Verify that the system to which you want to connect is running and that you are using the correct system name.
`Network unreachable`	The network is down.	Find out how long the network administrator expects it to last. If the downtime could cause the SharePlex queues to exceed their disk space, take measures to avoid having to resynchronize the data. For more information, see How to Resolve Disk Space Shortage on page 350.
`Export cannot connect to import on hostname: timeout waiting for ack`	Export cannot connect to the target because its connection was timed out by the network configuration. This can occur when there is little replication activity and the network has a timeout setting.	Set the SP_XPT_KEEPALIVE parameter to 1. This setting tells the Export process to send a "hello" message to Import at regular intervals to prevent the TCP timeouts.
`User is not authorized as a SharePlex user -- check /etc/group`	You do not have user permissions to execute the operation.	SharePlex users must be listed in the **/etc/group** file (Unix and Linux) under one of the SharePlex user groups: SharePlex Admin, spoper, spview.
`unauthorized connection attempt from host hostname. net`	A connection from a remote machine was denied because its name is not listed in the **auth_hosts** file.	See the error message for the name of the system. To allow that system to connect to the **sp_cop** on the local system, add its name to the **auth_hosts** file.

Common command errors

Error	Cause	Solution
`Deactivate/flush a nonactive datasource`	You are attempting to flush a configuration that is not active.	None required.
`Bad routing specification`	The syntax in the routing map is incorrect.	For more information, see Routing Specifications in a Configuration File on page 69.
`Status db file is corrupt.`	The Status Database has been damaged.	Shut down SharePlex and remove the **statusdb** file, which resides in the **data** subdirectory of the SharePlex variable-data directory. SharePlex will create another one when you start **sp_cop** again.
`Parameter does not exist in database.`	You tried to set a parameter, and you entered the wrong name or the parameter is deprecated for your SharePlex version.	Use the **list param** command to view the SharePlex parameters for your version and to verify the spelling.
`Parameter type checking failed - look in param - defaults file.`	You might have entered the wrong data type for the parameter.	Use the list param command to determine the valid data type.

`Unknown service specified.`
`or...`
`No such module.`
`or...`
`Service may be only one of: post, read, import, export, capture, all.`
	Valid service (process) names are **capture**, read, **export**, **import**, **post**.	Issue the command again with the correct name.

`Command was called with an invalid argument.`
`or`…
`Unknown keyword used in command.`
	The command contains invalid input.	Issue the **help** command to view valid input for the command.

`Permission denied for command - check your authorization level.`	You are not a member of the user group that can issue this command.	Issue the **authlevel** command to view your authorization level.
`Default host is not defined: use the 'host' command or [on host] option.`	SharePlex cannot to determine which system you want the command to affect.	Either establish a default host with the **host** command or use the [**on** *host*] option with the command that you want to issue (if available).

How to Resynchronize Source and Target Tables

The following instructions help you decide how to resynchronize out-of-sync tables.

- For Oracle tables, if only a few tables are out of synchronization, and they are not large, you can use the **compare** command in **sp_ctrl** to see how many rows are out-of-sync in each one. If the number of out-of-sync rows is small, you can run the **repair** command to resynchronize them. If the number of out-of-sync tables is large, it might take less time to re-synchronize the databases than it will to use the **compare** and **repair** commands to repair them. For more information, see Overview of Compare and Repair on page 353.
- For both Oracle and Open Target databases, you can use the SQL statement(s) in the *ID*_**errlog.sql** file to patch the tables manually after you correct the problem. For more information, see Manually patch out-of-sync tables on page 341.
- **You can resynchronize the data in a number of ways. See the following topics:**
  - Manually patch out-of-sync tables on page 341
  - Resynchronize by copying the source tables on page 342
  - Resynchronize with Oracle transportable tablespace on page 343
  - Resynchronize with an Oracle hot backup on an active database on page 344

Manually patch out-of-sync tables

Valid for: All database types

If the number of synchronization errors is small, you can try to repair out-of-sync tables manually. When the Post process detects an out-of-sync condition, it ignores the error and continues to apply the next operations in the post queue. However, Post logs source SQL statements that cause out-of-sync errors to an error file calle*ID*_**errlog.sql**. (*ID* is the identifier that SharePlex uses for *the target* instance, such as the ORACLE_SID or the database name.) You can apply those SQL statements to a target table through the native SQL interface of the database. Because this procedure bypasses the comparison made by Post, the operations should succeed assuming the structure of the target table did not change.

SharePlex stores *ID*_**errlog.sql** in the **data** sub-directory of the variable-data directory on the target system. The entries in the file are similar to the following example:

```
-- Host (irvlabua) Sid (al920u64)
-- session 2, 1 error --
--
-- [1] Tue Dec 11 13:31:32 2007
-- redolog seq#/offset 26622/26980368
-- redolog timestamp 641050290 (12/11/15 13:31:30)
-- original rowid AAE0m8AAWAAAAFEAAA
-- -- NOT FOUND
delete from "SP_5"."QA_LOB_DISABLE_INROW" t where rownum = 1 and "KEY"='01';
```

To apply the SQL manually:

1. Stop user access to the affected source table.
2. On the target system, open the *ID*_**errlog.sql** file.
3. Apply the SQL statement(s) to the target table.
4. Reactivate the configuration if you had to make any changes to it.

   ```
   sp_ctrl> activate config filename
   ```

5. Allow user access to the source table.

Resynchronize by copying the source tables

Valid for: All database types

This procedure restores synchronization to out-of-sync target tables by applying a copy of the source tables. You only need to resynchronize the tables that are out of synchronization, so users can continue accessing all other tables.

IMPORTANT! Before you start, review this procedure and see the SharePlex Reference Guide for more information about the commands that are used.

1. On the **source** and **target** systems, make sure **sp_cop** is running.
2. On the **target** system, run **sp_ctrl**.
3. [If necessary] On the **target** system, issue the **show sync** command to identify the tables that are out of synchronization.

   ```
   sp_ctrl> show sync
   ```

4. On the **source** system, stop activity for the out-of-sync tables.
5. On the **source** system, run **sp_ctrl**.
6. On the **source** system, issue the **flush** command.

   NOTE: This command has additional options for use with named queues or multiple targets. See the SharePlex Reference Guide for more information about this command.

   ```
   sp_ctrl> flush datasource
   ```

7. On the **source** system, copy the tables.
8. On the **source** system, reactivate the configuration file if you had to make any changes.

   ```
   sp_ctrl> activate config filename
   ```

9. On the **source** system, allow users back onto the **source** tables.
10. On the **target** system, issue the **status** command until it shows that Post stopped.

    ```
    sp_ctrl> status
    ```

11. On the **target** system, restore the tables.

12. On the **target** system, disable or modify triggers, referential integrity constraints and check constraints according to the requirements of your replication strategy.
13. On the **target** system, determine the status ID of each message by viewing the Status Database.

    ```
    sp_ctrl> show statusdb detail
    ```

14. On the **target** system, clear each message with the following command.

    ```
    sp_ctrl> clear status statusID
    ```

15. On the **target** system, start the Post process.

    ```
    sp_ctrl> start post
    ```

Resynchronize with Oracle transportable tablespace

Valid for: Oracle database

The transportable tablespace feature enables you to resynchronize numerous out-of-sync tables quickly and with minimal downtime. To use the transportable tablespace feature, follow the instructions in the Oracle documentation for generating a tablespace set, moving the tablespace set to the target database, and plugging the set into the database. The following instructions contain steps only for using this feature to resynchronize data. It assumes familiarity with using the transportable tablespace feature.

IMPORTANT! Before you start, review this procedure and see the SharePlex Reference Guide for more information about the commands that are used.

1. On the **source** system, set the source tablespace to READ ONLY.

   ```
   SQL> ALTER TABLESPACE name READ ONLY;
   ```

2. On the **source** system, run **sp_ctrl**.
3. On the **source** system, issue the **flush** command in **sp_ctrl**.

   NOTE: This command has additional options for use with named queues or multiple targets. See the SharePlex Reference Guide for more information.

   ```
   sp_ctrl> flush datasource
   ```

4. Export the metadata to an export file according to the Oracle documentation.
5. When the export is finished, copy the datafiles to a secondary location on the **source** system. This minimizes the impact on the source database of copying the files to the target system.
6. On the **source** system, set the source tablespace(s) to READ WRITE mode.

   ```
   SQL> ALTER TABLESPACE name READ WRITE;
   ```

7. On the **target** system, drop the existing datafiles and tablespaces from the target database so that the copied files can be applied.
8. Copy the files from the secondary location on the **source** system to the **target** system.
9. On the **target** system, use the Oracle import utility to import the metadata and the tablespace definitions.

10. On the **target** system, set the tablespace(s) to READ WRITE mode.

    ```
    SQL> ALTER TABLESPACE name READ WRITE;
    ```

    NOTE: SharePlex must be the only user permitted to have write access to the target tables, unless you are using peer-to-peer replication.

11. On the **source** system, reactivate the configuration file if you had to make any changes to it.

    ```
    sp_ctrl> activate config filename
    ```

12. On the **target** system, run **sp_ctrl**.
13. On the **target** system, start the Post process.

    ```
    sp_ctrl> start post
    ```

Resynchronize with an Oracle hot backup on an active database

Valid for: Oracle database

When you use an Oracle hot backup and the **reconcile** command to resynchronize a target instance, users can continue to access the production data while the backup is made and applied.

IMPORTANT:

- To resynchronize **centralized reporting**, such as a data warehouse, you cannot use a hot backup from all source systems. One backup would override the data from the previous one. You can use a hot backup of one of the source instances to establish the target instance, and then use another method such as export/import or transportable tablespaces to copy the tables from the other instances.
- To resynchronize **peer-to-peer** replication, you must quiet **all** of the secondary source systems for the duration of this procedure. Move all users to the primary system, and then follow the procedure. After the procedure has been performed on **all** of the secondary systems, users can resume activity on them.
- Before you start, review this procedure and see the SharePlex Reference Guide for more information about the commands that are used.

To resynchronize with a hot backup:

1. On the **source** and **target** systems, run **sp_ctrl**.
2. On the **target** system, stop the Post process. This allows the replicated data to accumulate in the post queue until the target instance has been recovered and reconciled.

    ```
    sp_ctrl> stop post
    ```

3. On the **source** system, run the Oracle hot backup.
4. On the **source** and **target** systems, verify that **sp_cop**, **sp_ctrl** and all SharePlex processes (Capture, Read, Export, Import, Post) are running.

    ```
    sp_ctrl> status
    ```

5. Switch log files on the **source** system.

    - To recover the database to a sequence number, make a note of the highest archive-log sequence number.
    - To recover the database to an Oracle System Change Number (SCN), pick an SCN to recover to on the target database.

6. Recover the **target** database from the hot backup:

    - If recovering to a sequence number, recover the database from the hot backup using the UNTIL CANCEL option in the RECOVER clause, and cancel the recovery after Oracle has fully applied the log from the previous step.
    - If recovering to a SCN, recover the database from the hot backup using the UNTIL CHANGE *SCN* option in the RECOVER clause, and cancel the recovery after Oracle has applied the logs matching the SCN from the previous step.

7. Open the database with the RESETLOGS option.

8. On the **target** system, issue the **reconcile** command. If you are using named post queues, issue the command for each one. Issue the **qstatus** command if you are unsure of the queue name.

    - If recovering to a sequence number, substitute the sequence number of the log that you noted in step 5.

        `sp_ctrl>` **reconcile queue** *queuename* **for** *datasource-datadest* **seq** *sequence_number*

        Example: **reconcile queue SysA for o.oraA-o.oraA seq 1234**

    - If recovering to a SCN, substitute the SCN that you noted in step 5.

        `sp_ctrl>` **reconcile queue** *queuename* **for** *datasource-datadest* **scn** *scn_number*

        Example: **reconcile queue SysA for o.oraA-o.oraA scn 0123456789**

    The reconcile process retains control of **sp_ctrl** until it is finished, and then the **sp_ctrl** prompt returns.

9. On the **target** system, log onto SQL*Plus as the Oracle user for SharePlex, and run the **cleanup.sql** script located in the **bin** sub-directory of the SharePlex product directory. This script truncates and updates the SharePlex tables, which are owned by the SharePlex user. If you are running multiple instances of **sp_cop** with multiple variable-data directories, there is a SharePlex Oracle user for each one. Make sure you run this script as the SharePlex user that owns the tables you want to restore. The script prompts you for the SharePlex user name and password.

    `SQL>` **@***/productdir*/**bin/cleanup.sql**

10. On the **target** system, disable or modify the following according to your replication strategy:

    - triggers
    - foreign key constraints
    - cascading delete constraints (or configure SharePlex to ignore them)
    - check constraints
    - scheduled jobs that perform DML

11. On the **source** system, reactivate the configuration file if you had to make any changes to it.

    ```
    sp_ctrl> activate config filename
    ```

12. On the **target** system, start the Post process. The two instances are now in synchronization, and SharePlex will continue replicating.

    ```
    sp_ctrl> start post
    ```

How to Restore Oracle Archive Logs

If you decide to restore the archive logs to enable SharePlex to resume capture and replication, use the following procedure to determine the required archive logs.

Perform the following steps to determine the required archive logs:

1. Determine the sequence number that Capture needs to resume processing from. Capture stops when it encounters a log wrap and prints a message to the Event Log (**event_log**) containing the redo log sequence number it needs. You also can find out this number by querying the SHAREPLEX_ACTID table and looking at the SEQNO column, as shown in the following example:

   ```
   SQL> select * from splex.shareplex_actid;

   ACTID    SEQNO    OFFSET     AB_      QUE_SEQ_NO_1  QUE_SEQ_   COMMAND
                                FLAG                   NO_2
   -----    ------   --------   ------   ------------  --------   ------------
                                --       -             ------
   14       114      9757200    0        672101000     0
   ```

2. Query the Oracle V$LOG_HISTORY table to find out when that sequence number was archived, then copy the logs from that point forward to the source system.

   ```
   SQL> select * from V$LOG_HISTORY;

   RECID    STAMP        THREAD#    SEQUENCE#    FIRST_        FIRST_TIM NEXT_
                                                 CHANGE#       CHANGE#
   -----    ------       --------   --------     -----------   --------------
                                                 --
   111      402941650    1          111          2729501       14-JUL-00 2729548
   112      402941737    1          112          2729548       14-JUL-00 2729633
   113      402941930    1          113          2729633       14-JUL-00 2781791
   114      402942019    1          114          2781791       14-JUL-00 2836155
   115      402942106    1          115          2836155       14-JUL-00 2890539
   ```

How to Release Semaphores after Process Failure

If database corruption or other system problem forced you to shut down SharePlex, verify that SharePlex released the semaphores and shared memory that it was using.

To verify and release semaphores:

1. Look for any SharePlex processes that did not shut down, and kill them.

   $ **ps -ef** | **grep sp_**

   $ **kill -9** *PID*

2. Change directories to the **rim** sub-directory of the SharePlex variable-data directory, then issue the **od -x** command for the **shmaddr.loc** and the **shstinfo.ipc** files.

   # od -x shmaddr.loc

   ```
   0000000 0000 00e1 ed40 0000 4400 9328 0080 0000
   0000020 0002 0021
   0000024
   ```

   # od -x shstinfo.ipc

   ```
   0000000 0000 00e0 ee90 0000 4100 9328 0010 0000
   0000020 0002 0020
   0000024
   ```

3. **Make a note of the following values:**

   - The first 32-bit word of each of the files above reveals the hexadecimal equivalent of the **ID** of the shared memory segment. Convert this value to decimal. For example, in the **shmaddr.loc** file shown in step 2, the first word is 0000 00e1, which equates to a decimal value of 225. In the **shstinfo.ipc** file, the first word is 0000 00e0, which equates to a decimal value of 224.
   - The third word of the **shmaddr.loc** and the **shstinfo.ipc** files reveals the hexadecimal equivalent of the **KEY** of the shared memory segment and the semaphore. (Each set has the same key value.) Do *not* convert this value to decimal. For example, in the **shmaddr.loc** file, the third word is 4400 9328. In the **shstinfo.ipc** file, the third word is 4100 9328.
   - The fifth word of each file is the **SEMAPHORE ID**. Convert this value to decimal. The semaphore IDs in the examples are hex 0002 0021 and 0020 0020, which in decimal are 131105 and 131104, respectively.

4. Issue the **ipcs -smaa** command to view all of the shared memory segments and semaphores. (Shared memory segments are listed first and are denoted with an “m.” Semaphores are denoted with an “s.”) The display looks similar to the following, but will be more extensive.

```
# ipcs -smaa

T         ID      KEY        MODE       OWNER GROUP  CREATOR CGROUP NATTCH    SEGSZ
CPID  LPID   ATIME    DTIME    CTIME
Shared Memory:
m         22  0x41009d0f --rw-r--r--  root spadmin   root spadmin  0    1048576
6517  6517  8:04:10  8:39:23 13:57:15
m         23  0x44009d0f --rw-r--r--  root spadmin   root spadmin  0    8388608
6517  6517  8:04:10  8:39:23 13:57:15
m        224  0x41009328 --rw-r--r--  root staff     root  staff   1    1048576
10030 13299 10:13:40 10:13:40  9:53:33
m        225  0x44009328 --rw-r--r--  root staff     root  staff   1    8388608
10030 13299 10:13:40 10:13:40  9:53:33

T       ID      KEY       MODE       OWNER    GROUP  CREATOR   CGROUP NSEMS   OTIME
CTIME
Semaphores:
s         30   0x41009d0f --ra-ra-ra-     root  spadmin     root  spadmin    12
8:39:19 13:57:15
s         31   0x44009d0f --ra-ra-ra-     root  spadmin     root  spadmin     2
8:39:19 13:57:15
s     131104   0x41009328 --ra-ra-ra-     root    staff     root    staff    12
```

5. Verify that the shared memory IDs from the **shmaddr.loc** and **shstinfo.ipc** are in the list and that the keys match.
6. For each shared memory segment, verify that the value in the **NATTCH** column is 0. This ensures that the SharePlex processes that you killed released their memory segments.
7. For the semaphores, verify that the semaphore IDs and keys match the file values.
8. As root, issue the **ipcrm -m** command for the ID values (224 and 225 in the examples) to remove the memory segments.

   # ipcrm -m 224

   # ipcrm -m 225

9. As root, issue the **ipcrm -s** command for the key values (131104 and 131105 in the examples) to remove the semaphores.

   # ipcrm -s 131104

   # ipcrm -s 131105

How to Resolve Disk Space Shortage

This topic helps you resolve disk space issues that can occur when something interferes with replication. See Solve Replication Problems on page 312 for possible causes.

How to conserve disk space on the target

SharePlex captures and processes data much faster than it posts it with SQL statements on the target system, so the target is where most disk problems can occur, assuming the network is operational and data is being sent from the source. If you think the post queue may exceed its disk space, there may be enough free space on the source system to store the data temporarily until the Post queue clears out.

To conserve the disk space on the target:

1. Stop the Import process.
2. Let the data accumulate on the source system until Post processes enough messages to clear the post queue.
3. Start Import.
4. Continue to stop and start Import until the amount of data accumulating in the post queue levels out.

When you implement this method, monitor the replication services and disk usage on the source system. On Unix and Linux systems, you can use the **sp_ps** script to monitor processes and the **sp_qstatmon** monitoring script to monitor the queues.

How to restore disk space

If a queue disk runs out of disk space, you may see messages similar to this in the Event Log:

```
11/22/07 14:14 System call error: No space left on device bu_wt.write [sp_
mport(que)/1937472]

11/22/07 14:14 System call error: No space left on device bu_rls.bu_wt [sp_
mport(que)/1937472]

11/22/07 14:14 Error: que_BUFWRTERR: Error writing buffer to file que_
writecommit(irvspxuz+P+o.a920a64z-o.a102a64z) [sp_mport(rim)/1937472]
11/22/07 14:14 Error: sp_mport: rim_writecommit failed 30 - exiting [sp_
mport/ 1937472]

11/22/07 14:14 Process exited sp_mport (from irvspxuz.domain.com queue
irvspxuz) [pid = 1937472] - exit(1)
```

If a queue disk is almost out of free space, you might be able to add disk space without the need to resynchronize the data.

To restore disk space:

1. Stop SharePlex on the affected system.
2. Add more disk space.
3. Start SharePlex.

4. View the Event Log and look for the messages "`queue recovery started`" and "`queue recovery complete`."
    - If both messages are there, SharePlex resumes processing where it stopped and the recovery succeeded. If your applications generate high volumes of transactions, there may be numerous backlogged messages in the queues. Depending on the nature of the transactions, how well the target database and the Post process are tuned, and your tolerance for latency, it might be more practical to resynchronize the data instead of waiting for replication to regain parity with transactional activity.
    - If one or more queues is corrupted, the Event Log records a message like this: `Bad header magic...` or `peekahead failure`. Or, you will see the message `queue recovery started`, but you will not see the `queue recovery complete` message that signifies successful queue recovery. In this case, you must restore replication an initial state.

To restore replication to an initial state:

1. Run **db_cleansp** to restore the variable-data directory and SharePlex tables. It must be run on all systems in the affected replicationconfiguration. See the utilities documentation in the SharePlex Reference Guide.
2. Synchronize the data using your method of choice, then reactivate the configuration. For more information, see Start Replication on your Production Systems on page 254.
3. You can prevent this problem from occurring again by using the SharePlex monitoring utilities to start unattended monitoring of key replication events, including queue volume alerts. For more information, see Monitor SharePlex on page 284.

How to find the ORACLE_SID and ORACLE_HOME

When setting up SharePlex to work with an Oracle database, you provide the ORACLE_SID and then SharePlex gets the ORACLE_HOME from the **oratab** file on Unix/Linux. Both values are stored in the SharePlex environment. SharePlex uses the Oracle libraries that are in the location specified with ORACLE_HOME.

To determine the ORACLE_SID and ORACLE_HOME being used by SharePlex:

Issue the **orainfo** command in **sp_ctrl**.

```
sp_ctrl (mysysl11:2101)> orainfo
Oracle instance #1:
     Oracle SID ora12
     Oracle HOME /oracle/products/12
     Oracle Version 12
Oracle instance #2:
     Oracle SID ora12
     Oracle HOME /oracle/products/12
     Oracle Version 12
```

To determine the default ORACLE_SID and ORACLE_HOME on UNIX and Linux:

On most Unix and Linux systems the **oratab** file is under **/etc/oratab**. On Oracle Solaris systems, it is under **/var/opt/oracle**, but sometimes there is an **oratab** file in the **/etc** directory as well.

The entry in the file looks like the following example:

```
qa12:/qa/oracle/ora12/app/oracle/product/12.0
```

In the example, `qa12` is the ORACLE_SID and `/qa/oracle/ora12/app/oracle/product/12.0` is the ORACLE_HOME.

19

Repair Out-of-sync Data

This chapter contains an overview of the SharePlex Compare and Repair feature. SharePlex provides this feature as built-in support for Oracle tables to help you maintain data that is synchronized between the source and target systems.

Contents

Overview of Compare and Repair

In addition to regularly monitoring the health and performance of replication, it is good practice to compare the source and target data on a regular schedule to ensure that all of the data is still synchronized. Post detects out-of-sync conditions for the rows that it is processing, but there can be hidden out-of-sync conditions. Examples of these are DML applied on the target or an incomplete backup restore. These conditions can go undetected until Post applies an operation that affects the out-of-sync row. The SharePlex Compare and Repair feature enables you to detect hidden out-of-sync conditions and then repair them.

NOTE: To understand how hidden out-of-sync conditions can occur, see Understand the Concept of Synchronization on page 35.

SharePlex provides the following commands for comparing and repairing out-of-sync data:

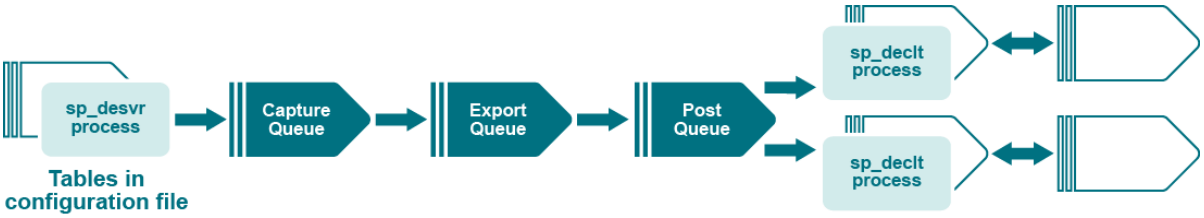
- **compare**: Compares an individual source table to its target table or compares a wildcarded set of tables in the same schema.
- **compare using**: Takes input from a file to compare some or all of the tables in the active replication configuration.
- **repair**: Repairs an individual target table or a wildcarded set of tables in the same schema.
- **repair using**: Takes input from a file to repair some or all of the tables in the active replication configuration.

Supported source and targets

Oracle to Oracle, PostgreSQL to PostgreSQL

Overview of the server and client processes

The compare and repair commands are always issued on the source system. The command spawns a server process on the source system and then sends a message through the SharePlex queues to spawn a client process on the target system.



The server and client processes then begin communication with each other. Depending on the syntax options included in the command, the processes may be multithreaded on the target. The two processes compare the source and target tables and then write the results to a log file.

.

How locks are managed

During a comparison, SharePlex obtains a brief exclusive lock on the source and target tables to get read consistency for row selection. This ensures the consistency of row data while SharePlex is processing it. After lock is released, with read consistent view the rows are read and sorted in an identical fashion on both the source and target. Next, a batch of rows is read, and a checksum is performed. If the checksums match, another batch of rows is processed the same way. If any checksums do not match, the processes determine which rows are out of synchronization, and then they create the SQL statements to repair them. The target table is locked during the repair process to prevent any other process from modifying its data.

If a WHERE clause, targetWHERE, or sourceWHERE options are provided, then only that set of rows matching the condition will be locked. (This applies only to PostgreSQL databases)

Before you Use Compare and Repair

Before running the compare or repair commands during Oracle to Oracle replication, review these guidelines.

- All of the SharePlex processes (Capture, Read, Export, Import, Post) must be running when you run a comparison or repair command.
- The tables that you want to compare or repair must be part of an active configuration file.
- Uncommitted transactions on a source table prevent the compare and repair processes from obtaining the brief locks they need to obtain read consistency. Make certain that all transactions are committed before you run a comparison or repair.
- If a table is large, it will probably need to be sorted in the TEMP tablespace. Before running the compare or repair commands, the TEMP tablespace may need to be made larger. The size depends on the setting of the SP_DEQ_THREADS parameter or the **threads** option within the command syntax, both of which controls the number of processing threads used by SharePlex on the target. Each thread processes a table. At the default of two threads, the size of the tablespace should be larger than the sum of the sizes of the two largest tables. If you set the number of threads higher, then increase the size of the tablespace to accommodate a proportionate number of the largest tables. However,
- The UNDO tablespace may also need to be increased. Based on transaction volume and the length of time it takes to compare the largest table, increase the size of the UNDO tablespace and increase the **undo_ retention** database parameter to avoid an ORA-1555 Snapshot too old error. Tables with LOBs take much longer to compare or repair than tables without them.

Before running the compare or repair commands during PostgreSQL to PostgreSQL replication, review these guidelines.

- All of the SharePlex processes (Capture, Read, Export, Import, Post) must be running when you run a comparison or repair command.
- The tables that you want to compare or repair must be part of an active configuration file.
- Uncommitted transactions on a source table prevent the compare and repair processes from obtaining the brief locks they need to obtain read consistency. Make certain that all transactions are committed before you run a comparison or repair.
- If the size of a table row (sum of data in each column) is large, it will likely require more memory, as the PostgreSQL database posts table data to the calling process that invokes the SELECT query. Before running the compare or repair commands, ensure that sufficient memory is available. The size depends on the setting of the SP_DEQ_THREADS parameter or the threads option within the command syntax, and the SP_DEQ_MALLOC value, which determines how many tables are processed at a time and the memory allocated for each thread. The value assigned to the SP_DEQ_MALLOC parameter is divided among the number of running threads.

Compare and repair parameters:

The following are commonly modified compare and repair parameters. Do not increase the values unless necessary. For details about these parameters, see their documentation in the SharePlex Reference Guide.

Parameter	Description
SP_DEQ_MALLOC	This parameter controls the fetch batch size. The batch size controls the number of rows that SharePlex selects at once for comparison. Larger batch sizes increase processing speed but require more memory. The value is divided equally by the number of compare threads to be used, and then the batch size is recalculated based on all column sizes added together.
SP_DEQ_PARRALLISM	This parameter manages the select statement Degree of Parallelism hint. The **parallelism** option of the command overrides this setting.
SP_DEQ_PART_TABLE_UPDATE	This parameter controls how the repair commands work on Oracle partitioned tables, depending on whether row movement is possible.
SP_DEQ_PG_DECLARE_FETCH_SIZE	This parameter determines whether the driver attempts to return a result set in a single fetch or across multiple fetches. This parameter is applicable only for PostgreSQL database.
SP_DEQ_READ_BUFFER_SIZE	This parameter controls the size of the buffer that holds fetched LONG and LOB data and can be adjusted based on available system memory.
SP_DEQ_ROW_LOCK_THRESHOLD	This parameter sets a threshold that controls whether SharePlex uses row-level or table-level locking when a **WHERE** option is used.
SP_DEQ_SKIP_LOB	This parameter determines whether or not LOBs are included in the compare/repair processing.

Parameter	Description
	• When the parameter is set to the default of 0, the compare processes include LOBs in their processing. • When the parameter is set to 1, only non-LOB columns are compared and repaired. If LOBs are not modified once inserted, you can speed up processing by setting this parameter to 1. Set this parameter on the **source** system.
SP_DEQ_TIMEOUT	This parameter sets a queue backlog threshold. High backlogs delay the establishment of a connection between the source and target compare/repair processes. If the backlog meets or exceeds this value, any compare or repair command that is issued on the source will exit and return an error. If this happens, consider running the compare or repair when the system is less busy.

How to Use the Repair and Compare Commands

The recommended procedure for maintaining synchronized data through the comparison and repair commands is to run the **compare** or **compare using** command first, then view the results with the **repair status** command. This command shows any rows that are out-of-sync and the possible cause. Unless the cause of the out-of-sync condition is corrected, replication will go out of synchronization again, even if you repair the rows this time. After the problem is fixed, issue the **repair** or **repair using** command.

You can run the **repair** or **repair using** command without doing a preliminary comparison. The command performs a comparison first, to identify the out-of-sync rows, and then it repairs those rows. However, the underlying cause of the out-of-sync condition must be corrected to prevent future out-of-sync conditions.

See for causes and solutions for out-of-sync conditions.

To view the status or results of a comparison, use the **compare status** command in **sp_ctrl**.

To view the status or results of a repair, use the **repair status** command in **sp_ctrl**.

When to run a repair

The best time to repair a target table depends on its size, the cause of the problem, the extent of out-of-sync rows, and how long you are willing to tolerate users being locked out. Before you initiate a repair, consider the following:

- Although the users of the tables are not usually affected by the brief locks that are applied when tables are compared, they are locked out of the target table for the duration of the repair process. For a small table, this might not be disruptive, but for a large table needing extensive repairs, the wait can be significant.
- Locks on a target table can reduce posting performance if Post must wait for the repair to finish before it can apply changes to that table and move on to other tables. This increases the latency of the target data and causes operations to accumulate in the post queue. If the objects that Post needs to change are different from those being repaired, the two processes run simultaneously.
- If you must repair a table immediately, but cannot tolerate locks or replication latency, you can use the **where** option to limit the repair to certain rows. An alternative is to use the **key** option, but this option may cause the repair to miss some out-of-sync rows.
- If the repair can wait, correct the cause of the problem immediately and then do the repair during non-peak hours.
- Replication latency can slow down the compare and repair processing. The message sent from the source to spawn the command processes on the target is sent through the queues along with regular replicated data. Delays caused by a data backlog will delay the spawn message and cause the process on the source to lose its read consistency, which results in errors. If possible, perform comparisons and repairs during off-peak hours.

How to run the compare and repair commands

To get additional information and syntax for the compare and repair commands, see the command documentation in the SharePlex Reference Guide.

20

Tune the Capture Process

This chapter contains instructions for improving the performance of the Capture process to prevent Capture from losing pace with the volume of redo that an Oracle source database generates.

Contents

Disable LOB Mapping

If you have PK/UK logging enabled on the source database (recommended to support more SharePlex features and faster processing), check the setting of the SP_OCT_ENABLE_LOBMAP parameter. This parameter controls whether or not SharePlex uses a LOB map when replicating tables that contain out-of-row LOB columns. The LOB map is used by the Capture process to map LOBIDs and rows when PK/UK logging is not enabled. LOB mapping is enabled by default. The SHAREPLEX_LOBMAP table stores these mappings.Transactions with numerous LOB operations can slow down Capture because it needs to maintain and refer to the mappings. If PK/UK logging is enabled on the database, you can disable LOB mapping by setting this parameter to 0.

To disable LOB mapping during active replication:

1. Run **sp_ctrl** on the source system.
2. Set SP_OCT_ENABLE_LOBMAP to 0.

   ```
   sp_ctrl> set param SP_OCT_ENABLE_LOBMAP 0
   ```

3. Stop Capture.

   ```
   sp_ctrl> stop capture
   ```

4. Truncate the SHAREPLEX_LOBMAP table.
5. Restart Capture.

   ```
   sp_ctrl> start capture
   ```

Tune Capture on Exadata

The Capture process can be configured to use multiple capture threads for faster performance on an Exadata system. Capture reads directly from the logs on the Exadata ASM disks.

The SP_OCT_ASM_MULTI_OCI parameter controls the number of threads that Capture uses to read the redo logs.

The value for this parameter must be set to at least 2 but no more than the number of disks in the redo log disk group.

A large number of threads is not required, and performance actually diminishes with too many threads. The more threads, the more memory Capture requires. Start with a small number of threads and monitor performance, then add threads if needed until you obtain an ideal balance between performance gain and memory usage.

To configure SharePlex for multi-threaded capture on Exadata:

1. Run **sp_ctrl**.
2. Set the SP_OCT_ASM_MULTI_OCI parameter to the number of threads that you want Capture to use.

   `sp_ctrl>` **set param SP_OCT_ASM_MULTI_OCI 3**
3. Restart Capture.

NOTE: Capture automatically adjusts its buffer size to the value of the AU_SIZE parameter that is set for the disk group where the logs reside. This is the recommended buffer size for best performance and should not be changed. The SP_OCT_ASM_MULTI_OCI_BLOCK_SIZE parameter can override the default behavior if necessary.

Tune Checkpointing

Capture checkpoints it state to disk on a regular basis to support recovery. This information includes the log and location within that log of the most recently processes data. In a database environment where there are frequent log switches, a switch can occur before SharePlex writes its checkpoint. You can use the `SP_OCT_CHECKPOINT_LOG` parameter to ensure that Capture issues a checkpoint before a log switch.

The checkpoint is triggered when Capture lags a specified number of logs behind Oracle. For example, with the default of 2, Capture does a checkpoint when it falls 2 or more logs behind Oracle.

The range of permissible values for this parameter is from 2 (the default) to a value equal to the number of logs you are using. A value of 0 disables this feature.

Add a Second Thread

You can set the `SP_OCT_OLOG_RDS_MINER` parameter to 1 to add a second thread to Capture. This thread can be used to address performance issues when Capture is lagging behind Oracle on a very busy system.

Due to the processing load incurred by using this thread, it is disabled by default. To enable it, set this parameter to 1.

NOTE: Enabling the SP_OCT_OLOG_RDS_MINER parameter is deprecated and no longer supported starting with Oracle 19c.

21

Tune the Post Process

This chapter contains instructions for improving the performance of the Post process. Because replicated data is applied through standard SQL mechanisms, the Post process provides the most potential for performance tuning.

Contents

Use Oracle INDEX Hints

Valid for: Oracle targets

When SharePlex performs UPDATEs and DELETEs on a target table, Oracle sometimes does not pick the most efficient index for SharePlex. Without the right index, the Post process slows down when multiple UPDATEs and DELETEs are performed. SharePlex enables you to make use of Oracle's INDEX hints to enforce the use of the correct index on target objects.

To use INDEX hints, use the **hints**.*SID* file, where *SID* is the ORACLE_SID of the target instance. When Post applies a SQL statement, it reads the hints file. If the file contains entries, Post reads the data into memory and then checks each UPDATE and DELETE statement that it processes. If any of those operations involve tables listed in the hints file, Post sends the hints to Oracle.

Use hints only for tables that need them. For example, if Post is doing full-table scans on tables where there are defined indexes, use hints only for those tables. The use of hints causes Post to read the **hints**.*SID* file for each operation on tables listed in the file. This can slow down processing if numerous tables are listed.

The default maximum number of hints (table/index pairs) is 100. You can adjust this value with the SP_OPO_HINTS_LIMIT parameter. See the SharePlex Reference Guide for more information.

Make certain all indexes are valid. Although SharePlex will use an invalid index as a hint, Oracle ignores invalid hints and returns no errors. SharePlex writes the following information to the **event_Log** if it detects abnormal conditions relating to the specified hints.

15050 – hint file not found

17000 – error opening hint file

15051 – missing column in the hint file (either table or index name)

15052 – syntax error for *tablename*

15053 – syntax error for *indexname*

15054 – source table's object_id not found in object cache

15055 – more than 20 valid entries were entered into the hints file

To use the hints file:

There is a blank **hints.***SID* file in the SharePlex variable-data directory on each system. Use the **hints.***SID* file that resides on the **target** system. If a hints file does not exist, create one in this directory and make certain to use the **hints.***SID* naming format.

1. Stop Post if it is running.
2. Open the file.
3. You can add comment lines anywhere in the file. Start a comment line with a pound symbol (#).
4. On a non-commented line, use the following template to specify a source table and the index that you want to use for that table. Put at least one space between the table name and the index name. Place each specification on a separate line.

"src_owner"."table"	*"tgt_owner"."index"*

Example

"scott"."emp" **"scott"."emp_index"**

Tune SQL Caching

SharePlex caches frequently-used SQL statements for reuse so that they do not have to be parsed and bound every time they recur. This is an adjustable feature of SharePlex that is named SQL Cache. You can tune this feature to maximize its advantages based on the amount of repetitive statements your application generates.

SQL Cache improves the performance of Post only if the same SQL statements are issued over and over again, with no variation except the data values. If that is not true of your environment, then SQL Cache adds unnecessary overhead to the Post process, and you should disable it.

Supported targets

All

Enable or disable SQL Cache

Control SQL Cache as follows:

Oracle

Parameter	Description
SP_OPO_ SQL_ CACHE_ DISABLE	Enables or disables SQL Cache. Enabled by default with a setting of 0. To disable SQL Cache set the parameter to 1. To disable SQL Cache only for batch operations set the parameter to 3, which reduces the amount of memory that Post uses.
SP_OPO_ MAX_CDA	Determines the number of active statements to cache per Post session. Post opens 50 cursors per session by default. You can increase or decrease this setting if needed. For more information, see Adjust Open Cursors on page 363.

Open Target

Parameter	Description
SP_OPX_SQL_ CACHE_DISABLE	Enables or disables SQL Cache. Enabled by default with a setting of 0. To disable SQL Cache set the parameter to 1.
Use the **target** command: **target r.***database* [**queue** *queuename*] **set resources max_active_ statements=***number_ of_active_statements*	Determines the number of active statements to cache per Post session. For Open Target databases, Post gets the number of allowed active statements from the ODBC driver. If that value is lower than the setting for **max_active_statements**, Post stops and returns an error. You can either disable the SQL Cache feature or reduce the value of **max_active_statements**.

Tune SQL Cache for best performance

Follow these steps to make certain that the number of active statements is optimal for the operations that are replicated.

1. Determine the hit ratio for cached statements by running **sp_ctrl** and issuing the **show post detail** command.
2. Look for the **SQL cache hit count** field. It shows the ratio of the total number of messages that are executed without parsing and binding divided by the total number of INSERT, UPDATE and DELETE operations. For example, a hit ratio of 36% indicates that Post is using cached statements 36 percent of the time.
3. View the hit ratio after several days of typical replication activity to gauge the ideal setting for the number of active statements. If the hit ratio is under 50 percent, increase the parameter value in a small increment of about 5 statements.
4. Monitor the hit ratio over the next few days. If the hit ratio increases, it means your applications are using all of the cursors allowed for active statements. Continue to increase the parameter value in small increments until the hit ratio remains constant.

Adjust Open Cursors

Valid for: Oracle targets

The value of the Oracle parameter OPEN_CURSORS needs to be set high enough to support the level of performance expected of the Post process. This parameter defines the maximum number of cursors that a process (such as Post) can open.

Internally, Post establishes its maximum total number of open cursors from the value of OPEN_CURSORS, minus the 10 required for routine calls. You view this value in the event_log. For the following example, OPEN_CURSORS is set to 512.

```
Notice: sp_opst_mt (for o.oracle-o.oracle queue oracle) Post will not open
more than 502 cursors (OPEN_CURSORS - 10).
```

Post maintains a record of the number of cursors it has open. If Post detects that it will exceed the maximum number of cursors, it closes the least-recently used cursor in the least-recently used session.

To avoid running out of cursors, the Post process queries the OPEN_CURSORS value when it starts. If the value is not high enough, Post writes the following warning to the event_log:

```
Warning: (sp_opst_mt for o.oracle-o.oracle queue oracle)Oracle parameter 'OPEN_
CURSORS' is < number. Check 'OPEN_CURSORS' setting.
```

The OPEN_CURSORS value can be modified or added if absent.

To view the OPEN_CURSORS value, query the database using the following SQL statement:

```
select value from v$parameter where name = 'open_cursors';
```

To estimate a value for OPEN_CURSORS that is high enough for the Post process:

1. Estimate the peak number of concurrent transactions (sessions) that will be expected for the target instance. Post opens a session on the target system for each one on the source system. You can get a good estimate of the number of transactions by issuing the show **post detail** command in **sp_ctrl** when production is at its maximum level. The **Number of Open Transactions** field in the display shows the number of concurrent transactions.
2. Use the following formulas to determine the correct setting for OPEN_CURSORS to support SharePlex (and other applications that may be accessing the target data).

   SQL Cache enabled (default): By default, Post needs to reserve 10 cursors for routine calls that are closed once they finish, plus a minimum of 7 cursors per transaction (the base minimum of 2 plus an additional 5). The formula is:

   10 + (*peak number of concurrent transactions* x 7) = *minimum open cursors needed*

   SQL Cache disabled: The Post process needs to reserve 10 cursors for routine calls that are closed once they finish, plus a minimum of 2 cursors per transaction. The formula is:

   10 + (*peak number of concurrent transactions* x 2) = *minimum open cursors needed*

Skip Large Maintenance Transactions

Valid for: Oracle targets

Large transactions that are applied by application patches or other internal Oracle operations can be omitted from replication if they are not relevant to the data needed by user applications. These operations can translate into thousands or millions of individual UPDATE or DELETE statements for SharePlex, all to be applied by Post. Such transactions can adversely affect Post performance and increase the latency between the source and target data that user applications need to perform their work. There may be reasons to prevent other DML operations from being posted to a target database.

There are two ways you can handle such transactions:

- Assuming there are no referential relationships between those operations and the user data, configure those operations to process through a dedicated named post queue. For more information, see Configure Named Post Queues on page 113.
- Configure Post to skip the operations, and then apply the SQL statement directly through Oracle. See the following instructions.

To skip maintenance DML:

1. On the source system, run the **create_ignore.sql** script from the **util** sub-directory in the SharePlex product directory. This script creates the SHAREPLEX_IGNORE_TRANS public procedure in the database. When executed at the start of the transaction, the procedure directs the Capture process to ignore the DML operations that occur until the transaction is committed or rolled back. Thus, the affected operations are not replicated. For more information about the script, its limitations, and how to run it, see **create_ignore.sql** in the SharePlex Reference Guide.
2. Edit your patch script to call SHAREPLEX_IGNORE_TRANS before UPDATE or DELETE operations. This allows SharePlex to ignore the transaction and not send it to the target. The script will also have to be run on the target to bring the database back into sync.

NOTE: Only DML operations are affected by the SHAREPLEX_IGNORE_TRANS procedure. It does not cause SharePlex to skip DDL operations, including TRUNCATE. DDL operations are implicitly committed by Oracle, so they render the procedure invalid.

Make Small Transactions Faster

Valid for: Oracle and Open Target (as indicated per feature)

You can improve the speed of Post when it is processing mostly small transactions, such as those most commonly found in OLTP.

There are two features you can use, depending on the supported database:

- Increase the level of concurrency
- Reduce the number of commits

Together these features are called Post Enhanced Performance, or PEP.

Increase the level of concurrency

Valid for Oracle, SQL Server, and PostgreSQL targets

The Transaction Concurrency feature configures a Post process to apply transactions in parallel to increase overall throughput. To use this feature, supplemental logging for primary and unique keys must be enabled on the source.

To enable Transaction Concurrency:

- For an Oracle target database, set the `SP_OPO_DEPENDENCY_CHECK` parameter to 1.
- For SQL Server and PostgreSQL, set the `SP_OPX_THREADS` parameter to 2 or greater.

NOTE: The use of Transaction Concurrency may reduce or eliminate the need to run multiple Post processes, but you can still benefit from that configuration because it eliminates a single point of failure. If a Post process fails, the other Post processes can continue, resulting in less recovery time, after the problem is resolved. The Transaction Concurrency feature can be used in a multi-Post configuration, so long as the rules for using multiple Post processes are followed (such as including all tables with referential integrity in the same process stream). For more information, see Configure Named Post Queues on page 113.

Reduce the number of commits

Valid for Oracle and all Open Target databases

The Commit Reduction feature of Post combines batches of small transactions into larger ones. One large transaction runs faster than multiple smaller ones by having fewer commits and acknowledgments to process.

Post skips the commits of small transactions until their combined size reaches the threshold specified by one of the following parameters:

- SP_OPO_COMMIT_REDUCE_MSGS (Oracle targets)
- SP_OPX_COMMIT_REDUCE_MSGS (Open Target)

The default batch transaction size is 100 messages. This value is an approximation. If the size of the last transaction in the batch exceeds the specified threshold, SharePlex waits for the remaining messages and the commit before applying the batch transaction to the target.

Commit reduction is enabled by default. To disable commit reduction, set this parameter to a value of 1.

Split a Large Transaction into a Smaller One

Valid for: Currently supported for JMS

You can configure Post to split a large transaction into a series of smaller ones. This option can work around resource limits that affect large transactions, such as the number of row locks permitted per transaction.

To split a large transaction into smaller ones:

Use the target command to set the **commit_frequency** parameter.

target r.*database* [**queue** *queuename*] **set resources commit_frequency=***number_of_operations*

This parameter specifies a maximum number of operations after which Post issues a commit. It can be any integer greater than 1.

Example:

target r.mydb queue q1 set resources commit_frequency=10000

Tune Queue Performance

You can tune the performance of Post by tuning the performance of the post queue.

Reduce queue contention

You can use the SharePlex queue contention reduction feature to ensure that shared memory is not swapped to disk when the post queue is becoming full. This feature is enabled by the `SP_IMP_QUEUE_PAUSE` parameter.

This parameter pauses the writing of data to the post queue when that queue contains the specified number of messages. Post stores queue messages in shared memory until it issues a checkpoint, after which it releases the data from memory.

If the post queue runs out of shared memory, the read and write functions will start incurring file IO to free up the memory buffers. By pausing the queue writing, this parameter helps Post maintain its performance by avoiding the need for disk storage and the resultant slowdown in IO.

Use the `SP_IMP_QUEUE_RESUME` parameter to set the number of messages at which Import resumes writing to the post queue. This parameter works in conjunction with SP_IMP_QUEUE_PAUSE. If the number of messages in the post queue is lower or equal to the value set with this parameter, Import resumes writing to the post queue.

To use this feature, both `SP_IMP_QUEUE_PAUSE` and `SP_IMP_QUEUE_RESUME` must be greater than zero, and `SP_IMP_QUEUE_PAUSE` must be greater than `SP_IMP_QUEUE_RESUME`.

Tune subqueue indexing

You can improve Post queue performance by enabling subqueue indexing to access the subqueue structures that represent a transaction session. A message "Subqueue index enabled *queuename*" is written to the Event Log for every Post queue for which this parameter is enabled.

To enable this feature, set the `SP_QUE_USE_SUBQUE_INDEX` parameter to 1. This parameter does not support VARRAYs. If you are replicating VARRAYs and this parameter is enabled, the parameter is ignored.

Tune Hash-based Horizontally Partitioned Replication

Hash-based horizontally partitioned replication uses a hash algorithm that is based on the rowid by default. You may be able to improve the processing of tables that use hash-based horizontally partitioned replication by switching the hash algorithm to one that is based on the block where the row resides.

Because changing the algorithm has the same effect as a routing change (the potential to switch partitions), you must reactivate the configuration file. The activation locks the tables that are affected by this change so that the hash change is applied when there are no open transactions. The locking eliminates the potential for out-of-sync conditions by preventing data that is processed under the new hashing algorithm from being posted before in-flight data that was processed under the old algorithm.

To switch to a block-based hash:

1. Set the `SP_OCF_HASH_BY_BLOCK` parameter to 1.
2. Reactivate the configuration file.

22

Recover Replication after Oracle Failover

This chapter contains instructions for recovering Oracle replication along with the database and applications during a failover in a high-availability environment. To support these procedures, SharePlex must be properly configured to support high availability. See Configure Replication to Maintain High Availability.

Contents

Recover Replication if the Primary System Fails

In an unplanned failure of the primary (source) machine, replicated data remaining in the SharePlex queues on that system will be unrecoverable as a result of buffering and the likelihood that the queues are corrupted. In a high-availability environment, you can move replication to the secondary (target) machine along with the database users to maintain data availability. When the primary system is restored, you can move users and replication back to that system with minimal downtime using a hot backup of the secondary instance.

For this procedure, you will activate a configuration file and then use the **reconcile** command to synchronize the results of the backup with ongoing replicated user changes after the copied instance is recovered.

Supported databases

Oracle database on Unix or Linux

Requirements

- SharePlex must be configured correctly to support high availability. For more information, see Configure Replication to Maintain High Availability on page 212.
- You must have a backup of the SharePlex replication environment.
- You must know how to run SharePlex. For more information, see Run SharePlex on page 41.
- You must understand the **activate config**, **reconcile**, and **delete queue** commands. See the SharePlex Reference Guide.

Procedure 1: Move replication to the secondary system

To move replication to the secondary system after an unplanned failure of the primary system:

1. Verify that Export is stopped on the secondary system.

   ```
   sp_ctrl> stop export
   ```

2. Use the **qstatus** command to view the post queue, and keep issuing this command until the number of *backlogged* messages is 0. **NOTE:** Do not wait for the actual *numberof messages* to be 0. If the primary system failed before the commit for some transactions were received, messages for these partial transactions will remain in the queue until cleared later in this procedure.

   ```
   sp_ctrl> qstatus
   ```

3. Run the script that grants INSERT, UPDATE and DELETE access to all users on the secondary system.
4. Run the script that enables triggers and constraints on the secondary system when users begin using this system.
5. Implement the failover procedure for relocating users to the secondary system, including starting the applications.
6. Move users to the secondary system and let them resume working, but do not start Export. Their transactions will now be accumulating in the export queue awaiting restoration of the source database.

NOTE: If started, Export will repeatedly try to connect to the primary system, wasting system resources.

7. Go to Procedure 2: Move replication to the restored primary system.

Procedure 2: Move replication to the restored primary system

This procedure moves users back to the primary machine after it is recovered from an unplanned failure. Follow each segment in the order presented.

Restore the replication environment on the primary system

To restore the replication environment on the primary system:

1. On the primary system, recover the SharePlex directories, system files and Oracle files from the backups and archives.
2. On the primary system, start **sp_cop** using the **-s** option to prevent the SharePlex processes (Capture, Read, Export, Import, Post) from starting.

   ```
   $ /productdir/bin/sp_cop -s &
   ```

3. On the primary system, start **sp_ctrl**.

4. On the primary system, deactivate the configuration file. When you copied the archived SharePlex variable-data directory to the primary system, you copied the configuration that was active before the system failed. This causes the Capture process to set the transaction number to "1" when replication from the primary system resumes.

   `sp_ctrl>` **deactivate config** *filename*

Purge the queues

To purge the queues:

1. Run **sp_ctrl** on the primary and secondary systems.
2. On the primary system, delete the capture queue.

   `sp_ctrl>` **delete capture queue for datasrc [on host]**

   Example: `sp_ctrl>` delete capture queue for o.oraA
3. On the primary system, delete the export queue.

   `sp_ctrl>` **delete export queue quename [on host]**

   Example: `sp_ctrl>` delete export queue sysA
4. On the secondary system, delete the post queue.

   `sp_ctrl>` **delete post queue quename for datasrc-datadst [cleartrans] [on host]**

   Example: `sp_ctrl>` delete post queue sysA for o.oraA-o.oraB

NOTE:

- You are issuing the **delete queue** command on the primary system because you restored the old capture and export queues when you restored the archived SharePlex directories.
- You are issuing the **delete queue** command on the secondary system because data remaining in the post queue cannot be posted. The primary system failed before Post received a COMMIT for remaining transactions. SharePlex will rebuild the queues when you reactivate the configuration and the two systems are reconciled.

Start replication from secondary to primary system

To start replication from secondary to primary system:

1. On the primary system, verify that all SharePlex processes are stopped.

   `sp_ctrl>` **status**
2. On the primary system, stop Post.

   `sp_ctrl>` **stop post**
3. On the secondary system, start Export. This establishes communication between the primary and secondary systems.:

   `sp_ctrl>` **start export**

Synchronize the source and target data

To synchronize the source and target data:

1. On the secondary system, run an Oracle hot backup.
2. When the hot backup is finished, switch log files on the secondary system and make a note of the highest archive-log sequence number.
3. On the primary system, recover the primary database from the backup by using the UNTIL CANCEL option in the RECOVER clause, and cancel the recovery after the log with the number you recorded has been fully applied.
4. Open the database with the RESETLOGS option.

   NOTE: This resets the sequences on the primary system to the top of the cache upon startup.

5. On the primary system, issue the **reconcile** command using the sequence number of the log that you noted previously. If you are using named post queues, issue the command for each one. If you do not know the queue names, issue the **qstatus** command first.

   ```
   sp_ctrl> qstatus
   ```

   ```
   sp_ctrl> reconcile queue queuename for datasource-datadest seq sequence_number
   ```

   Example: **reconcile queue SysB for o.oraA-o.oraA seq 1234**
6. On the primary system, disable triggers on the tables, or run the **sp_add_trigger.sql** utility script so that the triggers ignore the SharePlex user.
7. On the primary system, disable check constraints and scheduled jobs that perform DML.
8. On the primary system, log onto SQL*Plus as the SharePlex Oracle user and run the **cleanup.sql** utility from the **bin** sub-directory of the SharePlex product directory. This truncates the SharePlex tables and updates the SHAREPLEX_ACTID table.
9. On the secondary system, truncate the SHAREPLEX_TRANS table. This table contains transaction information that the Post process was using before the primary system failed, and therefore that information is obsolete. Truncating the table restores transaction consistency between the two systems.

Activate replication on the primary system

To activate replication on the primary system:

1. On the primary system, activate the configuration file.

   ```
   sp_ctrl> activate config filename
   ```

2. On the primary system, start Post.

   ```
   sp_ctrl> start post
   ```

Restore the object cache

To restore the object cache:

1. On the primary system, view the status of the SharePlex processes.

   ```
   sp_ctrl> status
   ```

   NOTE: This command shows that Post stopped due to an error, which occurred because the object cache is missing.

2. On the primary system, issue the **show log** command with the **filter** option, and filter on the keyword "**objcache**."

   ```
   sp_ctrl> show log filter=objcache
   ```

3. Look for a Post error message that refers to a file with a name similar to the following example:

   ```
   0x0a0100c5+PP+sys4+sp_opst_mt+o.quest-o.ov-objcache_sp_opst_mt.18
   ```

   You are looking for a string that contains the string “objcache_sp_opst_mt,” followed by a number. This is the object-cache file that the Post process needs. If you are using named post queues, there will be more than one error message, each referring to a different object-cache file but ending with the same number, such as the number .18 in the example.

4. Make a note of the full pathname of the Post object-cache file(s) named in the error message. The path will be the **state** directory in the SharePlex variable-data directory, for example:

   ```
   splex_vardir/state/0x0a0100c5+PP+sys4+sp_opst_mt+o.quest-o.ov-objcache_
   sp_opst_mt.18
   ```

5. On the secondary system, shut down SharePlex.

   ```
   sp_ctrl> shutdown
   ```

6. On the secondary system, change directories to the **state** sub-directory of the SharePlex variable-data directory and locate the Capture object-cache file. This file will have a name similar to the one in the following example.

   ```
   o.quest-objcache_sp_ocap.18
   ```

   IMPORTANT! If there are more than one of these files, use the one with the most recent number at the end of it — this number should match the number at the end of the Post object-cache file, such as .18 in the example.

7. Copy the Capture object-cache file to the primary system and rename it to the full pathname of the Post object-cache file that you noted previously.

8. On the primary system, start Post.

   ```
   sp_ctrl> start post
   ```

Switch users back to the primary system

To switch users back to the primary system:

1. On the secondary system, stop user access to the database.
2. On the primary system, view the post queue and continue to issue the command until the number of messages is 0.

   ```
   sp_ctrl> qstatus
   ```

3. On the secondary system, shut down the Oracle instance.

   ```
   svrmgr1> shutdown
   ```

4. On the secondary system, start the Oracle instance.

   ```
   svrmgr1> startup
   ```

   NOTE: This resets the sequence on the secondary system to the top of the cache to synchronize with the primary system.

5. On the primary system, enable database objects that were disabled.
6. On the secondary system, start SharePlex.
7. On the secondary system, stop Post.

   ```
   sp_ctrl> stop post
   ```

8. On the secondary system, disable triggers on the tables, or run the **sp_add_trigger.sql** utility script so that the triggers ignore the SharePlex user.
9. On the secondary system, disable check constraints and scheduled jobs that perform DML.
10. On the secondary system, start Post.

    ```
    sp_ctrl> start post
    ```

11. On the primary system, view the number of messages in the post queue, and keep checking until the number of messages is 0.

    ```
    sp_ctrl> qstatus
    ```

12. When the number of messages is 0, switch users back to the primary system.
13. On the secondary system, stop Export to prevent accidental changes made on that system from being replicated to the primary system.

    ```
    sp_ctrl> stop export
    ```

NOTE: The secondary system is now in a failover-ready state again, with:

- no users
- an active configuration
- disabled or modified triggers, constraints, and scheduled jobs
- a stopped Export process

Recover Replication if the Secondary Oracle Instance Fails

In an unplanned failure of the secondary (target) Oracle instance, the replication environment from that system to the primary system is corrupted. This procedure enables you to restore the replication configuration without affecting the database users on the primary system, and without the need to reactivate the configuration file on the primary system. Only the secondary configuration is affected.

This procedure cleans out the SharePlex queues and restores the target instance by means of a hot backup from the source system. You will use the **reconcile** command to synchronize the results of the backup with ongoing replicated user changes after the copied instance is recovered.

Supported databases

Oracle database on Unix or Linux

Requirements

- SharePlex must be configured correctly to support high availability. For more information, see Configure Replication to Maintain High Availability on page 212.
- You must know how to run SharePlex. For more information, see Run SharePlex on page 41.
- You must understand the **activate config**, **reconcile**, and **delete queue** commands. See the SharePlex Reference Guide.
- This procedure assumes that the secondary system itself is operational so that you can interact with SharePlex on the system.

Procedure

This procedure is divided into logical segments. Follow them in the order presented.

Purge the queues

To purge the queues:

1. On the secondary system, stop Post.

   ```
   sp_ctrl> stop post
   ```

2. On the secondary system, deactivate the configuration file.

   ```
   sp_ctrl> deactivate config filename
   ```

   NOTE: The deactivation causes the error "`Error in sp_cnc.`" in the Event Log. You may ignore it and continue with the procedure.

3. On the primary system, run **sp_ctrl**.

4. On the primary system, delete the post queue.

   `sp_ctrl>` **delete post queue quename for datasrc-datadst [cleartrans] [on host]**

   Example: `sp_ctrl>` **delete queue sysB:P for o.oraA-o.oraB**

   NOTE: You are deleting the queues on the primary system because there could be messages remaining from uncommitted transactions sent from the secondary instance before it failed.

5. On the primary system, truncate the SHAREPLEX_TRANS internal table in the SharePlex schema. This table contains transaction information that the Post process on that system was using before the secondary instance failed, and therefore the information is obsolete. Truncating the tables restores transaction consistency.
6. On the secondary system, run **sp_ctrl**.
7. On the secondary system, delete the capture queue.

   `sp_ctrl>` **delete capture queue for datasrc [on host]**

   Example: `sp_ctrl>` **delete queue o.oraB:C**

8. On the secondary system, delete the export queue.

   `sp_ctrl>` **delete export queue quename [on host]**

   Example: `sp_ctrl>` **delete queue sysB:X**

   NOTE: You are deleting the queues on the secondary system because the capture and export queues on that system still retain a record of the transactions that have already been processed.

Synchronize the data

To synchronize the data:

1. On the primary system, begin a hot backup of the primary Oracle instance.
2. On the primary system, switch log files.

   On-premises database:

   `svrmgr1>` **alter system switch logfile;**

   Amazon RDS database:

   Use Amazon RDS procedure **rdsadmin.rdsadmin_util.switch_logfile**.

3. Make a note of the highest sequence number of the archive logs.
4. On the secondary system, recover the secondary database from the hot backup using the UNTIL CANCEL option in the RECOVER clause. Cancel the recovery after Oracle has fully applied the log from the previous step.
5. On the secondary system, open the secondary database with the RESETLOGS option. This resets the sequence on the secondary system to the top of the cache upon startup
6. On the secondary system, run SQL*Plus as the SharePlex database user.

7. In SQL*Plus, run the **cleanup.sql** script from the **bin** subdirectory of the SharePlex product directory.
8. On the secondary system, issue the **reconcile** command using the sequence number of the log that you noted previously. If you are using named post queues, issue the command for each one. If you do not know the queue names, issue the **qstatus** command first.

   ```
   sp_ctrl> qstatus
   sp_ctrl> reconcile queue queuename for datasource-datadest seq sequence_number
   ```

   Example: **reconcile queue SysA for o.oraA-o.oraA seq 1234**
9. On the secondary system, disable triggers on the tables, or run the **sp_add_trigger.sql** utility script so that the triggers ignore the SharePlex user.
10. On the secondary system, disable check constraints and scheduled jobs that perform DML.
11. On the secondary system, after the **sp_ctrl** prompt returns, stop Export. This ensures that nothing accidentally gets replicated to the primary system when you activate the configuration on the secondary system.

    ```
    sp_ctrl> stop export
    ```

Start replication on the secondary system

To start replication on the secondary system:

1. On the secondary system, activate the configuration file.

   ```
   sp_ctrl> activate config filename
   ```
2. On the secondary system, start Post.

   ```
   sp_ctrl> start post
   ```
3. Use the **status** command to determine if any other SharePlex processes have a status of `Stopped due to Error` and start them.

   ```
   sp_ctrl> status
   sp_ctrl> start process
   ```

NOTE: The secondary system is now prepared for future failover.

Move Replication during Planned Failover and Failback

In a planned failover of database activity to a secondary Oracle instance, you can quickly move SharePlex to the secondary system. While users continue their transactions on that system, SharePlex captures their changes and stores them until the primary system is back online and activity is moved back to that system.

Supported databases

Oracle database on Unix or Linux

Requirements

- SharePlex must be configured correctly to support high availability. For more information, see Configure Replication to Maintain High Availability on page 212.
- There must be enough disk space where the queues reside on the secondary system to contain the data that accumulates while users are working on that system.
- You must know how to run SharePlex. For more information, see Run SharePlex on page 41.
- You must understand the SharePlex **flush** command. For more information, see the SharePlex Reference Guide.

Procedure

This procedure is divided into logical segments. Follow them in the order presented. Do not shut down the primary instance until prompted in the procedure.

Switch users to the secondary system

To switch users to the secondary system:

1. On the primary system, stop user access to the primary instance.
2. On the primary system, flush the data in the queues to the secondary system. This command stops Post on the secondary system and places a marker in the data stream to establish a synchronization point between the primary and secondary data.

   ```
   sp_ctrl> flush datasource
   ```

   where: *datasource* is the datasource specification of the primary Oracle instance, for example **o.OraA**.
3. On the secondary system, verify that Post stopped. (Continue to issue this command until it shows that Post stopped.)

   ```
   sp_ctrl> status
   ```

4. On the primary system, verify that there are no messages in the capture and export queues. The **Number of Messages** and the **Backlog (messages)** fields must be 0.

   ```
   sp_ctrl> qstatus
   ```

5. On the secondary system, verify that there are no messages in the post queue. The **Number of Messages** and the **Backlog (messages)** fields must be 0.

   ```
   sp_ctrl> qstatus
   ```

6. On the primary system, shut down SharePlex.
7. On the primary system, shut down the Oracle instance with the **abort** option. Do not use the **immediate** option.

   ```
   svrmgr1> shutdown abort
   ```

   NOTE: This resets the sequence on the primary system to the top of the cache when the database starts.

8. On the secondary system, verify that Export is stopped. This prevents user changes from being replicated to the primary system until it is back online and SharePlex is ready to receive them. Stop Export if needed.

   ```
   sp_ctrl> status
   sp_ctrl> stop export
   ```

9. On the secondary system, run the script that grants INSERT, UPDATE and DELETE access to all users.
10. On the secondary system, run the script that enables triggers and constraints on the secondary instance.
11. Run your failover procedure for relocating users to the secondary system, including starting the applications and starting jobs that were running on the primary system.
12. Move the users to the secondary system to resume working, but do not start the Export process.

Switch users back to the primary system

To switch users back to the primary system:

1. On the primary system, open the Oracle instance. The sequence on this system should now be at the top of the cache.
2. On the primary system, disable triggers on the tables, or run the **sp_add_trigger.sql** utility script so that the triggers ignore the SharePlex user.
3. On the primary system, disable check constraints and scheduled jobs that perform DML.
4. On the primary system, start SharePlex.
5. On the secondary system, start Export so that SharePlex sends the accumulated replicated data to the primary system.

   ```
   sp_ctrl> start export
   ```

   NOTE: SharePlex passes any sequence updates from the secondary system back to the primary system when Export starts.

6. On the primary system, stop Export.

   ```
   sp_ctrl> stop export
   ```

7. On the primary system, allow Post to process the message backlog that was sent from the primary system.
8. On the secondary system, stop user access to the Oracle instance.
9. On the secondary system, flush the data in the queues to the primary system.

   ```
   sp_ctrl> flush datasource
   ```

   where: *datasource* is the datasource specification of the secondary Oracle instance, for example **o.OraB**.

10. On the primary system, verify that Post stopped. (Continue to issue this command until it shows that Post stopped.)

    ```
    sp_ctrl> status
    ```

11. On the secondary system, verify that there are no messages in the capture and export queues. The **Number of Messages** and the **Backlog (messages)** fields must be 0.

    ```
    sp_ctrl> qstatus
    ```

12. On the primary system, verify that there are no messages in the post queue. The **Number of Messages** and the **Backlog (messages)** fields must be 0.

    ```
    sp_ctrl> qstatus
    ```

13. On the secondary system, shut down SharePlex.

    ```
    sp_ctrl> shutdown
    ```

14. On the secondary system, shut down the Oracle instance with the **abort** option. Do not use the **immediate** option.

    ```
    svrmgr1> shutdown abort
    ```

    NOTE: This resets the sequence on the secondary system to the top of the cache when the database starts.

15. On the secondary system, start the Oracle instance.

    ```
    svrmgr1> startup
    ```

    NOTE: The sequence on the secondary system is now at the top of the cache. When the next value is selected on the primary system, a new cache is acquired and is replicated to the secondary system. Now, the primary system is at the start of a cache, and the secondary system is at the top of a cache.

16. On the primary system, run the script that grants INSERT, UPDATE and DELETE access to all users.
17. On the primary system, run the script that enables triggers and constraints on the primary system when users begin using this system.
18. Run your failover procedure for moving users back to the primary system, including starting the applications and starting jobs that were running on the secondary system.
19. Switch the users to the primary system to resume working, but do not start the Export process. This prevents replicated data from being sent to the secondary system until SharePlex is ready to receive it there.

Resume replication to maintain the secondary instance

To resume replication to maintain the secondary instance:

1. On the secondary system, disable triggers on the tables, or run the **sp_add_trigger.sql** utility script so that the triggers ignore the SharePlex user.
2. On the secondary system, disable check constraints and scheduled jobs that perform DML.
3. On the secondary system, start SharePlex.
4. On the secondary system, stop Export. This prevents any accidental DML on that system from being replicated to the primary system.

   ```
   sp_ctrl> stop export
   ```

5. On the primary system, start export.

   ```
   sp_ctrl> start export
   ```

6. On the secondary system, start Post.

   ```
   sp_ctrl> start post
   ```

   Replication from the primary instance to the secondary instance is now active to keep the two databases synchronized and ready for future failover when needed.

Resume Replication after Failure and Recovery

The procedure is typically used in these situations:

- When the source system fails and replication must be switched to a standby database system
- When replication must be positioned back in time to re-read old archive logs.

Requirements to support SharePlex replication recovery

To resume replication when the source, target or both have failed, there must be the following in place *at the onset of replication*:

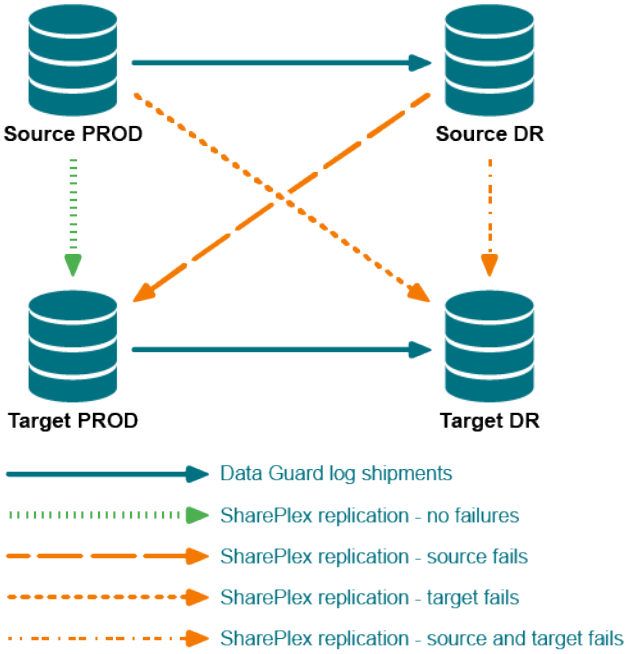
- A disaster recovery (DR) solution that provides a physically identical copy of the *production source instance* and another physical copy of the *production target instance*. Methods such as Oracle Data Guard or disk mirroring, tape backups and other methods support this requirement.
- The SP_OPO_UPDATE_SCN parameter must be set to a value of 1. This parameter directs SharePlex to keep a record of the SCNs of the transactions that it processes. When you set this parameter to 1, it also disables the Post Enhanced Performance feature.

Overview of initial setup

The following diagram depicts a DR configuration at the onset of replication. There is a source production instance and a mirrored source DR instance that is kept current by Oracle Data Guard. Similarly, there is a production target instance and a mirrored DR target instance that is kept current by Oracle Data Guard.

- The solid (blue) lines represent the Oracle Data Guard DR deployment.
- The dotted (bright green) line between the production source instance and the production target instance represent SharePlex replication under normal operating circumstances.
- The dashed lines (red, orange or aqua) show possible replication recovery paths if the source, target or both fail.

Figure 2: DR configuration at the onset of replication



Example failure/recovery scenario

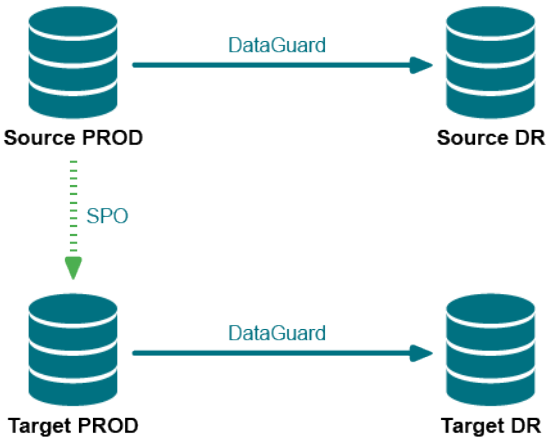
This example illustrates one of the potential failure/recovery scenarios, in this case where the production target instance fails. The recovery path is shown as the diagonal, orange dotted line in the DR configuration at the onset of replication diagram.

Normal replication

The following diagram illustrates the configuration and the names that are used in this example:

- The production source is named **Source PROD** and the DR source is named **Source DR**.
- The production target is named **Target PROD** and the DR target is named **Target DR.**
- SharePlex (SPO in the diagram) replicates from **Source PROD** to **Target PROD**.

Figure 3: Normal replication and mirroring configuration

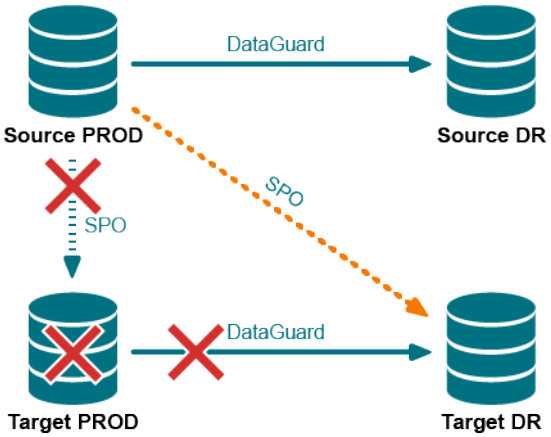


Production target fails

The **Target PROD** target fails, as represented by the red X across it in the following diagram. SharePlex can no longer replicate to **Target PROD**, as represented by the red X over the original replication data stream.

Because **Target PROD** is offline, Oracle Data Guard can no longer keep **Target DR** up to date. However, SharePlex can. SharePlex (SPO in the diagram) is able to resume replication from **Source PROD** to **Target DR**, thus resuming data availability.

Figure 4: Failure and recovery by SharePlex (SPO)



Resume replication after failover

In this procedure you will do the following to direct SharePlex to recover replication:

- Direct SharePlex to capture the correct Oracle SCN of the last committed transaction processed by each post queue.
- Direct SharePlex, through the **reconcile** command, to discard all transactions that were committed to the target before the failure, so that SharePlex resumes replication at the correct point in the data stream.

NOTE: This procedure requires the following:

- The source instance is recovered to a later point in time than the target instance; otherwise, this method will not work.
- The SP_OPO_UPDATE_SCN parameter is set to 1.

To resume replication:

NOTE: In these instructions, the *source* and *target* systems are whichever source and target are operational after the failover.

1. Shut down SharePlex on the source system, if it is still running.

   ```
   sp_ctrl>  shutdown
   ```

2. On the target, start **sp_cop** if it is not running already.

   ```
   $ /productdir/bin/sp_cop &
   ```

3. On the target, use the qstatus command to make certain that all of the message in the queues are posted to the target database. The command output should show 0 backlog in the post queue.

   ```
   sp_ctrl>  qstatus
   ```

4. From the command line of the target, run the **show_scn** utility from the **bin** subdirectory of the SharePlex product directory. For *ORACLE_SID* use the ORACLE_SID of the **target** database.

   ```
   $ /productdir/bin/show_scn  ORACLE_SID
   ```

5. Keep the output of the **show_scn** utility open. The output displays the complete **reconcile** command that you will use for each of your post queues to reposition Post to the correct transaction for recovery. It also shows the SCN to which you will activate the configuration later in these steps.

6. Shut down **sp_cop** on the source and target.

   ```
   sp_ctrl>  shutdown
   ```

7. Run **ora_cleansp** on the source and target to clean out the queues.

   ```
   $ /productdir/bin/ora_cleansp
   ```

8. Start **sp_cop** on the source and target.

   ```
   $ /productdir/bin/sp_cop &
   ```

9. On the target, stop Post.

   ```
   sp_ctrl> stop post
   ```

10. On the source, issue the **activate config** command with the **scn** option to activate the configuration. For *scn_value*, use the value that is shown in the output of the **show_scn** utility on the line that states `On source activate to scn=`*nnnnnnn*.

    ```
    sp_ctrl> activate config configname scn=scn_value
    ```

    Example:

    ```
    sp_ctrl> activate config myconfig scn=510012416
    ```

11. On the target, copy the first **reconcile** command from the **show_scn** output and then execute it in **sp_ctrl**. Then do the same for the second **reconcile** command, and work your way down the list.

    Example:

    ```
    sp_ctrl> reconcile queue spx11 for o.ora112-o.ora112 scn 235690
    sp_ctrl> reconcile queue pq1 for o.ora112-o.ora112 scn 132436
    sp_ctrl> reconcile queue pq2 for o.ora112-o.ora112 scn 246843
    sp_ctrl> reconcile queue pq3 for o.ora112-o.ora112 scn 123457
    ```

    The **reconcile** command may seem stalled until new data comes in. However, the command is working.

12. On the target, start Post.

    ```
    sp_ctrl> start post
    ```

23

Make Changes to an Active Replication Environment

This chapter contains instructions for making datbase changes, or performing system and software maintenance, on systems where SharePlex replication is active.

Contents

Change an Active Configuration File

Many procedures that change an element of the replication environment will also involve changing the SharePlex configuration file. The recommended method to modify an active configuration file is to make a copy of the file first using a new file name. For example, append the date to the file name. By copying the file first, you preserve the original file in case it is needed again or you encounter a problem with the new file. By copying the file first, you also can control when to activate the new configuration file.

In most cases, activation of an edited configuration is less time-consuming than the original activation *if you do not deactivate the original configuration*. The activation of the new configuration automatically deactivates the original configuration, and SharePlex only needs to analyze the new, changed, and removed objects. If you deactivate the original configuration before you activate a new one, SharePlex re-analyzes all of the objects.

To change a configuration file, see Add or Change Objects in an Active Configuration on page 387.

Add or Change Objects in an Active Configuration

This section provides instructions for adding a supported object to replication, or changing the specifications of an object, while replication is active.

Supported databases

Oracle source

All targets

Oracle procedure

NOTE: To add sequences to replication, see Add Oracle Sequences to an Active Replication Configuration on page 390.

If you are using wildcards and an object that you are adding satisfies the wildcard specification, it is not necessary to add the object to the configuration file if the source is Oracle. Any new objects that match the wildcard criteria are automatically added into replication. Only add objects that must be explicitly stated by name.

IMPORTANT! Do not deactivate the original configuration.

1. If adding new tables, add them to the source and target (populated in both places, if applicable) to establish a synchronized initial state. Do not allow transactional access to the source table yet.
2. In **sp_ctrl**, issue the **copy config** command to make a copy of the active configuration file.

   ```
   sp_ctrl> copy config filename to newname
   ```

   Where: *filename* is the name of the active file and *newname* is the name of the new one.
3. Issue the **edit config** command to open the new configuration file in the default text editor.

   ```
   sp_ctrl> edit config newname
   ```
4. Add the entries for the new tables or change existing entries.

   NOTE: To change partitioned replication, see Change Partitioned Replication on page 388.
5. Save the configuration file.
6. Activate the new configuration. This deactivates the original configuration. Only the new or changed tables are activated, so the activation should not be as long as the initial activation.

   ```
   sp_ctrl> activate config newname
   ```
7. Allow access to the newly added tables.

Change Partitioned Replication

You can change a horizontally partitioned or vertically partitioned replication configuration while replication is active. Both of these procedures require the reactivation of the active replication configuration, but SharePlex only locks tables that are associated with those changes.

NOTE: To learn more about changing a configuration file, review Change an Active Configuration File.

For more information about partitioned replication, see Configure Partitioned Replication.

Supported databases

Oracle

All targets

To change horizontally partitioned replication:

1. Run **sp_ctrl**.
2. Issue one of the following commands to change the partition or partition scheme. For syntax and other information, see the alphabetical command listings in the SharePlex Reference Guide.

Command	Auth. level	Description
add partition	**2**	Creates partition schemes and row partitions.
drop partition	**2**	Removes a row partition from a partition scheme.
drop partition scheme	**2**	Removes a partition scheme.
modify partition	**2**	Modifies a row partition of a partition scheme.

3. If you dropped a partition scheme:
   a. Copy (but **do not deactivate**) the active configuration file to a new file.

      `sp_ctrl>` **copy config** *filename* **to** *newname*

   b. Edit the copy to remove or change the routing map where the partition scheme was specified.

      `sp_ctrl>` **edit config** *filename*

4. Activate the new configuration file.

   `sp_ctrl>` **activate config** *filename*

To change vertically partitioned replication:

1. Make a copy of (but **do not deactivate**) the active configuration file.

   ```
   sp_ctrl> copy config filename to newname
   ```

2. Edit the copy to change the appropriate column partition.

   ```
   sp_ctrl> edit config filename
   ```

3. Activate the new configuration file.

   ```
   sp_ctrl> activate config filename
   ```

Add Oracle Sequences to an Active Replication Configuration

The procedure to use to add a sequence to an active configuration file depends on whether or not you can stop user access to the objects that use the sequence. If the sequences are used to populate a column in a table, you may not be able to stop user access.

Review the following procedures to determine which one will work best in your environment:

Enable auto-add of sequences

Add sequences if auto-add is not enabled

Supported databases

Oracle source and target

Enable auto-add of sequences

You can configure SharePlex to add sequences to replication automatically if their names satisfy a wildcard in the configuration file. For more information, see Control Oracle DDL Replication on page 215.

Add sequences if auto-add is not enabled

The following procedures apply if the auto-add feature for sequences is not enabled.

Add a sequence if the sequence does not populate a column

1. Stop user activity to the objects on the source system.
2. In **sp_ctrl**, issue the **copy config** command to make a copy of the active configuration file.

   ```
   sp_ctrl> copy config filename to newname
   ```

   Where: *filename* is the name of the active file and *newname* is the name of the new one.
3. Issue the **edit config** command to open the new configuration file in the default text editor.

   ```
   sp_ctrl> edit config newname
   ```
4. Add the new sequences to the configuration file.
5. Save and close the file.
6. Create the target sequence on the target system. To ensure uniqueness on the target system, the start value of the target sequence must be larger than the start value of the source sequence. Use the following formula to determine the target START_WITH value:

   source_INCREMENT_BY_value = START_WITH_value

7. Activate the new configuration. This deactivates the original configuration.

   ```
   sp_ctrl> activate config newname
   ```

8. Allow users to access the objects.

Add a sequence if the sequence populates a column

1. In **sp_ctrl**, issue the **copy config** command to make a copy of the active configuration file.

   ```
   sp_ctrl> copy config filename to newname
   ```

   Where: *filename* is the name of the active file and *newname* is the name of the new one.

2. Issue the **edit config** command to open the new configuration file in the default text editor.

   ```
   sp_ctrl> edit config newname
   ```

3. Add the new sequences to the configuration file.
4. Save and close the file.
5. Activate the new configuration. This deactivates the original configuration.

   ```
   sp_ctrl> activate config newname
   ```

6. On the source system, flush the data from source system to the target system. This command stops Post and places a marker in the data stream that establishes a synchronization point between source and target data.

   ```
   sp_ctrl> flush datasource
   ```

   Where: *datasource* is **o.***ORACLE_SID* of the source instance — for example **o.oraA**.

7. After Post stops, issue the following Oracle command on the target system to find the last known value of the sequence. Make a record of this value.

   select max(*column_name*) **= last known value**

8. Determine the value of the following equation.

   source_INCREMENT_BY_value x *source_CACHE_value*

   For example, if the source sequence is incremented by 2 and the cache size is 10, the value would be 20.

9. Starting with the value that you recorded for the **select max** (*column_name*) command, determine the next highest multiple of (*source_INCREMENT_BY_value* x *source_CACHE_value*).

   Example:

   INCREMENT_BY = 2

   CACHE = 10

   select max(*column_name*) = 24

   Next highest multiple of (2 x 10) after 24 = 40.

10. To the value obtained in the previous step, add another multiple of (*source_INCREMENT_BY_value* x *source_CACHE_value*). The result determines the START WITH value of the target sequence. For example, in the previous equation the START WITH value would be: 40 + (2 x 10) = 60.
11. Create the target sequence with the START WITH value that you calculated.
12. On the target, start Post.

    ```
    sp_ctrl> start post
    ```

    SharePlex will continue replicating the data, while keeping the target sequence at least one multiple of (*source_INCREMENT_BY_value* x *source_CACHE_value*) ahead of the source sequence.

IMPORTANT! Sequences continue to be incremented even when a transaction is rolled back. If numerous rollbacks are issued for a source table that uses a replicated sequence, it causes the sequence values to increase without actually being used in columns in the table. As a result, when Post applies the next valid operation, the sequence value on the target system could be less than the value in the replicated row. When there are numerous rollbacks, view the target table regularly to ensure that the current value of the target sequence remains greater than the maximum value in the table. If the current value of the target sequence is less than the maximum value in the table, repeat the preceding procedure to re-establish the sequence relationships.

Remove Source Objects from Replication

To remove source objects from replication, the configuration must be reactivated.

NOTE: (Oracle only) Objects being removed are locked when the configuration is activated, but only those objects are locked, so the activation is less time-consuming that the original activation.

You can prevent posting to a table without removing it from the configuration file. You may need to do this if, for example, there is data corruption and you do not want DML or DDL operations to be applied to that table. To prevent posting to a table, use the SP_OPO_DISABLE_OBJECT_NUM parameter. For more information about this parameter, see the SharePlex Reference Guide.

Supported databases

All databases supported by SharePlex

Procedure

To remove the source objects from replication:

1. In **sp_ctrl**, issue the **copy config** command to make a copy of the active configuration file.

   ```
   sp_ctrl> copy config filename to newname
   ```

   Where: *filename* is the name of the active file and *newname* is the name of the new one.

2. Issue the **edit config** command to open the new configuration file in the default text editor.

   ```
   sp_ctrl> edit config newname
   ```

3. In the new configuration file, delete the entries for the objects that you want to remove from replication. If the object that you want to remove from replication satisfies a wildcard, use the **not** notation to exclude the object. For more information, see Use Wildcards to Specify Multiple Objects on page 84.
4. Save and close the file.
5. Activate the new configuration. This deactivates the original configuration.

   ```
   sp_ctrl> activate config newname
   ```

6. Allow users to access the removed objects.

Make DDL Changes in an Active Replication Configuration

This procedure applies to DDL changes that are *not* of a type that is supported by SharePlex. DDL that is supported by SharePlex can be applied to the source database without reactivating the configuration file or stopping user access to objects, assuming the applicable SharePlex parameters are set correctly. Supported DDL is replicated by SharePlex to the target, where it is applied by Post. For a list of supported DDL operations and required parameters, see the SharePlex Release Notes.

Use this procedure to apply DDL that is not of a type that is supported by SharePlex. The DDL must be applied outside SharePlex on both the source and target systems. This procedure requires stopping access to the objects in the configuration file and a reactivation of the configuration file to update the internal tables. However, only the changed objects are analyzed, so the activation time will be shorter than the time required for a full activation.

Supported databases

Oracle

Requirements

- You must know how to run SharePlex. For more information, see Run SharePlex on page 41.
- You must understand how to activate a configuration file with the **activate config** command.
- You must understand the SharePlex **flush** command. For more information, see the SharePlex Reference Guide.

Procedure

1. On the source system, stop access to the source objects (on all systems if using peer-to-peer replication).
2. On the source system (trusted source in peer-to-peer), flush the data from the source system to the target systems. This command stops the Post process and places a marker in the data stream that establishes a synchronization point between the source and target data.

   ```
   sp_ctrl> flush datasource
   ```

   where: *datasource* is the database specification of the source instance, for example **o.oraA**.
3. On the target system (all secondary systems in peer-to-peer) verify that the number of messages in the post queue is 0 on each system and that Post stopped.

   ```
   sp_ctrl> lstatus
   ```
4. On the source system, make the DDL changes.
5. On the source system, reactivate the configuration file.

   ```
   sp_ctrl> activate config filename
   ```
6. On the source system, allow user activity to resume. Their replicated changes will accumulate in the post queue.

7. On the target system, make the corresponding DDL changes.
8. [High availability and peer-to-peer replication only] On the secondary systems, reactivate the configuration file.

   ```
   sp_ctrl> activate config filename
   ```

9. On the target systems, start Post.

   ```
   sp_ctrl> start post
   ```

SharePlex resumes replication from the last stop point and the data remains synchronized.

Make Oracle Changes that Affect Replication

This topic helps you make common changes to the Oracle environment while replication is active.

Supported databases

Oracle on Linux and UNIX

Move the location of ORACLE_HOME

If you change the ORACLE_HOME, you need to relink SharePlex to the Oracle libraries.

Perform the following steps to relink SharePlex to the Oracle libraries:

1. Shut down SharePlex.

   ```
   sp_ctrl> shutdown
   ```

2. Move the ORACLE_HOME.
3. Edit the **oratab** file to point to the new ORACLE_HOME.
4. Edit the **connections.yaml** file to point to the new ORACLE_HOME. This file is in the **data** subdirectory of the SharePlex var directory.
5. Start SharePlex.

Change the target ORACLE_SID

1. On the source system, run **sp_ctrl**.
2. On the source system, copy the active configuration file to a new name, but do not deactivate it.

   ```
   sp_ctrl> copy config filename to newname
   ```

3. On the source system, open the new configuration file.

   ```
   sp_ctrl> edit config filename
   ```

4. Change the ORACLE_SID to the new one in all of the routing maps that include this target database and target system.
5. Save and close the configuration file, but do not activate it.
6. On the source system, stop user access to the objects involved in replication.
7. On the source system, flush the data in the queues to the target. This stops the Post process and establishes a synchronization point between the source and target databases.

   ```
   sp_ctrl> flush datasource
   ```

   where: *datasource* is the database indicator of the source instance, for example **o.oraA.**

8. On the source system, activate the new configuration file. This will deactivate the original configuration file.

   ```
   sp_ctrl> activate config filename
   ```

   NOTE: The activation will be brief because SharePlex does not need to analyze the tables.

9. On the source system, allow users to access the objects involved in replication.
10. On the target system, verify that Post stopped. If Post is not stopped, continue to issue the command until it shows that Post stopped.

    ```
    sp_ctrl> status
    ```

11. On the target system, shut down the database and then rename the ORACLE_SID.
12. On the target system, start Post.

    ```
    sp_ctrl> start post
    ```

Change the SharePlex Database Account

You can change the user name (schema or database), the password, or both in the SharePlex database account. The database account was established during the installation of SharePlex. These procedures guide you through the process in the correct order to maintain an active replication configuration.

Supported databases

All SharePlex-supported databases

Procedure

This procedure changes the user account name and/or password of the SharePlex user account in a database. This user account is the one that the SharePlex processes use to connect to the database when performing replication tasks.

IMPORTANT! If using multiple variable-data directories, you must run this procedure for each one that you want to change.

1. (Unix and Linux only) If you are using multiple variable-data directories, export the environment variable that points to the variable-data directory for the SharePlex instance for which you are changing the account name or password.

   ksh shell:

   export SP_SYS_VARDIR=*/full_path_of_variable-data_directory*

   csh shell:

   setenv SP_SYS_VARDIR=*/full_path_of_variable-data_directory*

2. Run **sp_ctrl** on the system where you are changing the password.
3. Stop the SharePlex processes on the system where you are changing the account information. For example, if you are changing the SharePlex account in the source database, stop Capture and Read. If you are changing the account in the target database, stop Post.

   `sp_ctrl>` **stop** *service*

4. Verify that all SharePlex replication processes for this instance of SharePlex are stopped.

   `sp_ctrl>` **status**

5. Log in to the database as a DBA user and change the SharePlex account name and/or password to the new ones. **Important!** Do not delete the SharePlex objects!

6. If you changed the account name, copy all of the SharePlex database objects from the old account to the new one.

   NOTE: Keep the old account and SharePlex objects as backup until you are certain replication resumes properly.

7. In **sp_crtl**, issue the following command to change the account name and/or password in the SharePlex internal records.

   To change the user account:

   ```
   sp_ctrl>  connection {o.SID | r.database} set user=username
   ```

   To change the password:

   ```
   sp_ctrl>  connection {o.SID | r.database} set password=password
   ```

   where:

   - *SID* is the ORACLE_SID of the database, if the database is Oracle.
   - *database* is the name (not the DSN) of the database, if the database is non-Oracle.
   - *username* is the new account name.
   - *password* is the new password.

8. Start the SharePlex processes.

   ```
   sp_ctrl>  start service
   ```

Change the name or IP address of a replication host

Use the **provision** utility to change a host name or IP address in the SharePlex configuration. For more information on changing the name or IP address of a replication host, see the **Provision** utility section in the SharePlex Reference Guide.

Set the SharePlex Port Number

The SharePlex processes use TCP to communicate with each other between different systems and uses UDP to communicate within a system. The default TCP and UDP port numbers for SharePlex are both set to 2100 at the time of installation. For some deployments of SharePlex you may need to change the TCP or UDP port numbers.

Before selecting a port number, review the following points:

- If your replication strategy requires multiple instances of **sp_cop** on a system, you must set a unique port number for each one. For more information, see Run Multiple Instances of SharePlex on page 48.
- When an non-default port is required, the same number must be used for both the TCP and UDP ports, and it must be used for the TCP and UDP ports of all other instances of **sp_cop** that are involved in the same replication configuration. If the ports are different, **sp_cop** on one system cannot connect to the **sp_cop** on another system to send or receive messages and data.

Supported databases

All databases supported by SharePlex on all supported platforms

Set the SharePlex port on Unix and Linux systems

To set the port number on Unix and Linux systems, a SharePlex Administrator must set both the TCP and UDP port parameters in the **SharePlex environment**. If there is an active configuration, you will be instructed to stop access to the source objects and shut down **sp_cop**.

To finish setting the port in an active configuration:

1. (If using multiple variable-data directories] Export the SP_SYS_VARDIR environment variable to point to the correct variable-data directory for the port you are setting.

   ksh shell:

   export SP_SYS_VARDIR=*/full_path_of_variable-data_directory*

   csh shell:

   setenv SP_SYS_VARDIR=*/full_path_of_variable-data_directory*

2. Export the following environment variables.

   ksh shell:

   export SP_COP_TPORT=*port*

   export SP_COP_UPORT=*port*

   csh shell:

   setenv SP_COP_TPORT *port*

   setenv SP_COP_UPORT *port*

   where: *port* is the new port number

3. Change directories to the SharePlex product directory.

4. Start **sp_cop** and **sp_ctrl**.

   NOTE: If you are using multiple variable-data directories, start **sp_cop** with the **-u***port* option, where *port* is the port number that you have chosen for the variable-data directory that you exported.

   ./sp_cop [**-u***port*] **&**

5. Run sp_ctrl.

   ./sp_ctrl

6. In **sp_ctrl**, set the following SharePlex parameters.

   `sp_ctrl>` **set param SP_COP_TPORT** *port*

   `sp_ctrl>` **set param SP_COP_UPORT** *port*

7. Do one of two things:

   - If there is not an active configuration, use the **shutdown** command in **sp_ctrl** to stop **sp_cop**. The next time you start **sp_cop**, the new port number takes effect.

     NOTE: If you do not have an active configuration, **you are finished setting the port number**.

   - If there is an active configuration, continue to the next step.

8. Stop access to the replicating objects on the source system, then issue the **flush** command in **sp_ctrl** on the **source** system to clear the queues.

   `sp_ctrl>` **flush o.***database_identifier*

   where: *database_identifier* is o. followed by the Oracle SID, TNS alias, or PDB name that SharePlex uses to connect to the source database (depends on whether the database is a regular Oracle instance, RAC, or PDB in a container database).

9. On the **source** system, issue the **qstatus** command to verify that all of the messages reached the target system.

   `sp_ctrl>` **qstatus**

   Continue to issue the command until the export queue is empty.

10. On the **target** system, issue the **qstatus** command to verify that all of the messages were posted to the database. Continue to issue the command until the post queue is empty.
11. On the target system, issue the **status** command to verify that Post stopped.

    ```
    sp_ctrl> status
    ```
12. Shut down SharePlex on the source and target systems.

    ```
    sp_ctrl> shutdown
    ```
13. Start **sp_cop** on the source and target systems.

    > **NOTE:** If you are using multiple variable-data directories, start **sp_cop** with the **-u***port* option, where *port* is the port number that you have chosen for the variable-data directory that you exported.

    ./sp_cop [**-u***port*] **&**
14. Run **sp_ctrl** on the target system.
15. Start the Post process.

    ```
    sp_ctrl> start post
    ```
16. Allow users to access the replicating objects.
17. Use the **status** command on the source and target systems to verify that all SharePlex processes are running.

    ```
    sp_ctrl> status
    ```

Initial Database Synchronization for PostgreSQL to PostgreSQL Replication

This section provides information about how initial synchronization of the database is performed when replicating data from PostgreSQL to PostgreSQL (cloud target databases).

You can set up an active replication between the following source and target platform combinations:

- On-Prem Logical Source to Aurora Target
- On-Prem Physical Source to Aurora Target
- Azure Flexi Server Source to Aurora Target
- Aurora Source to Azure Flexi Target
- Aurora Source to On-Prem Target

To set up an active replication:

1. Run the **pg_setup** utility on both the source and target servers to prepare the environment.
2. Start SharePlex and execute the **stop post** command on the target server:

   ```
   sp_ctrl (pslrhel7linux01:4073)> stop post
   ```

   It ensures that even if a configuration file is not activated on the source, the Poster queue will be created but remain stopped.
3. Close access to the source database for users.
4. Activate the configuration file on the source.

   ```
   sp_ctrl> activate config <config_file_name>
   ```

   This creates the necessary replication slot as defined in **pg_setup**.
5. In the source database, create a snapshot while keeping the session open:

   ```
   begin transaction isolation level REPEATABLE READ;

   select pg_export_snapshot();
   ```
6. Use the following snapshot to run **pg_dump** on the source:

   ```
   pg_dump --snapshot=00000003-000001BC-1 -d <database_name> -Fc -b -v -f
   dump04172024.sql -n <schemas>
   ```

   For example:

   ```
   pg_dump --snapshot=00000003-000001BC-1 -d testdb -Fc -b -v -f dump04172024.sql
   -n splex --username=splex
   ```
7. Resume user access to the source database while **pg_dump** is in progress.

8. Use the dump file to execute **pg_restore** on the target database after **pg_dump** is completed:

   ```
   pg_restore -v -h pslaurorapgdb01.cihp157rpcvu.us-west-1.rds.amazonaws.com -U
   splex -d auroratest -j2 dump04172024.sql -n splex
   ```

9. Once **pg_restore** is executed, log in to the target database and check the tables.
10. After successful restoration, start the post queue on the target server using **start post** in **sp_ctrl**:

    ```
    sp_ctrl> start post
    ```

11. Validate replication to ensure everything is working correctly.

24

Apply an Oracle Application Patch or Upgrade

This chapter contains procedures to follow when you need to apply an application patch or upgrade and there is an active replication configuration. These procedures apply to Oracle databases.

Contents

Before you Patch or Upgrade an Application

Review the following topics before you patch or upgrade an application on a system where SharePlex replication is active.

Which procedure to use?

There are different procedures for applying an application patch or upgrade to an Oracle database while replication is in process. Which one to choose depends on what changes the patch or upgrade makes.

Changes made by the patch/upgrade	Steps to take
If the patch/upgrade applies DDL that is not supported by SharePlex. For details on the DDL that SharePlex supports, see the SharePlexSharePlex Release Notes.	Manually apply the patch/upgrade to the source and target by following either of these procedures: Apply Patch/Upgrade to Source then Copy it to Target on page 407 Apply Patch/Upgrade to Source and Target on page 410
If the patch/upgrade does any of the following: • Performs DML changes.	Manually apply the patch/upgrade to the source, then allow SharePlex to replicate the changes to the target. Follow this procedure: Apply Patch to Source and Replicate it to the Target on page 412

Changes made by the patch/upgrade	Steps to take
• Performs supported DDL on the source system. For details on the DDL that SharePlex supports, see the SharePlexSharePlex Release Notes. • Changes users and security on source system (other than SharePlex)	**NOTE**: Because this procedure assumes that SharePlex can replicate all of the changes that the patch or upgrade applies, the patch/upgrade is not applied to the target.

The effect of patches and upgrades on partitioned replication

A patch or upgrade can make changes that affect the column partitions of vertically partitioned replication in your configuration file. Take the following into account when you perform this procedure.

If the patch or upgrade does this to a table:	Do this:
Adds columns that do not satisfy the column partition of the table	(Optional) Drop the columns from the target table after the patch or upgrade is applied.
Adds columns that need to be in the column partition of the table	Add those columns to the source and target column partition lists in the configuration file.
Drops columns that are part of the column partition of the table	Remove those columns from the source and target column partition lists in the configuration file.
Changes the name of a column that is in the column partition of a table	Change the column name in the source and target column partition lists in the configuration file.

For more information, see Configure Vertically Partitioned Replication on page 141.

Naming conventions used

In these procedures, the "source" system is one of the following:

- The source system of a single-direction replication configuration, including cascading replication.
- All source systems of a consolidated replication configuration.
- The trusted source system in a peer-to-peer replication configuration.

In these procedures, the "target" system is one of the following:

- The target system of a single-direction replication configuration, including cascading and consolidated replication.
- The secondary systems in a peer-to-peer replication configuration.

In this procedure, the SharePlex commands in the procedure apply to all **sp_cop** instances that apply to the replication strategy you are using (for example, all **sp_cop** processes on a target in consolidated replication).

Apply Patch/Upgrade to Source then Copy it to Target

Supported databases

Oracle on all supported platforms

When to use this procedure

Use this procedure if the patch or upgrade makes DDL changes of a type not replicated by SharePlex. For a list of objects for which DDL is supported, see the SharePlex Release Notes.

Overview of the procedure

Use this procedure to run an Oracle hot backup to copy patches or upgrades from the source system to the target system, instead of applying the patch or upgrade directly on the target system. This is useful if the patch or upgrade makes extensive changes that are of the type(s) not supported by SharePlex replication, or if you are unsure of what it does.

With this procedure, you can keep the configuration file active on the source system. You use the reconcile command to identify and eliminate the following:

- Duplicate DML and supported DDL from the patch or upgrade operations that were replicated but also applied by the backup.
- Production transactions that were replicated but also applied by the backup.

Apply the patch/upgrade

To apply the patch or upgrade:

1. Stop user access to the instances involved in replication on the source and target systems, but do not shut down SharePlex.
2. On the source system, run **sp_ctrl**.
3. On the source system, flush the data to the target system. This command stops Post and places a marker in the data stream that establishes a synchronization point between source and target data.

   ```
   sp_ctrl> flush datasource
   ```

   where: *datasource* is the datasource of the source instance, for example **o.oraA**.
4. On the source system, apply the patch or upgrade.
5. On the source system, restore user access to the source instance.

6. [If the patch/upgrade adds objects that must be replicated] Edit the configuration file as follows (do not deactivate it). The patch or upgrade may have affected column partitions or column conditions in partitioned replication. For more information, see Change an Active Configuration File on page 386.

   - Copy the configuration file.

     ```
     sp_ctrl> copy config filename to newname
     ```

   - Edit the copy.

     ```
     sp_ctrl> edit config newname
     ```

     Save the file.

7. **Do one of the following:**

   - If you added objects in the previous step, activate the new configuration file.

     ```
     sp_ctrl> activate config newname
     ```

   - If you did not make any changes to the original configuration file, activate that one.

     ```
     sp_ctrl> activate config filename
     ```

8. On the source, run the Oracle hot backup.
9. On the source, switch log files and make a note of the highest archive-log sequence number.

   On-premises database:

   ```
   svrmgr1> alter system switch logfile;
   ```

   Amazon RDS database:

   Use Amazon RDS procedure **rdsadmin.rdsadmin_util.switch_logfile**.

10. On the target, recover the target database from the hot backup using the UNTIL CANCEL option in the RECOVER clause, and cancel the recovery after Oracle fully applies the log that you recorded in the previous step.
11. On the target system, open the database with the RESETLOGS option.
12. Run the Database Setup utility on the target instance, but do not create a new user. Choose the existing SharePlex user and password (copied in the backup). For more information, see Database Setup Utilities in the SharePlex Reference Guide.
13. On the target system, issue the **reconcile** command using the sequence number of the log that you noted previously. If you are using named post queues, issue the command for each one. If you do not know the queue names, issue the **qstatus** command first.

    ```
    sp_ctrl> qstatus
    sp_ctrl> reconcile queue queuename for datasource-datadest seq sequence_number
    ```

    Example: **reconcile queue SysA for o.oraA-o.oraA seq 1234**

    NOTE: The reconcile process retains control of **sp_ctrl** until it is finished.

14. On the target system, if the patch or upgrade installed triggers on the tables in replication, disable them or run the **sp_add_trigger.sql** utility script so that the triggers ignore the SharePlex user.

15. On the target system, if the patch or upgrade added check constraints or scheduled jobs that perform DML, disable them.
16. On the target system, perform any cleanup required by partitioned replication.
17. On the target system, start Post.

    ```
    sp_ctrl> start post
    ```

    The two instances are now in synchronization, and SharePlex resumes replication.

Apply Patch/Upgrade to Source and Target

Supported databases

Oracle on all supported platforms

When to use this procedure

Use this procedure if the patch or upgrade makes DDL changes of a type not replicated by SharePlex. For a list of objects for which DDL is supported, see the SharePlex Release Notes.

Overview of the procedure

Use this procedure to apply an application patch or upgrade if it includes changes to the database that are not replicated by SharePlex and you can stop user access to the source database to deactivate and reactivate the configuration file. It requires deactivation of the configuration file so that SharePlex can rebuild its object information to incorporate the changes that the patch or upgrade applied. When you reactivate the configuration, SharePlex will re-analyze all of the objects again. You can allow users to access the source data while the patch or upgrade is applied to the target system.

Apply the patch/upgrade

To apply the patch or upgrade:

1. Stop user access to the instances involved in replication on the source and target systems, but do not shut down SharePlex.
2. On the source system, flush the data to the target system. This command stops Post and places a marker in the data stream that establishes a synchronization point between source and target data.

   ```
   sp_ctrl> flush datasource
   ```

   where: *datasource* is the datasource of the source instance, for example **o.oraA**.
3. On the source system, deactivate the configuration.

   ```
   sp_ctrl> deactivate config filename
   ```
4. On the source system, apply the patch or upgrade.
5. [If the patch/upgrade adds objects that must be replicated] On the source system, edit the configuration file, including making any changes to column partitions or column conditions if using partitioned replication. For more information, see Change an Active Configuration File on page 386.

   ```
   sp_ctrl> edit config filename
   ```

6. On the source system, activate the configuration file.

   ```
   sp_ctrl> activate config filename
   ```

7. On the source system, restore user access to the source instance.
8. On the target system, apply the patch or upgrade.
9. On the target system, if the patch or upgrade installed triggers on the tables in replication, disable them or run the **sp_add_trigger.sql** utility script so that the triggers ignore the SharePlex user.
10. On the target system, if the patch or upgrade added check constraints or scheduled jobs that perform DML, disable them.
11. On the target system, perform any cleanup required by partitioned replication.
12. On the target system, start Post.

    ```
    sp_ctrl> start post
    ```

    The two instances are now in synchronization, and SharePlex resumes replication.

Apply Patch to Source and Replicate it to the Target

Supported databases

Oracle on all supported platforms

When to use this procedure

Use this procedure if all of the operations applied by a patch or upgrade are supported by SharePlex and can be replicated to the target. This includes DML changes and DDL that is supported by SharePlex. If you are not sure whether the patch or upgrade performs operations that are not supported by SharePlex, use the procedure Apply Patch/Upgrade to Source then Copy it to Target on page 407.

NOTE: For a list of operations that SharePlex supports, see the SharePlex Release Notes.

Apply the patch/upgrade

To apply the patch or upgrade:

1. Stop user access to the Oracle instances on the source and target systems.
2. On the source system, flush the data to the target system. This command stops Post and places a marker in the data stream that establishes a synchronization point between source and target data.

   ```
   sp_ctrl>  flush  datasource
   ```

   where: *datasource* is the datasource of the source instance, for example **o.oraA**.
3. On the source system, apply the patch or upgrade.
4. On the source system, restore user access to the source instance.
5. On the target system, if the patch or upgrade created or modified triggers, disable them or run the **sp_add_trigger.sql** utility script so that the triggers ignore the SharePlex user.
6. On the target system, restore user access to the target instance.

25

Back up Data on the Source or Target

This topic contains procedures for making backups of source and target data while replication is active.

Contents

Perform a Partial Backup of the Source Data

To perform a partial back up of a source system (for example, to extract data or populate a data warehouse) while data is being replicated, you can perform the partial backup on the target system instead and copy the same data as you would on the source system.

This procedure does not interrupt user access to the source data and does not deactivate or reactivate the configuration file.

Supported databases

Oracle to all targets

Procedure

To perform a partial backup of the source data:

1. On the source system, start **sp_ctrl**.
2. On the source system, flush the data to the target system. This command stops the Post process and places a marker in the data stream that establishes a synchronization point between source and target data.

   ```
   sp_ctrl>  flush datasource
   ```

   where: *datasource* is the datasource specification of the source database in the configuration file, for example **o.ora1**.
3. On the target system, back up the data.

NOTE: The data now matches the source data at the flush marker.

4. On the target system, start the Post process.

   ```
   sp_ctrl> start post
   ```

Perform a Full Backup of the Source System

To back up the entire source system, including SharePlex, you must shut down SharePlex replication while the backup is performed.

This procedure stops user access to the source data but does not deactivate or reactivate the configuration file. Replication resumes when started after the backup.

Supported databases

Oracle to all targets

Procedure

Perform these steps on the source system:

1. Stop all system activity.
2. Start **sp_ctrl**.
3. Flush the data to the target system. This command stops the Post process and places a marker in the data stream that establishes a synchronization point between source and target data.

   ```
   sp_ctrl> flush datasource
   ```

   where: *datasource* is the datasource specification of the source database in the configuration file, for example **o.ora1**.
4. Shut down SharePlex. This command shuts down SharePlex.

   ```
   sp_ctrl> shutdown
   ```
5. Shut down the database.
6. Perform the backup.
7. Start the database.
8. Start **sp_cop** (Unix and Linux).
9. Start **sp_ctrl**.
10. Allow users to access the database.
11. Verify that the SharePlex Capture, Read, and Export processes started.

    ```
    sp_ctrl> status
    ```

Perform these steps on the target system:

1. Start Post.

   ```
   sp_ctrl> start post
   ```

2. Verify that Post started.

   ```
   sp_ctrl> status
   ```

26

Troubleshooting Tips

Use the following resources from the Quest Support Portal to help you troubleshoot SharePlex.

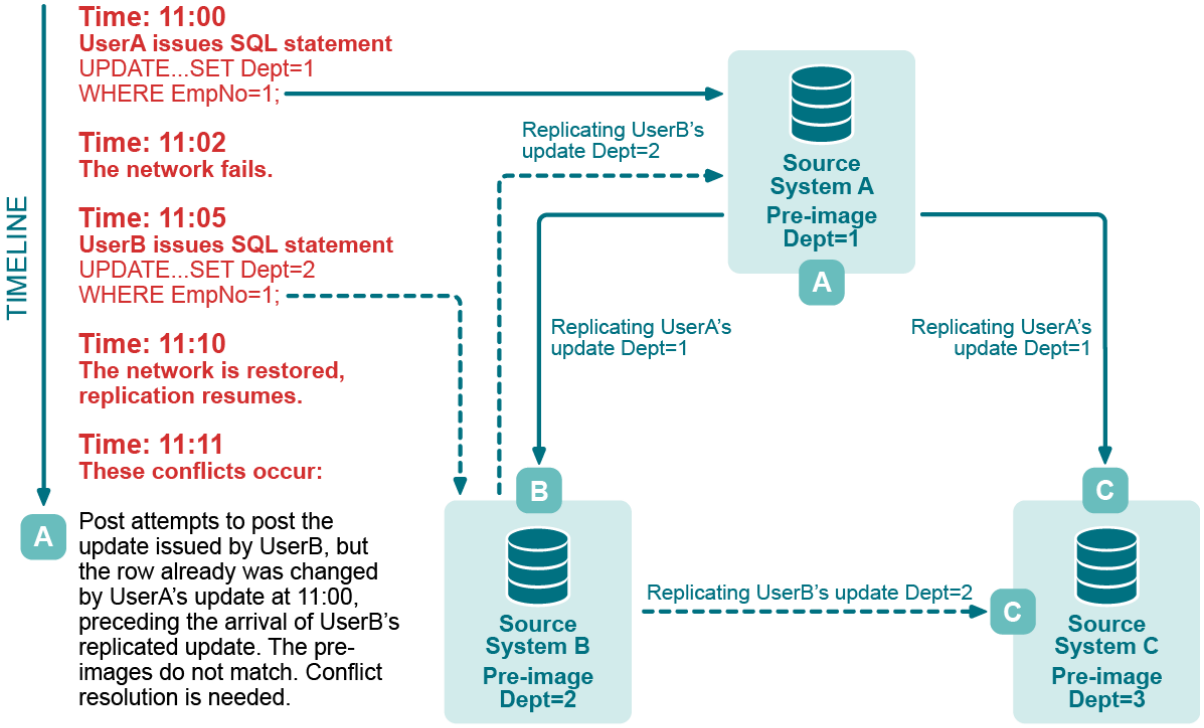
- SharePlex Knowledge Base articles
- How to do basic troubleshooting with SharePlex (video)

See also, Prevent and Solve Replication Problems on page 304.

A

Appendix A: Peer-To-Peer Diagram

This diagram visually explains the concept of peer-to-peer replication. For more information, see Configure Peer-to-Peer Replication on page 165.



A Post attempts to post the update issued by UserB, but the row already was changed by UserA's update at 11:00, preceding the arrival of UserB's replicated update. The pre-images do not match. Conflict resolution is needed.

B Post attempts to post the update issued by UserA, but the row already was changed by UserB's update at 11:05, preceding the arrival of UserA's replicated update. The pre-images do not match. Conflict resolution is needed.

C Post attempts to post the updates issued by UserA and UserB, but if the two arrive at the same time, a conflict occurs. the row already was changed by Or, if one arrives before the other the pre-images will not match. Conflict resolution is needed.

B

Appendix B: SharePlex Variables

SharePlex uses the following environment variables, which you may need to set in certain situations. Usually you must perform additional steps before or after setting a variable, so refer to the recommended instructions before setting a SharePlex variable.

Environment Variable	Description
EDITOR	Sets the default ASCII text editor for **sp_ctrl** commands that use one, for example the **create config** command.
HOST	Sets a host name for all locally run sessions of **sp_ctrl**.
SP_COP_TPORT	Sets a non-default port number for an instance of SharePlex. The default port number is 2100. You may need to set a different port number if one of the following is true: • You are setting up additional instances of **sp_cop**. • A different port number than 2100 must be used.
SP_SYS_HOST_NAME	Sets the host name that SharePlex binds to during configuration activation. This variable is used for the following: • Sets the virtual IP address (also known as the *global cluster package name*) on a clustered system, such as Oracle RAC. This variable must be set on all cluster nodes. • If SP_SYS_HOST_NAME is set to an IPV6 address on the source system, SharePlex on the target system must be version 9.0 or later.
SP_SYS_VARDIR	Sets the full path to the SharePlex variable-data directory so that **sp_cop** can locate the configuration data, queues, logs and other information. If there is only one instance of **sp_cop** on the local system, this variable is set by default*. If there are multiple instances of **sp_cop** on the local system, always set this variable to point to the correct variable-data directory of an instance before setting any other SharePlex variables for that instance.
SP_SYS_SECURE_MODE	Suppresses the output of the compare and repair SQL log file for all compare and repair runs while the current instance of SharePlex is running. This variable must be set before starting SharePlex, so if the **sp_cop** process is running it must be restarted after setting this variable. When **sp_cop** is run with this environment variable, the compare and repair commands will not put data into SQL files and the Post process will not put data into the SharePlex error log.

* On Unix and Linux, the variable-data directory is set in the *proddir*/**data**/**default.yaml** file.

To set an environment variable in Unix or Linux:

ksh shell:

export *variable_name*=*value*

csh shell:

setenv *variable_name value*

ksh shell:

export SP_SYS_VARDIR=*full_path_of_variable-data_directory*

csh shell:

setenv SP_SYS_VARDIR *full_path_of_variable-data_directory*