

Hitachi ID Bravura Security Fabric

Replication and Recovery

Software revision: 12.2.4
Document revision: 30072
Last changed: 2022-03-01

Contents

1	About the documentation	2
1.1	Conventions	2
1.2	Feedback and help	3
I	REPLICATION	4
2	About Replication	5
2.0.1	Terminology	5
2.0.2	Why is replication required?	7
3	File Synchronization	9
3.1	File synchronization architecture	10
3.1.1	What is propagated	10
3.1.2	Distribution of application proxies in the network	11
3.1.3	When file/registry propagation happens	11
3.1.4	Troubleshooting	12
4	Database Replication	13
4.1	Database replication architecture	14
4.1.1	Fault tolerance: geographic distribution	14
4.1.2	Application level versus database-native	14
4.1.3	The need for application-level replication	15
4.1.4	Replicated network topology alternatives	17
4.2	About the database replication process	22
4.2.1	The process	22
4.2.2	Problematic states	22

4.3	Replication in detail at a service level	24
4.4	What is not replicated	26
4.5	Impact of environment	27
4.5.1	Impact of database server issues	27
4.5.2	Impact of application server resources	28
4.5.3	Impact of network latency	29
4.5.4	Impact of bandwidth constraints	29
4.6	Load balancing	31
4.7	Configuring database replication	32
4.7.1	Preparation	33
4.7.2	Adding a node	37
4.7.3	Checking replication status as an Administrator	43
4.7.4	Automated node check when using a load balancer	45
4.7.5	Modifying node settings	46
4.7.6	Removing a node from replication	50
4.7.7	Cleaning up replication configuration	52
5	Recovering from Network Faults	53
5.1	Data Replication Queue	53
5.1.1	Queue settings	53
5.1.2	DB COMMIT SUSPEND mode	54
5.1.3	Event actions (exit traps)	55
5.1.4	Time available to fix problems	56
5.2	Modes of Failure	57
5.2.1	User loses connectivity	57
5.2.2	Single <i>Bravura Security Fabric</i> server fails	59
5.2.3	Link between single <i>Bravura Security Fabric</i> server and its database goes offline	60
5.2.4	Single <i>Bravura Security Fabric</i> database goes offline	62
5.2.5	Link between two <i>Bravura Security Fabric</i> servers goes offline	63
5.2.6	Link between <i>Bravura Security Fabric</i> servers and proxy servers goes offline	64
5.2.7	A single proxy server fails	66
5.2.8	A link to a site where there are target systems fails	67

5.2.9	A single target system goes offline	68
5.2.10	A site where one or more <i>Bravura Security Fabric</i> servers is installed is offline	69
5.2.11	A site where one or more <i>Bravura Security Fabric</i> proxy servers is installed is offline . . .	70
5.2.12	A site where one or more target systems is installed is offline	71
5.3	Troubleshooting <i>Bravura Security Fabric</i> server failures	71
5.3.1	Replication errors in log files	71
5.3.2	Databases contain unsynchronized data	72
5.3.3	Missing iddb queue files	73
5.3.4	Missing psupdate replication batch files	74
5.3.5	Unreadable passwords	74
5.3.6	Server clocks unsynchronized	74
5.4	Synchronizing a new node with an existing set of <i>Bravura Security Fabric</i> replicas	75
5.4.1	Resynchronizing databases	75
5.4.2	Configuring the new application node and testing replication	77
5.5	Recommended maintenance plan	79
6	Analytics (SSRS) report synchronization	80
II	TOOLS	82
7	updinst	83
7.0.1	Use Case: updinst config file	86
8	idfilerep	87
9	updproxy	90
10	dbperfchk	92
11	iddbadm	94
12	servicemove	95
13	File Locations	97
13.1	<i>Bravura Security Fabric</i> directories and files	97
13.1.1	Instance directory	98

13.1.2 Log directory 100

13.1.3 Locks directory 101

13.2 Connector pack directories and files 102

III APPENDICES 103

A Calculating the Number of Application Nodes Required 104

Hitachi ID Systems, Inc.

DISCLAIMER

Although every effort has been made to ensure that the information in this manual is accurate, some information may be inconsistent with the most current software release.

In some cases, information may only be relevant to products you do not have installed.

For assistance with installation and configuration, please contact support@Hitachi-ID.com.

About the documentation

1

1.1 Conventions

This document uses the following conventions:

This information ...	displayed in ...
Variable text (substituted for your own text)	<angle brackets>
Non-text keystrokes – for example, [Enter] key on a keyboard.	[brackets]
Terms unique to <i>Hitachi ID Bravura Security Fabric</i>	<i>italics</i>
Button names, text fields, and menu items	boldface
Web pages (names)	<i>italics and boldface</i>
Literal text, as typed into configuration files, batch files, command prompts, and data entry fields	monospace font
Wrapped lines of literal text (indicated by the → character)	Write this string as a →single line of text.
Hypertext links – click the link to jump to a section in this document or a web site	Purple text
External document – click the link to jump to a section in another document. The links only work if the documents are kept in the relative directory path.	Magenta text

1.2 Feedback and help

If you have feedback about this document or wish to report an omission or error, please contact doc-feedback@Hitachi-ID.com.

If you require technical assistance with *Hitachi ID Bravura Security Fabric*, contact support@Hitachi-ID.com.

Part I

REPLICATION

About Replication

2

This document describes database replication between multiple physical or virtual servers hosting a single instance of any *Hitachi ID Bravura Security Fabric* product. It is intended to help architects, network engineers and database administrators to understand, plan for and operate the *Bravura Security Fabric* in a fault-tolerant and scalable fashion.

In a replicated *Bravura Security Fabric* environment, there are multiple data sets that have to be kept consistent on all nodes, from two points of view:

- Data use (application data and application configuration).
- Storage mechanism (files/registry and backend database) configuration data; files and the backend database contain both configuration and application data.

There are two types of replication in *Bravura Security Fabric*:

- **File Synchronization** Changes are copied always from primary node to secondaries and proxies (master-slave).
- **Database Replication** Changes from each node are sent to all other nodes (multi-master). Proxies are not involved.

2.0.1 Terminology

Latency the time it takes for a data packet to be transmitted from one physical server to another over the network.

Bandwidth the maximum number of bits per second that can be transmitted from one physical server to another over the network.

Local area network a set of connections between servers where the worst-case latency is low (for example, less than 10ms) and the worst-case bandwidth is high (for example, over 100Mbps).

Wide area network a set of connections between servers where the best-case latency is high (for example, more than 50ms) and the best-case bandwidth is low (for example, under 10Mbps).

Application this is a reference to the *Hitachi ID Bravura Security Fabric* application being deployed, being one of:

- *Hitachi ID Bravura Identity*
- *Hitachi ID Bravura Privilege*
- *Hitachi ID Bravura Pass*

Instance a single running copy of the *Bravura Security Fabric* application, or all copies of the application running under the same instance name and linked through replication (a *replicated* instance).

Server a physical or virtual server running the Windows 2012 or 2016 operating system on which at least one application instance and/or a MSSQL database server is installed.

Application node a single physical or virtual server housing at least one instance of the *Bravura Security Fabric* application.

Backend database a single instance of Microsoft SQL Server housing the internal data used by a single instance; this is recommended to be replicated between database nodes using data replication.

External database A couple of SQLite databases residing in the db\ directory of each application node and housing configuration data for the component framework; this can be changed only on the primary node and is replicated using file replication.

Logical schema a set of database objects (the description of tables, views, indexes, procedures, etc, but without their data) used as the back-end for one or more application nodes. It is specific to each version of *Bravura Security Fabric*. External databases also have their own, version-specific logical schemas. The same logical schema must be used for all replicated database nodes and for all instances used for developing and testing it.

Database node One or more application nodes using the same backend database, all hosted in the same data center, as close as possible to each other on the network (to reduce latency).

A single application instance may be distributed over multiple nodes, where the database nodes may be very far apart and have relatively low network bandwidth / high latency between them as compared to connectivity between servers in the same database node.

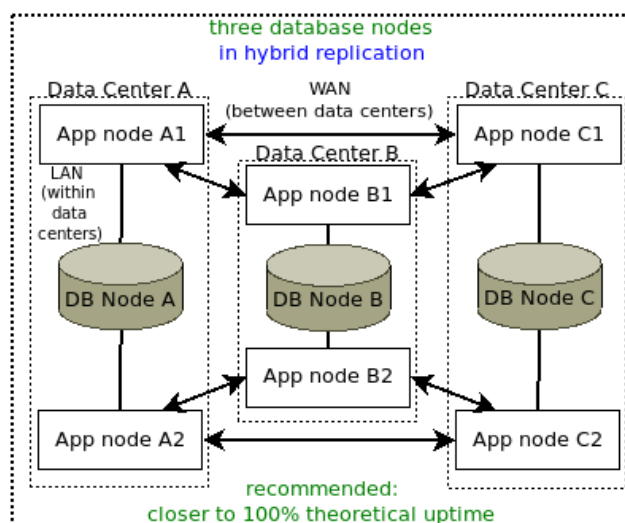


Figure 2.1: Network topology

2.0.2 Why is replication required?

There are quite a few reasons why an enterprise application like *Hitachi ID Bravura Security Fabric* requires scalability.

Two or more replicated nodes

If there is a lot of interaction expected from users, the web-based interface processing will have to require attention: OS processes and IIS concomitant processing and other OS resource limits can be reached.

In that case, adding nodes in *shared schema* would be the solution (that is 2 or more nodes connecting to the same database).

At least two replicated full nodes (application+DB) are required if:

- Geographical distribution and disaster failure have to be considered, (one node per Internet-routing geographical area, at a minimum);
- Resource-consuming functionality can be segregated on separate dedicated nodes:
 - **psupdate** and other discovery processes
 - User interaction (so **psupdate** and other automated processes do not slow it down)
 - Disaster recovery (one database node in a different data center)
 - For *Hitachi ID Bravura Privilege*, different managed-system-policy processing can be assigned to different nodes.
 - Storage-intensive session monitoring can be limited to one (hybrid architecture) database node that does not do anything else.
 - For *Hitachi ID Bravura Identity*, any resource-intensive automation can be processed on a (hybrid) node that doesn't do anything else, *if* it does not depend on the most recent data processed on other nodes.

Hybrid architecture replication

After adding a certain number of database nodes, the transactional overhead between linked **iddb** services reduces the additional resource benefits.

- After five or six nodes in direct (1:1) replication, the added benefits are negligible.
- The upper number of nodes that allow for scaling up of the instance, depends on the characteristics of the available network between the nodes (latency and in some cases, bandwidth; see the sections on those topics).
- For that reason, *hybrid architecture* replication can be added; x sets of (y nodes connected to the same database) can perform better than x*y nodes in direct replication.
- Hybrid architecture combines the benefits of direct data replication with those of shared schema, to provide resources for critical instances, or instances serving many users (in the hundreds of thousands or more), or with very complicated, resource-intensive solutions.

One database node, no replication

On small, low use, low automation instances, resources can be saved by placing the database server on the application server (Database node A, primary-only, with its own database, no replication):

- This removes network latency from App-DB communication, speeding up the instance.
- It also makes *Hitachi ID Bravura Security Fabric* easier to maintain (including backups).
- Even if there is not much activity (for instances serving under 5000 profiles with few targets), the data on a single node with the database running on the same server will be increasing to the point where the database engine and the application will start competing for resources. At that point a separate database server will be required - still no replication is involved.

However, this *one-node* topology presents a single point of failure, and is *not* recommended for mission-critical solutions.

Not all configuration changes get replication between nodes

When working in a replicated environment, making a configuration change on the primary node will usually replicate the change to the other application nodes and proxies.

- Most system variables and other backend database changes are replicated instantly. External database and component changes are sent every five minutes by default. This interval can be configured in **Windows Task Scheduler** → **Library** → **HID external database replication**.
- Changes to instance files and registry, including some system variables, are sent once a day, or whenever **PSUPDATE** in **Manage the system** → **Maintenance** → **Scheduled jobs** is configured to run:
 - These are sent to secondary nodes from the **updinst** utility on the primary node sending to the **idfilerep** service on each secondary node,
 - and to proxies with the **updproxy** utility on the primary node sending to the **psproxy** service on each proxy.
- Some changes are not replicated at all:
 - Most files from the instance directory are replicated. The Program Files\Common Files\Hitachi ID files are not, so Connector Packs have to be installed on each secondary node and each proxy node separately.
 - To see the blacklist(s) for file/registry replication, open a command prompt in the instance's util\ directory and run `updinst -showconfig`.
 - The blacklist for **updproxy** is built into the binary. There is no config file. Contact support@Hitachi-ID.com with the *Hitachi ID Bravura Security Fabric* version installed on your instance if you want to know that blacklist.

Keep this in mind and document your configuration for the Hitachi ID Systems maintenance team to be aware of which specific configuration changes will be replicated to other nodes, and when.

File Synchronization

3

In addition to database replication, there is a need to replicate the contents of each application node's files and registry key belonging to the instance. This is to forward configuration changes, user interface customization, copies of software updates, and so on, from the primary application node to all other application nodes and proxies, so as to minimize administration effort and reduce the likelihood of error.

File/registry replication happens at the end of the nightly auto-discovery process on the “primary” *Hitachi ID Bravura Security Fabric* application node; the primary application node sends files and relevant registry entries to each “secondary” application node and each configured *Bravura Security Fabric* proxy server. If a configuration change is made on the primary node that needs to be replicated earlier than the scheduled process, replication can be triggered either by running the entire auto-discovery process manually, using the administrative console (**Manage the system** → **Maintenance** → **File synchronization**) or from the command-line by running **updinst**, located in the util directory.

The product installer creates a Windows Scheduled Task that uses **updinst** to propagate changes in the external databases (by copying the `extdb.db` and `component.db`) from the primary node to secondaries, every five minutes. As part of the initial configuration, and later if the primary application node changes, that scheduled task should be disabled on secondary nodes, and even on replicated instances whose configuration and other data saved in those databases does not change.

Similar to data replication, file/registry replication is also encrypted using a shared-key handshake. It is a batch process that typically takes just under a minute to complete, in a typical scenario where there is relatively little daily-changed configuration data to forward.

There are white-lists and black-lists that specify what data to include and what data to omit in the context of file/registry replication. For example, log files (`idmsuite.log`) and database service configuration files (`iddb.cfg`) are not replicated by default, as this would be wasteful (log files) or cause incorrect configuration to be forwarded (database service configuration).

These lists are a combination of items built into the replication software plus user-configurable items that are set by editing a configuration file (`psconfig/updinst.cfg`).

3.1 File synchronization architecture

Hitachi ID Bravura Security Fabric's native file synchronization does not synchronize in the true sense of the term. The process is one of propagation.

Bravura Security Fabric's file propagation:

- Sends all files inside the instance's folders.
- It also sends the registry entries of the instance.

CAUTION: Check that the `servicelist:address` field values in the backend database are FQDN. This is especially important for *Bravura Privilege*.

- All of this propagated data contains mostly configuration.
- Files that are deleted on the primary node are deleted by the file replication service **idfilerep** on the secondaries and proxies.

From the primary node, files are sent using various utilities:

updinst The file replication service **idfilerep** on the secondary node sends the primary node a list of files and registry entries. The **updinst** utility on the primary node sends changes back to the secondary nodes.

updproxy The **psproxy** service on the application proxy server sends the primary node a list of files and registry entries. The **updproxy** utility on the primary node sends changes back to the proxy server.

CAUTION: The **updinst** utility should be triggered ONLY from the primary node (with the exception of the `-extdb` option that is to be used only for the rare troubleshooting step of recovering from secondaries external databases corrupted on the primary node). When starting **updinst** from a secondary node you will receive a warning that the current node is not the primary and service will close again.

The Hitachi ID Systems *external database replicator*, which is enabled by default on each node installation, should be disabled on secondaries and in single-node instances.

3.1.1 What is propagated

Not all files and registry entries are propagated. To determine which, there are both blacklists and whitelists configured by default inside the **updinst** binary, and new ones can be added with a `cfg` file.

The utility itself allows you to check what files are copied and which are not by running: `updinst -showconfig` from the `instances_util` directory.

If **psupdate** fails for some reason, and the instance's email configuration is correctly configured, the admin(s) listed in the configuration will receive an email about the failure.

The **updproxy** binary does not use a config file, all white/black-lists are built into the binary. Contact support@Hitachi-ID.com with the *Hitachi ID Bravura Security Fabric* version installed on your instance to obtain the blacklist information.

See also:

- [Use Case: updst config file](#) for a detailed example of the updst config file.
- [updst](#) for more information about the **updst** utility.
- [idfilerep](#) for more information about File Replication Service (idfilerep).
- [Analytics \(SSRS\) report synchronization](#) for more information about Analytics (SSRS) report synchronization.

3.1.2 Distribution of application proxies in the network

Application proxies do not have a backend database. They are added to a solution in order to run agents in environments without VPN tunnels configured between their sites, where all *Hitachi ID Bravura Security Fabric* nodes cannot reach all target systems.

An application proxy allows for a single port to be open in firewalls (any port), even different for different proxies. For security reasons it is useful to add an application proxy which makes a single obscure port between subnets and data centers, instead of opening the many ports required for integration with target systems. Actual application ports are known to hackers and often targeted to penetrate network security.

3.1.3 When file/registry propagation happens

While database replication is instant (happens every second), file and registry propagation from primary node happens manually or on schedule:

- Manually by an administrator logged into the primary node's web-based interface:
 - **Manage the system** → **Maintenance** → **Auto discovery** → **Execute autodiscovery**
or
 - **Manage the system** → **Maintenance** → **File synchronization**
- Scheduled:
 - In **Manage the system** → **Maintenance** → **Scheduled jobs** → **PSUPDATE** (or additional psupdate jobs)

Note: **psupdate** triggers **updst** and **updproxy** only if the system variable "PSUPDATE FILE REPLICATION" is enabled.

- Scheduled in the **Administrative tools** → **Task Scheduler** → **Library** → **HID External Database Replicator** task. This copies the external database and components database only.
- With any other scheduler by running `updinst+upproxy` or `psupdate` following the instructions in the `reference.pdf` for each of those utilities.

3.1.4 Troubleshooting

3.1.4.1 Dealing with locked files

When `idfilerep` cannot replace a file because the file is locked by some process:

- and there is no `.busy` extension version of that file, it leaves a copy with the `.busy` extension.
- If the `.busy` version already exists, it leaves a copy with an `.nnnn` extension.
- The timestamps on the files can usually be reliable enough to determine which file is newest by checking the creation time.

For an administrator to restore the correct version of the file, the newest file has to be renamed with the original file extension; for example, `exe`, `dll` extensions. The rest of the files should be removed to a backup location in case there was something important in any of them; they act as a tiny history for that file.

3.1.4.2 Tracing file propagation activity

File "sync" activity (success/failure, per file) can be found in the `idmsuite.log`:

- When you search for entries logged by the `updinst` utility.
- On the receiving node end, it is the *file synchronization* service (by default listening on port 2380). See `idfilerep` entries in the secondaries' `idmsuite.log` that handles placing those files in their correct locations.
- For application proxy servers, the sender is `updproxy`. Check its entries on the primary nodes log. The receiving end is the `proxy` entries in the proxy log.
 - Unlike `updinst`, which replicates with parallel threads, `updproxy` updates serially, one file at a time, so it can take much longer to update an application proxy than a node, especially the first time the proxy is added or when the `updproxy` is forced to send all files.
 - However, `updinst` is used on the primary node for retrieving log files from both secondary nodes and application proxy servers.

Database Replication

4

Data replication can be done in many ways. We will start with a short summary of the database replication chapter, then go into details in separate sections.

Hitachi ID Systems does not recommend using natively replicated databases or cluster nodes on the back-end, because it can result in:

- Best case, in a shared schema, with the database engine moving too much data between data centers (including many cache and temporary tables needed only locally, about 40% of the database size).
- In cases where the replicated database or cluster configuration solution used lock the product schema while processing native replication, it can lead to failed psupdates or even data corruption.

The highly recommended replication strategy, included by default in *Hitachi ID Bravura Security Fabric*, is application-level database replication, which is transaction-based:

- Only the data that needs to be replicated is copied over.
- Any bulk changes are applied as stored procedures (they run fast on each database and only a few strings are sent over: the stored procedure name and its arguments).
- The schema is never locked by replication itself, so it is available for changes being done, without locking records or entire tables, resulting in faster performance.

How many databases to use:

- One database node exposes the instance to failure with no possible recovery of historical data.
- Two database nodes in replication expose the instance to outages whenever database propagations from one node to another are required.
- Three database nodes provide the most reliable architecture for instances which have a legal responsibility to Audit and a requirement for maximized application uptime.

4.1 Database replication architecture

When an instance is distributed on more than one database node, even if it's within the same data center, its data and configuration should be kept very closely synchronized, so that user experience is the same no matter what application node they are currently using.

This section explains why Hitachi ID Systems developed data replication at the application level, and then provides details on the possible topologies briefly introduced in the previous section.

4.1.1 Fault tolerance: geographic distribution

It is strongly recommended that at least two application nodes be installed and – if possible – that they be located in different data centers. This is especially true of *Hitachi ID Bravura Privilege*, where the geographical distance between application nodes should be maximized, so as to minimize the probability that a regional disaster (hurricane, earthquake, tsunami, etc.) causes privileged passwords for systems in other regions to be lost.

The built-in replication, both of file/registry contents (during psupdate) and of database contents (near real time and nightly) supports multiple servers, distributed over a wide area network. It is reasonably tolerant of low bandwidth (under 10Mbps), high latency (up to 150ms) and is encrypted to protect against man-in-the-middle attacks.

4.1.2 Application level versus database-native

Some applications require a high performance backend database but not near-100% uptime. In these cases, it makes sense to construct a high-availability, high-capacity database cluster and share this cluster between many applications; this is also known as backend/database-native replication.

It is recommended to deploy an application in a geographically distributed architecture without having to configure multiple (expensive) database clusters – one per region; however, there are pros and cons with each type of replication.

The drawbacks of using a natively-replicated database or a cluster are:

- If the cluster is damaged or becomes unreachable, then all of the applications that use it go off-line.
- A cluster or any other form of database-level replication sends all data changes to its replicas.
- Backups are critically important in a database cluster, since failures impact many applications.
- Modern database server software from Microsoft and other vendors have poor isolation between logical instances. An overloaded or badly written application using one logical schema can consume too much session space, disk I/O and CPU, impacting the performance and availability of other applications that use the same cluster.

Application-level replication, on the other hand:

- Eliminates any single point of failure, so is more fault tolerant.
- Can be configured to provide total isolation between applications. A badly behaving database node with its own, private database server cannot adversely impact the data on other database nodes.
- Can reduce network traffic and increase performance by only replicating the data that is important to exist on all replicas.

The main drawbacks of application-level replication are:

- As a result of replication failures, the database nodes can fall out of sync; in application-level replication, the sync status has to be verified as part of regular database maintenance.
- Applications have to be written with replication in mind. In particular, the process used to assign unique identifiers to new records has to be replication-aware.
- More physical databases are deployed.
- As mentioned above, more maintenance is needed to verify data synchronization, and decisions on what data to keep and what data can be left out of sync have to be made. This is actually a trade-off; in database-level replication data is lost on all nodes, and in application-level replication data is lost only on the database node where the error occurs.

Hitachi ID Systems Recommendation:

Since high availability is such an important design parameter for the Hitachi ID Bravura Security Fabric in general and of Hitachi ID Bravura Privilege in particular, Hitachi ID Systems strongly recommends that customers deploy a replicated architecture, rather than overburden an existing database cluster.

4.1.3 The need for application-level replication

Database products from Microsoft and other vendors include native support for database replication. This raises the question of whether another replication layer is required. Following are reasons to use the application-level replication provided by *Hitachi ID Bravura Security Fabric*:

- Native replication is very complex to configure in order to make its performance optimal for each application. The set of available configuration options depend heavily on the runtime behavior of the application using that database; this may even differ from table to table:
 - Should logical or physical replication be used? Physical replication operates at the level of disk blocks while logical operates at the level of some database objects (details below).
 - If logical replication is chosen, at what level of granularity should replication be configured? Options include record-level, column-level, table-level or transactional.
 - Should replication be synchronous (potentially making each instance of the application slower) or asynchronous (where there are short time intervals during which each instance has different, possibly inconsistent data)?
 - Should failed updates be queued up or discarded?

- How can unique keys generated by the database itself be guaranteed to be consistent across replicas?
- What should the topology connecting nodes be?
- Can communication between replicated database nodes be encrypted, to protect against data snooping and injection by a man-in-the-middle attack?
- Without a deep understanding of application behavior, database administrators cannot make appropriate decisions about these parameters and may configure a non-performant or unreliable database back end for an application.

In contrast to the above complexity, the database replication built into all Hitachi ID Systems products:

- Is included at no extra cost.
- Provides for encrypted communication between nodes out of the box.
- Performs reasonably well on wide area networks (not just high-bandwidth, low-latency local area networks).
- Can be very simple to configure, with a few settings on the product configuration web based interface. This is the case for both direct and shared schema replication. Even though hybrid topology is not very simple to understand and configure, requires complex maintenance steps, it still uses fewer network resources and is more adapted to the way *Bravura Security Fabric* works compared to native-database replication.
- Is fault-tolerant, queuing up updates made during failed network communication or other node's unavailability due to maintenance, until a peer application node becomes available again.
- Is asynchronous by default, and replicates only business/application-relevant data, offering the best possible performance for any given network conditions.

Hitachi ID Systems Recommendation:

Since replication is so valuable to Bravura Security Fabric deployments in general and to Hitachi ID Bravura Privilege in particular, Hitachi ID Systems strongly recommends that customers leverage the built-in replication capability and always deploy at least two database nodes (or when 100% uptime is required, at least three database nodes).

Hitachi ID Systems Recommendation:

In the case of Bravura Privilege, Hitachi ID Systems further recommends that the database nodes be installed as far apart as possible – preferably in different data centers, located in different cities that are not likely to be affected by the same natural disaster at the same time.

If the arguments above are not enough to sway the database administration team, or if the customer company has a hard requirement to use existing database-native replication or clustering:

- Setting up the replicated nodes at the different data centers using native database replica nodes will have to be done as if all nodes are in shared schema (since all application nodes will see the distributed database as the same one)

- Some functionality may break (because native database replication places locks on various parts of the schema as the data is replicated)
- Some product operations will be much slower (since stored procedures will have to wait for data to replicate as they run, and since ALL data changes will be replicated instead of only relevant data).

4.1.4 Replicated network topology alternatives

In this section we review stand-alone instances (one database node, not recommended), and the three supported and recommended instance topologies for the following ratios of database:application node replication:

- 1:1 - direct replication
- 1:N - shared schema replication
- M:N - hybrid replication

Replication architecture and the choice of replicated instance topology is up to the Solution Architect and managers who decide what amount of application usage and disaster mitigation they want to dedicate IT resources for.

4.1.4.1 Direct database node replication (1:1 data and application nodes)

Given that multiple application nodes will each have their own database node, there are several architectural possibilities:

Three or more database nodes: each database node replicates to all other nodes.

This doesn't mean linear replication. All application nodes are required to be DNS-resolvable and network-reachable from all other application nodes. All database replication ports (by default 5555 for data replication) to all application nodes have to be reachable, not blocked by firewalls or other network appliances or rules. Port 2380 for file replication has to be reachable from the primary application node to all other application nodes.

This topology is represented in [Figure 4.1](#).

Two database nodes: bi-directional replication

This cannot provide even theoretically 100% uptime, because an attempt to resynchronize the two databases when they fall out of sync will take both nodes down.

This topology is represented in [Figure 4.2](#).

No replication: One database node, with the database running on the same server as the application node or separately.

Though possible for demonstration and development purposes, Hitachi ID does not recommend using a single database node for production purposes, as it becomes a single point of failure.

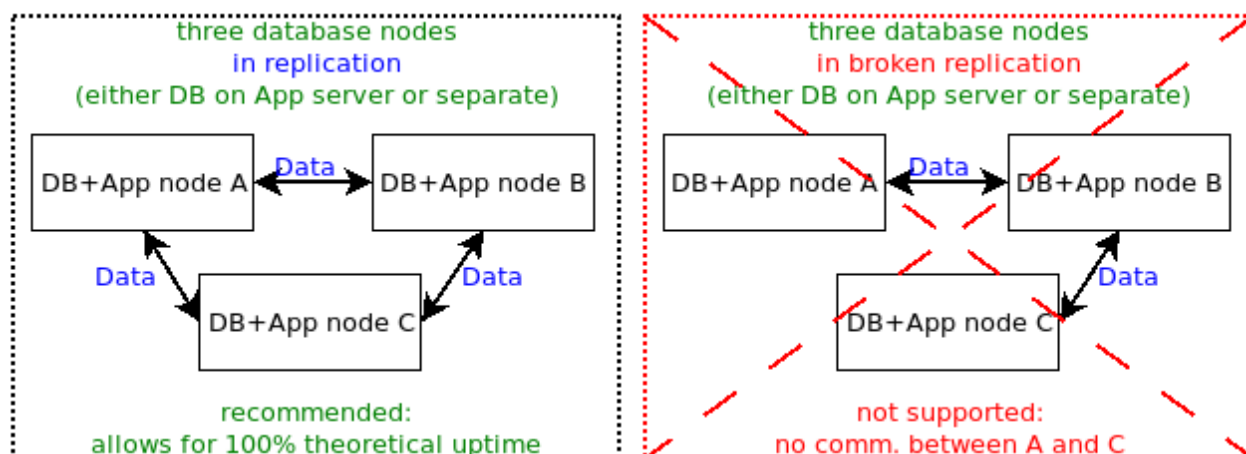


Figure 4.1: Network topology: Three or more database nodes with replication

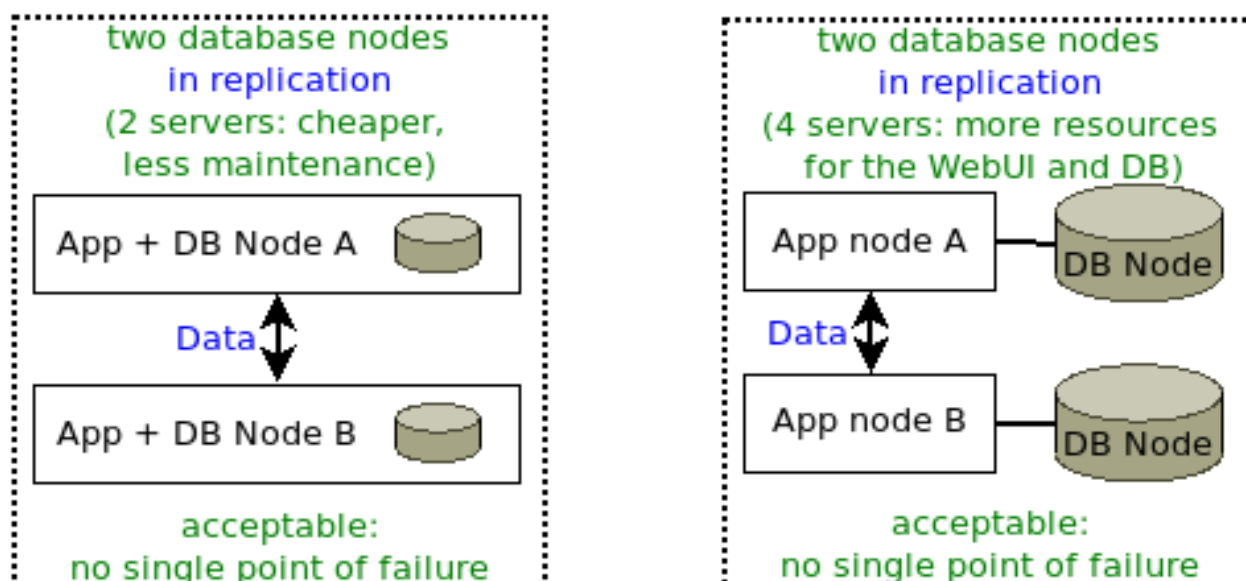


Figure 4.2: Network topology: Two database nodes with replication

This topology is represented in [Figure 4.3](#).

Hitachi ID Systems Recommendation:

Hitachi ID Systems recommends either two or three database nodes, at different locations.

It is important, no matter what topology is selected for a given *Hitachi ID Bravura Security Fabric* instance, to keep an up-to-date "topology document" that contains:

- The relationship diagram using simple names as topology designations (like DB-A for database node A; A-A1, A-A2 for application nodes 1 and 2 connected to DB-A; and so on.)

This diagram should also contain any networking devices used, like switches, WANs, VPNs, load

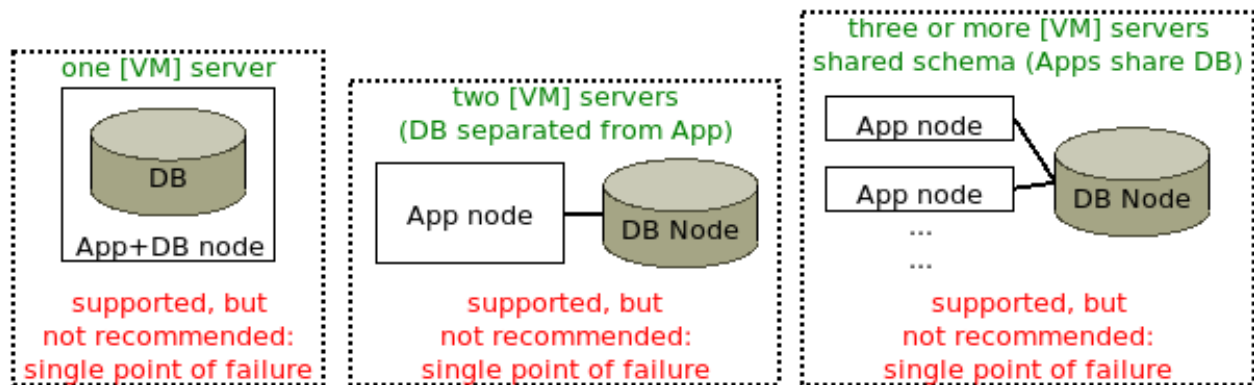


Figure 4.3: Network topology: One database node with no replication. This is good for development but not recommended mission-critical solutions. There is a single point of failure.

balancers, and so on, which could affect system communication and functionality.

- All the nodes (application and database), in a list containing:
 - Their topology designation
 - Fully qualified domain name
 - Geographical/data-center location
 - Any specific jobs and tasks assigned specifically to each node

4.1.4.2 Configuring replication nodes within a shared schema environment

Replication can be configured with more than one application node using the same backend database. This is *not* recommended for a production instance with a single database node, because that would not be fault tolerant (see [Fault tolerance: geographic distribution](#)).

When an application node is added to an existing database node, all nodes that share the schema will be listed on the **Database replication** page; for example:

- A *Hitachi ID Bravura Security Fabric* instance, MYPASS, is installed on two different servers: A and B.
- MYPASS (A) and MYPASS (B) are sharing the same data schema; that is, they store their data on the same database, or same database-native replicated database or cluster.
- If either MYPASS (A) or MYPASS (B) is added as a replication node, both will be listed in the same database node row in the **Database replication** page, as illustrated below.

In database migration cases when product administrators expect to find the node normally in direct replication, but forget to change the value from the schemaID table in the backend database, the product sees the node(s) with the same schemaID in shared schema instead.

Database replication

Node ID	Description	Address / Port	Mode	Status
hasit201212H2_default Shared schema: hasit201212L_default	node description	10.0.116.33 / 5555	Enabled	✓
<div> Delete Resynchronize Add new... </div>				

In a shared-schema environment, re-synchronization needs to be disabled when you add a node to replication. On the **Database Replication** page, the option to **Propagate and reload replication configuration on all servers (without resynchronizing nodes)** is available for this purpose.

To learn how to install a node in a shared-schema environment, see <https://docs.hitachi-id.net/#/home/5634/10/11>.

4.1.4.3 Combining database-native with application-level replication

The third available node topology is a hybrid (combination) of the direct application-level replication, and shared schema topologies introduced above. Each application node from a shared schema database node has to replicate with exactly one application node from all other database nodes; for example, in a hybrid replication topology with three application nodes providing web-based interface resources to each database node:

DB/Schema A	Replication		DB/Schema B
Node A1	←	→	Node B1
Node A2	←	→	Node B2
Node A3	←	→	Node B3

Crucially, it is not possible to eliminate one application node and configure another application to replicate to two other application nodes which are serving the other database. For example, it is not possible to eliminate node A3 and then configure node A2 to replicate with both nodes B2 and B3; doing so would violate the constraint that node A2 must replicate into schema B exactly once. If node A3 is removed and not replaced, node B3 must be removed as well.

The most complex configuration for replication is one where a database cluster with shared schema is combined with multiple application nodes. This is illustrated in [Figure 4.4](#).

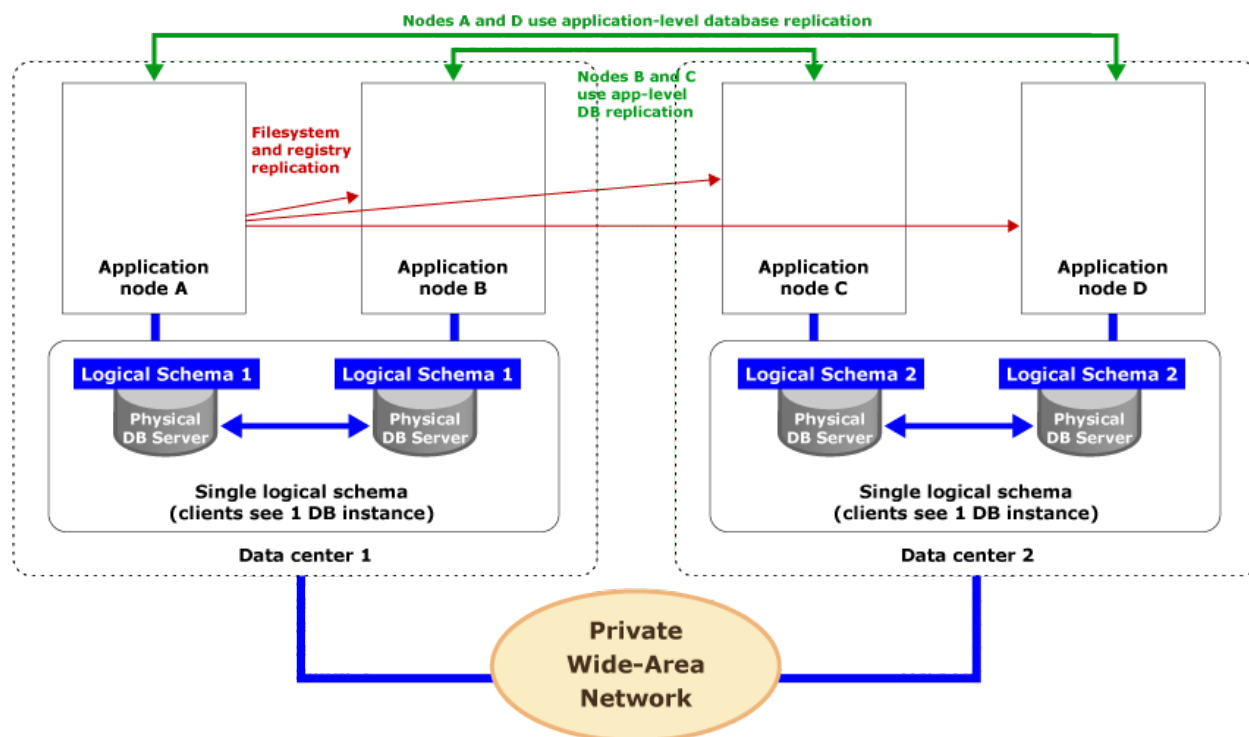


Figure 4.4: Example of hybrid topology configuration using two shared schema database clusters

In this scenario, database-native replication is used to create fault-tolerant database clusters at each site. Each of these clusters appears as a single, local database instance to every application node in the same data center. Application-level replication takes care of replicating changes between corresponding application nodes at different sites.

Operationally, when application node A processes an update using stored procedure P1 in data center 1, native database replication between the two physical database servers in the cluster at data center 1 forwards the call to this stored procedure from one database server to another. In other words, application node B will see the results of P1, regardless of which physical database it calls, as a consequence of database-native replication. At the same time, P1 is forwarded from application node A to application node D at another data center, to be written into the database cluster there. At the second data center, both application nodes will see the results of the update because of physically local native replication in that database cluster.

File system and registry replication takes place independently of all this. In [Figure 4.4](#), application node A is the primary node and pushes changes to its files/registry to all other application nodes and application proxies as a part of its nightly processing.

For more on file/registry replication, see [File Synchronization](#).

4.2 About the database replication process

Only transactions (stored procedures (sprocs) and their arguments) are replicated over to other nodes:

- The sprocs and arguments are sent to and queued on all db nodes at the time they happen on the node of origin.
- During **psupdate** some bulk data is replicated as well: diffsets.

If changes are made out-of-band directly to the databases (delete, insert or update table rows), those changes have to be applied separately to all nodes' databases, because the normal replication process described below will not send such changes over to other databases. This is a positive side effect that ensures that if some disastrous out-of-band process affects one database, the data in the other replicated databases will not be affected.

4.2.1 The process

When an operation is performed on one node (requested from the web-based interface or some automation), the sprocs triggered by various utilities, interface modules and services are sent from other binaries to the **iddb** service, that places them in a local queue, and sends them to all other node's **iddb** services, which in turn place them in their own local queues and execute them in the order they are received.

Those replication queues, both "send" (sq) and "receive" (rq), are kept in the instance's db\replication directory of every node engaged in database replication.

The settings for those queues are recorded in a node's service\iddb.cfg and can be changed in the web-based interface from **Manage the system** → **Maintenance** → **Database replication** and click a nodes **Configuration** tab.

The queues are not deleted when replication is suspended or disabled or nodes are removed, so if they are not empty when replication is re-enabled, **iddb** will continue to play back their content.

There are several ways for these transactions will not be replicated, thus leaving the databases in desynchronized states; see below.

4.2.2 Problematic states

In order to optimize replication, make it faster and reduce the amount of data transmitted, there are a couple of problematic states which can result if an application or database node is running out of resources (CPU, RAM, disk space, very rarely extreme network latency or other abnormal network states) that can cause delays and failures. These can lead to the database nodes becoming more and more out of sync over time (desynchronized).

4.2.2.1 Causes for data replication delay

If the local database or one of the remote databases is busy, or the tables required for one sproc are locked by another sproc or by the db engine (maintenance) operation a replication delay can occur.

In these cases, the data they create, update or remove may not be available in the affected node's web-based interface on the nodes where it was not updated.

The queue delay increases when some sproc execution plan dead-locks and that sproc locks tables needed for other sprocs:

- Usually, a sproc deadlock is found by the database engine and reported back to the client, which reports it to **iddb**, which logs it and then retries. When that does not happen, or if something slows down processing at the database side, the sproc queues on all application nodes using that database as a backend start growing.
- Eventually the limit of running threads on the database engine is reached and the database client cannot send its server other sprocs.
- At that point, **iddb** queues them. In **Manage the system** → **Maintenance** → **Database replication** → **<node>** the "Queue empty" value changes from the good value of "Yes" to "No".
- A good indication of how long this has been happening is in the "Time since last queue item was processed (seconds)" value.

4.2.2.2 Replication failure

Replication failure occurs if the data on one node is already missing required dependencies from other tables. The execution of those changes can fail independently on either of the nodes, including the source node (see "desynchronized" below).

When the **iddb** service processes data as intended, sprocs that fail on a specific node (and are supposed to be sent for replication), are recorded in the instance's `db\ iddb-failed-procs*.log` files:

- These files are not rotated like the normal logs; instead, if the files are not empty, they show up as critical errors in the healthcheck administrative interface dashboards on each node.
- When failed-sprocs files are not empty, their contents are supposed to be checked by administrators as part of normal maintenance, and emptied.
- Those contents have to be preserved outside of the instance file structure for later analysis.
- If the failed-sprocs affect the functionality of the product, they may have to be re-played on the node where they failed or on, after correcting the situation that made them fail in the first place.
- Each row in those text files contains: timestamp of the occurrence, the sproc name, the module that called the sproc and the (short, not helpful) summary of the error as returned to **iddb** from the database engine.

- Many failed sprocs during **psupdate** are irrelevant if they do not re-occur on subsequent runs, because *Hitachi ID Bravura Security Fabric* is engineered to self-heal to some extent.

These failures have to be addressed within seven days of occurrence. This is the default number of days the `idmsuite.log` is preserved. The *Bravura Security Fabric* log contains a more detailed description of the error as returned from the database engine. The administrator can search for "Native Client" (case sensitively), to find any such records in the log. These entries detail the error, including what caused it, as best as the database engine can determine. To interpret them, experience with the product and a knowledge of the database's schema and constraints is required.

4.2.2.3 How to address failures

If the failed-procs files contain few rows, send the files to support@Hitachi-ID.com to create a ticket and have the developers decide on a solution.

The `dbcmd` utility does pass the sproc and its argument to `iddb`, however this is not recommended as a way to "replay" sprocs which failed previously. Sending the sprocs back to `iddb` could cause them to replicate again, which in some cases, guarantees their failure on the nodes where the sproc already succeeded.

If too many of these sprocs have failed, either during a disastrous event or over time while no administrator addressed them, the database containing most of the common data will have to be propagated to the other nodes.

See the *Bravura Security Fabric* Migration Reference Manual for various propagation methods.

4.3 Replication in detail at a service level

The built-in application-level replication between *Hitachi ID Bravura Security Fabric* nodes forwards calls to stored procedures that are known to alter the state of the database. Calls to stored procedures that do not alter the user data or product configuration are not replicated.

To describe the replication process, a reference model will be used:

- Assume there are two application nodes (A-A and A-B).
- Each application node has its own database node (DB-A and DB-B).
- Each application node has one writer thread responsible for writing updates to its local database that were initiated on another node (W-A and W-B).
- The writer threads (W-A and W-B) each have their own queues on the local file system. These are used to retry database stored procedure calls that failed due to connectivity or database problems.
- Each application node also has one propagation thread per replicated peer, responsible for sending updates to peer application nodes. Thread P-A runs on A-A and sends updates to W-B. Thread P-B runs on A-B and sends updates to W-A.

- The propagation threads (P-A and P-B) each have their own queues on the local filesystem. These are used to retry connections to peer application nodes (W-A and W-B) in the event of a connectivity or service outages.

This arrangement is highly reliable – it connects simple components in a fault-tolerant fashion, as illustrated in Figure 4.5.

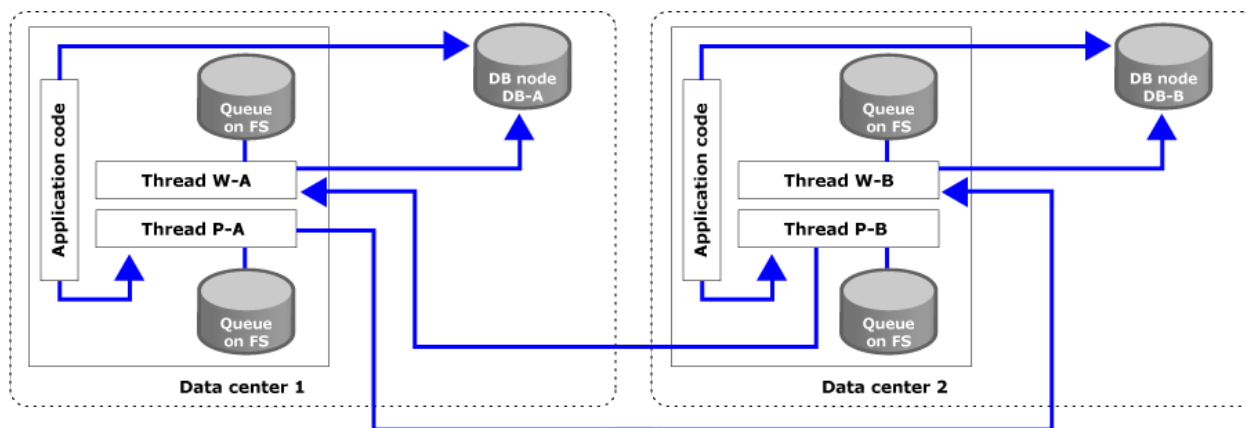


Figure 4.5: Data flow between components of a two-database-node replicated *Bravura Security Fabric* instance

At runtime, replication operates as follows:

1. The user or an automation process makes a change in relevant data on application node A-A.
2. The application on A-A calls a stored procedure P1 on DB-A.
3. A-A also queues P1 for replication to DB-B through P-A.
4. If transmission from P-A to W-B fails, P-A will continue to retry the transmission – either when the next stored procedure arrives (that is, try to send {P1,P2}) or after five minutes, whichever happens first.
5. Every transmission failure is logged on A-A's `idmsuite.log`.
6. All communication between nodes is encrypted using a shared-key handshake with the AES encryption algorithm. Both endpoints are authenticated in the handshake.
7. P1 will only be removed from P-A's queue once transmission to W-B has been confirmed. Confirmation in this case means that W-B received P1 and inserted it into its queue, not that it has completed executing P1 on DB-B.
8. W-B will process transactions, including P1, from its queue and attempt to issue them to DB-B.
The transaction includes its source, so the audit data will contain the name of the application node where the transaction originated, no matter on which database it is written.
9. If DB-B reports successful execution, P1 will be removed from W-B's queue.
10. If DB-B reports an error when attempting to run P1, such as an attempted violation of an integrity constraint, the full error, as reported by the database client, will be logged in `idmsuite.log` on A-B.

A summary of the P1 store procedure call (including a timestamp of the failure, the name of the stored procedure, its arguments and a short failure string) will be recorded in a `sproc-failure` log in the `db\` directory of the instance on A-B. After that, P1 will be removed from W-B's queue.

11. If W-B experiences a database connectivity error (that is, W-B cannot contact its database client, or its database client fails to send P1 to DB-B) then P1 will not be removed from W-B's queue. W-B will retry database updates every 30 seconds until connectivity to DB-B is re-established..

4.4 What is not replicated

In general, reports and session data are *not* replicated.

For a full list of files that are excluded from file replication, run the `updinst` program, located in the util directory, with the `-showconfig` option:

```
updinst -showconfig
```

You can modify the whitelist and blacklist which determines which files get replicated and which don't. See [updinst](#) for details.

The following stored procedures are not replicated:

- ReportSet
- ReportDelete
- SessionClean
- SesskeyExpiryUpdate
- SesskeySet
- UserSessionDelete
- ReportStageAdd
- ReportStageDelete
- SchemaldSet

This affects the following tables:

- report
- sesskey
- reportstg
- schemaid

4.5 Impact of environment

Normally, the application-level replication is fast and efficient. However, several environmental measurements can affect the delay of data and configuration changes made on one node being available on the rest of the replicated application nodes.

For example, if a database deadlock does not allow changes to be saved on a database, none of the application nodes connected to that database, even the one where the change originated, will not be able to display the change.

The subsections below are listed in the order they can impact the change recording transmission delay, from more to less:

- Database server issues
- Application server resources
- Network latency
- Bandwidth constraints

4.5.1 Impact of database server issues

There are many issues that can impact the proper functionality of the back-end database, and that's exactly the source of the majority of product reported issues and outages.

Database server resources The database server has to be correctly provisioned and its requirements as specified by Microsoft have to be met. There are some notes about what can affect application services in the next section, and for the database, the important resources are RAM size and speed, single-CPU speed and disk I/O.

Free disk space available has to be 1.5 times the size of the database files. Even more space is needed on the disk where the database's tempdb files are stored, depending on the frequency of maintenance. During auto discovery, several large tables are completely replaced, which leads to large transactions, with tempdb files growing and shrinking very fast.

Database configuration Follow procedures as detailed in <https://docs.hitachi-id.net/#/home/2209/10/11>. If your database administrator does not agree with the settings, check with support@Hitachi-ID.com on what impact any deviation from the listed configuration may have.

Database-user mapping If the database Login which is configured for use by the *Hitachi ID Bravura Security Fabric* does not have enough privileges or even has sysadmin role granted, it can fail to run stored procedures in the correct context, or not even find them. The correct settings are also listed in <https://docs.hitachi-id.net/#/home/2209/10/11>.

Database maintenance This is important, especially after the count of table row passes the one million mark. Maintenance like reindexing, defragmenting, cleanup of obsolete historical data, and so on, can double or triple performance. Such maintenance has to be scheduled outside of working hours for most users, and also outside of application-scheduled tasks which change the database drastically, like `psupdate`, `idtrack`, `autores`, and so on.

- Ideally, the Database Service (iddb) on all the application nodes that use a database where maintenance is performed, should be stopped and disabled in order to prevent table and cross-sproc locking, and to avoid changing indexing while an already-calculated complex sproc execution plan is already running.
- It is also recommended to wait for the replication queues to empty after database maintenance completes and the affected application nodes are brought back online, before taking down another database node for maintenance.

Database schema and stored procedures These are also very important. They are designed with existing use-cases in mind, and as time and versions go by, both are improved and optimized for the types of loads and data patterns our clients require. The MSSQL Activity Monitor tends to point to "missing indexes" or "better index configuration" whenever long-running sprocs are involved. However, in most cases those "solutions" would increase the database size considerably.

4.5.2 Impact of application server resources

The first step toward a performant *Hitachi ID Bravura Security Fabric* instance is to provision all production application servers according to <https://docs.hitachi-id.net/#/home/2218/10/11>.

The following server resources are important:

- Multi-CPU (the more CPUs and faster, the more binaries can run at the same time)
- Disk I/O is crucial for the application server. Services do cache most files they need as they start up, but a large part of the automation in the application will use some sort of scripting in Python and components, which are all loaded on demand. Low disk latency (imposed by network conditions at the data center) can easily slow down any automated process, especially when millions of records are read from list databases;
- RAM is important from its absolute minimum of 4GB to an acceptable value of 8GB available to leave space for troubleshooting tool to run when needed.
 - In the current version, RAM use will peak during psupdate and skin generation (node.exe by itself will grow to 1.2GB as it compiles the various Angular apps that make up the web-based interface).
 - Under high use (60-100 users per node at the same time), the Ajax service can grow to 2GB or more.
 - The more concurrent agents are to be run from the primary or from a proxy server, the more RAM will be needed;
 - Starting with *Bravura Security Fabric* version 12, list files from target systems are SQLite databases instead of structured text files, which will increase the listing and filtering performance, at the expense of more RAM use per agent, for the **iddiscover** service, and for any other process that handles list files.

4.5.3 Impact of network latency

In the context of file/registry replication, each transmission is followed by acknowledgment from the peer application node. This means that the impact of replication is approximately a product of the number of files to be transferred times the packet response time. For example, to replicate 100 files over a 100ms network, a total of 20 seconds of packet latency overhead is introduced per pair of replicated nodes.

In the context of database replication, each batch of stored procedure calls is impacted by the latency twice per TCP receive window size. The TCP receive window size is an operating system parameter, typically between 8KB and 64KB. In general, throughput drops as latency increases, but this is not normally a problem so long as latency is well under 500ms per packet and overall transaction volumes are “normal.”

Hitachi ID Systems Recommendation:

Hitachi ID Systems recommends placing all Hitachi ID Bravura Security Fabric servers which made up a database node at locations with no more than 150ms latency between them.

In practice, high network latency (where application nodes are on opposite sides of a continent or separated by an ocean) has the following impact:

- The time required to complete file/registry replication during nightly auto-discovery will grow by several minutes per pair of replicated application nodes.
- The time required to complete database replication for large volumes of data – that is, during nightly auto discovery and under heavy load conditions – can grow from seconds to minutes.

4.5.4 Impact of bandwidth constraints

File/registry replication is impacted by bandwidth between replicated nodes in the sense that the time required to transmit changes from one node to another will be at least the size of the change divided by the bandwidth available. For example, if a 100MByte executable is installed on the primary node and 10Mbps is available between application nodes, then the time to transmit the change will be at least:

$$T = (100 \times 10^6 \times 10\text{bits/byte}) / (10 \times 10^6) = 100\text{sec} \quad (4.1)$$

In practice, latency adds further delay to this calculation, as described in [Impact of network latency](#), so in the best case scenario using the data above, the actual time would be more than 100sec.

Similarly, database replication is impacted by bandwidth by limiting the rate at which stored procedure calls can be forwarded from one node to another. Since stored procedures arrive at each application node sporadically, they are more impacted by latency, which adds a “fixed cost” to each batch of stored procedures. In practice, it is only transfers of relatively large data sets – for example during auto-discovery, that are noticeably impacted by bandwidth.

Hitachi ID Systems Recommendation:

Hitachi ID Systems recommends placing Hitachi ID Bravura Security Fabric database nodes at locations with at least 5Mbps bandwidth available between them.

In practice, low network bandwidth, where application nodes have less than 1Mbps of bandwidth available to propagate changes from one to another, has the following impacts:

- The time required to complete file/registry replication during nightly auto-discovery will grow – with the delay being determined by (a) the volume of data that needs to be forwarded and (b) the available bandwidth.
- The time required to complete database replication for large volumes of data – that is, during nightly auto discovery and under heavy load conditions – can grow. In some cases (that is, high load (for example, Gigabyte volumes), very low bandwidth (for example, 100kbps)) a substantial backlog can develop.

4.5.4.1 Estimating bandwidth requirements

Nightly auto-discovery

The bulk of data transmission between application nodes during the nightly auto-discovery process is to transfer list files from the primary application node, where they are generated, to all other nodes. Since compression is used, on average, the total data transmitted will generally be less than half of the disk space consumed on the primary node by these files.

For example, if lists of users, groups, account attributes and computer groups on the primary node consume 50MB of disk space then no more than 25MB of network bandwidth will be used during nightly auto-discovery to transfer this data set to each secondary application node.

Real time database replication

The volume of data replication between servers depends on the workload generated by each server. Some rough rules of thumb are:

- With *Hitachi ID Bravura Pass*, every user login session, either to change passwords or enroll security questions, will generate about 29 replicating procedures.
- With *Hitachi ID Bravura Identity*, every workflow request (input, approvals, fulfillment) will generate about 200 replicating procedures.
- With *Hitachi ID Bravura Privilege*, every scheduled password randomization will generate about 20 replicating procedures.

4.6 Load balancing

Hitachi ID Bravura Security Fabric has a fundamentally multi-master architecture. That means that two or more application nodes are typically active at any one time, able to service user requests, execute unattended processes, and so on.

To distribute incoming workload to multiple servers, either a TCP/IP based load balancer or a DNS round-robin scheme can be used.

Once a user's web browser is connected to a given *Bravura Security Fabric* application node, the session should be bound to that server via a "sticky sessions" mechanism. This is because session-specific data is **not** replicated, for performance reasons. A user can initiate a new session on a new application node (if a different one is available in the same geographic region) but should complete an already-initiated session on the node where that session started.

If using a TCP/IP based load balancer, sticky sessions are a required configuration item.

If using DNS round-robin, sticky sessions can be implemented by having users' login sessions redirected from a common virtual directory (shared by all *Bravura Security Fabric* servers) to a server-specific URL. That is:

1. User accesses `http://identity-manager.acme.org/`
2. The DNS server resolves this to the IP address of one of the servers, selected at random – namely: `http://10.0.100.1/`.
3. The user's web browser is redirected by the web server on the application node to a new URL: `identity-manager-1.acme.org`.
4. The user's web browser resolves this always to the same IP – for example: `http://10.0.100.2/`.
5. All further interaction is between the user and the URL `http://identity-manager-1.acme.org` at `http://10.0.100.2/`.

In addition to load balancing incoming workload, with *Hitachi ID Bravura Privilege* it is possible to associate each application node with geographically nearby managed systems. This is done by setting the "Managed by" application node on each managed system policy.

In the event of a permanent hardware failure on a single node, it may be necessary to reassign the node affinity of managed system policies. See [Single Bravura Security Fabric server fails](#).

Also, for both *Hitachi ID Bravura Identity* and *Bravura Privilege*, each system has its own instance of the workflow manager service. Requests submitted to a given workflow manager are processed by the same workflow manager throughout their life cycle. For example, a request to provision a user on application node A will lead to the workflow manager service on application node A to send email invitations to authorizers, to invite implementers to complete tasks, and so on.

In the event of a permanent hardware failure on a single node, it may be necessary to reassign the node affinity of in-flight workflow requests.

Finally, auto-discovery is the responsibility of a single node, which is designated the primary. Responsibility for running the nightly auto-discovery process is the main distinguishing characteristic of the primary application node, as compared to secondaries.

In the event that the primary application node fails, permanently, a secondary application node can be assigned this responsibility by adjusting the list of scheduled jobs on that node to include `psupdate`.

See [Automated node check when using a load balancer](#) for important details on monitoring application nodes status from the load balancer.

See [load-balancer-config.pdf](#) for specific details on configuring a F5 LTM load balancer for *Bravura Security Fabric*.

Summary

Hitachi ID Bravura Security Fabric includes two simple to configure, secure and fault tolerant replication mechanisms – one for database updates and one for file/registry changes. Hitachi ID Systems customers are strongly urged to take advantage of these features to deploy a multi-master architecture in all *Bravura Security Fabric* deployments, and especially when deploying *Hitachi ID Bravura Privilege*.

Physical distance between *Bravura Security Fabric* servers is recommended, to minimize the impact of a physical disaster at one data center. This recommendation is only bounded by reasonable constraints on bandwidth (10Mbps is recommended) and latency (up to 150ms between sites).

4.7 Configuring database replication

Configuring *Hitachi ID Bravura Security Fabric* in a replication environment offers the following advantages:

- In the event that a primary server crashes, a backup server can take over *Bravura Security Fabric* functions
- Multiple servers allow for load balancing

Hitachi ID Systems *strongly* recommends that you create at least three replicated *Hitachi ID Bravura Privilege* database nodes for fault tolerance and backup. See <https://docs.hitachi-id.net/#/home/5632/10/11> to learn how to calculate the number of nodes required.

You can configure your replication environment in multiple ways depending on your requirements. See [Database Replication](#) for detailed information.

Each server on which the *Bravura Security Fabric* main application is installed is referred to as an *application node*. The node ID is automatically generated in the format `<servername>_<instancename>`.

As a result, all application nodes set in replication must be installed with the same instance name. The name is case sensitive (`Instance_Dev` is different from `instance_dev`); Hitachi ID Systems suggests keeping the instance names in lower case only.

Life cycle of an application node

A *Hitachi ID Bravura Security Fabric* application node typically follows the following life cycle:

1. Preparation
2. Adding a node
3. Checking replication status as an Administrator
4. Automated node check when using a load balancer
5. Modifying node settings
6. Removing a node from replication
7. Cleaning up replication configuration

4.7.1 Preparation

Requirements

Before setting up a replication environment, ensure that:

- All server clocks are synchronized.
- Only one user configures the replication servers at any point.
No other users should be logged into any of the other *Hitachi ID Bravura Security Fabric* servers while the replication environment is configured.
- The product administrator who is configuring replication has the right to maintain servers.
- All servers have the same instance name and virtual directories.
- All servers use the same *Connector Pack* version and *Connector Pack* type.
- All servers have the same installed components.
- All replication configuration changes are made from the primary server; that is, the server that runs auto-discovery.
Once auto-discovery has run on the primary server, the gathered information is replicated to other nodes. This may take some time. Concurrent changes can cause race conditions in a replicated environment.
- The Microsoft SQL versions are the same on all servers, including the same edition and service pack level.
When upgrading Microsoft SQL, schedule the upgrades on all databases as close together as possible, because replication will be stopped while the application nodes are served by different version databases.

Note: The email server configuration must be manually set up on each individual node in the replication environment.

Recommendations

Hitachi ID Systems recommends:

- All servers use the same *Bravura Security Fabric* license.
- All nodes are installed with the same directory paths, psadmin user and password.
- Using the `idmsetup.inf` file from the main server to aid the installation of the replication node.
- Not using database-level clustering (including AWS RDS Always-On Availability Groups).

4.7.1.1 Installing Bravura Security Fabric using idmsetup.inf

When you install *Hitachi ID Bravura Security Fabric* on the main server, an `idmsetup.inf` file is created in the `\<instance>\psconfig\` directory. You can use to file to aid the installation of a replication node. It contains the encrypted communication key (or Master Key) and other correct default values and encryption keys for prompts during installation.

To install a replication node using the `idmsetup.inf` configuration file:

1. Log into the host Windows server as member of the Administrators group.
2. If required, download and unzip the *Bravura Security Fabric* distribution folder.
Contact your Hitachi ID account representative for details.
3. Copy `idmsetup.inf` from the `\<instance>\psconfig\` directory on the primary server to the replication node and place it in the root of the distribution folder.
4. Launch the `setup` program located at the root of the distribution folder.
The `setup` program asks you to choose a product to install.

BEST PRACTICE Do not install *Connector Pack* first; choose *Bravura Security Fabric*.

5. Select ***Bravura Security Fabric***.
Click **Next**.
6. Verify that instance name is already filled in and optionally enter a description.
Click **Next**.
7. The `setup` program performs a pre-installation check and verifies all of the requirements for installation.

8. If all of the checks are successful, click **Next** to proceed with the installation.

Note: If any of the pre-install checks produce warnings or errors, click **Report** for details.

The **Configure a Dedicated Database User** page is displayed.

9. Choose your database user setup option, then click **Next**.

- Create an configure a new dedicated database user for a new database node, as illustrated below.

On the **Configure a Dedicated Database User** page, when you select **Create a new dedicated database user for the new instance**, **setup** displays the database user and connection information page depending on your database system.

Enter the database connection and user information, then click **Next**.

- Use an existing database user when creating a shared schema node, or a hybrid schema node if the database node already exists.

On the **Advanced** tab uncheck the options to install the schema and default data.

The **setup** program launches **idm.msi**.

10. Click **Next**.
11. Read and accept the license agreement.
Click **Next**.
12. Verify the location of the license file already filled in.

Note: It is recommended that all replicated servers use the same license file.

Click **Next**.

13. Choose the setup type:

- **Typical** to accept the settings from the `idmsetup.inf` file.
- **Custom** to customize settings.

Note: It is recommended that all replicated servers have the same installed components.

14. Verify that the psadmin passwords are already filled in (masked with stars).

Click **Next**.

15. Verify that the communication key is the same as the one entered for the main *Hitachi ID Bravura Security Fabric* server.

Click **Next**.

16. Verify that the database encryption key is the same as the one entered for the main *Bravura Security Fabric* server.

Click **Next**.

17. Verify that the workstation authentication encryption key is the same as the one entered for the main *Bravura Security Fabric* server.

Click **Next**.

18. Verify that the connector encryption key is the same as the one entered for the main *Bravura Security Fabric* server.

Click **Next**.

19. Verify that the IDMLib encryption key is the same as the one entered for the main *Bravura Security Fabric* server.

Click **Next**.

20. Verify the administration login ID and password.

Click **Install**.

21. Click **Finish** to exit.

The post-installation tasks begin.

CAUTION: Do not stop the post-installation tasks. The installer is attempting to load connectors from the *Connector Pack*, language tags, and reports.

If you install the *Bravura Security Fabric* before a *Connector Pack* has been installed (recommended), a warning appears at this stage that no connectors could be found. It is safe to proceed.

If any of the post-installation tasks produce warnings or errors, click:

- **Report** for details on all post-installation tasks
- Or,
- **Messages...** for details on a specific post-installation task

Otherwise, wait until the status changes to *success*, then click **Finish**.

Run **setup** again to install a *Connector Pack* for this instance (unless you installed a global *Connector Pack* first, which is not recommended, as it can introduce maintenance issues).

Install all software required for integration with target systems and for accessing all resources. Configure the firewall on the new server and other required networking that allows access to needed resources.

See also:

- The setup of the application node's *Hitachi ID Connector Pack* should match the setup of the primary node's *Connector Pack*. See the *Connector Pack Integration Guide* for details.

4.7.2 Adding a node

To replace an application node that has failed, or add a net-new application node:

1. Install the hardware (physical or virtual), operating system and database software on the new node. Apply whatever patches are available from the OS and database vendors. See [Preparation](#).
Note that everything must be the same on the replicated nodes, including: OS version and bitsize, application locations, database versions, instance name (spelled exactly the same, case sensitive), software required to communicate with target systems, and other integration requirements.
2. Verify that there is connectivity on the *Hitachi ID Bravura Security Fabric* ports, bidirectionally, from all other node servers to the new server (usually, this involves opening firewall ports and/or setting up VPN connections between sites/data-centers to allow network routing between the servers on those ports).
3. Install the *Bravura Security Fabric* software on the new application node. See [Installing Bravura Security Fabric using idmsetup.inf](#).
4. On the application node that contains the most data (usually the primary node):
 - (a) Add the new node to the replication configuration. See [Adding a node details](#).
 - (b) Propagating the replication configuration will cause the current instance to synchronize its database with the new instance. See [Propagating replication changes](#).
 - (c) Synchronizing data is optional (sending the relevant data from the database node where the operation is performed, to the database node being added); for example, when adding a node in shared schema, or when the database is very large and it's more efficient to migrate the database, sending the data is not required. See [Configuring file synchronization](#).

More detail is provided in [Recovering from Network Faults](#). You should also refer to native database administration tools and their related documentation for the process to generate and load snapshots.

After finishing any changes, remember to verify the replication configuration. See [Cleaning up replication configuration](#).

4.7.2.1 Adding a node details

CAUTION: Before starting this procedure, ensure that you have created the new server according to the requirements described in [Preparation](#).

Verify that there are no database-intensive scheduled tasks that can be triggered on any of the database nodes of the instance if database resynchronization will have to run:

1. On the web-based interface, click **Manage the system** → **Maintenance** → **Scheduled jobs** → **PSUP-DATE** → **Disable**.

Repeat that for any other jobs like **autores** if it was scheduled.

Disable **rotatelog** to avoid the log rotation to remove the temporary **iddb.cfg** before the operation ends and to be able to find the logs of the operation in the same file.

2. On each application node's Windows Scheduler → Library, right-click and disable any "HID*" task.
3. Disable any other database-intensive API automation that is scheduled elsewhere.

Remember to re-enable these tasks after the database resynchronization successfully completes.

Depending on your chosen replication network topology, there are different steps for adding an application node for:

- [Direct replication](#) (p38)
- [Shared schema](#) (p41)
- [Hybrid schema](#) (p41)

4.7.2.2 Adding a database node in direct replication

A database node is an application node connected to its own database, not shared by any other node.

To add a new database node (application node connected to its own database, not shared yet by any other application node) to *Bravura Security Fabric* replication environment:

1. Log onto your primary node as a product administrator.

WARNING!: Adding a node from a replicated node will result in the new node overwriting existing nodes and becoming the primary node.

2. Click **Manage the system** → **Maintenance** → **Database replication**.
3. Before the very first node can be added, you have to enable replication on the primary node. To do this, select the default node and set **Mode** to **Enabled** and click **Update**.

4. To add the next node click **Add new...**
5. Type the **Address** of a replicating instance's server and the corresponding Database Service (iddb) **Port** number (the default port number is 5555).
6. Click **Continue**.
7. Type a node **Description**.
8. Select the **Mode** to:
 - **Enabled** if you want this node to send and receive information to and from other replications nodes.
 - **Suspended** if you want to temporarily stop communication with other nodes.
If the replication is suspended on a node, then changes made on other replication nodes will not be propagated to this node, but will be queued.
 - **Disabled** if you want to disable communication with other replication nodes.
9. Set the received and/or send **Queue size limit** , if required.
When database replication is unavailable, replication data is queued until the queue file has reached its size limit, or the node comes back online. You can modify:
 - **Receive queue configuration for this node**
 - **Send queue configuration for all nodes replicating to this node**

You can set the queue limit to a fixed size, using the following settings:

 - **Minimum queue size** - default is 100MB
 - **Maximum queue size** - for fixed size limit
 - **Queue usage warning threshold (%)**: - default is 60%


You can set the queue limit to a percentage of disk, using the following parameters:

 - **Minimum queue size** - default is 100MB
 - **Maximum usage before queue stops growing (%)**: - default is 90%
 - **Disk usage warning threshold (%)**: - default is 60%
10. Click **Add**.
11. Click **Propagate and reload replication configuration on all servers**.
This propagates the changes and replicates the entire database to the new node. If you do not want data on the newly added node to be dropped and replaced with data from the current node, click **Propagate and reload replication configuration on all servers (without resynchronizing nodes)**.

Propagating replication changes

Once you have modified the configuration of a node in a replication environment, you must propagate the node configuration to all connected servers. The status for the nodes will change to "Reload required" on the **Select a replication node** page.

To do this:

1. Ensure all nodes can successfully communicate with each other. When propagating changes, the system checks that all nodes can connect, and if any connection issues are detected, propagation will fail.
2. If adding a new database node, verify there is enough space available on:
 - The source application node (the one from whose web-based interface the propagation is triggered)
 - The target nodes (the ones where the propagation goes)
 - Their respective backend databases.
 See [Resynchronizing databases](#) for details.
3. Click **Propagate and reload replication configuration on all servers** on the *Database replication* page.
 This propagates the changes. When a new node is added the entire database will be replicated to the new node.
4. Wait while the Database Services reload the changes and possibly builds queue files.
 Click Refresh  to check progress, as indicated in the status column.

Note:

- If a node is in “Reload in progress” status, or is running auto discovery, you cannot make any changes to the replication nodes until the reload is complete.
- The **Propagate and reload replication configuration on all servers** only appears if a change in the configuration has occurred.

Configuration notes

- When you configure database replication, *Hitachi ID Bravura Security Fabric* creates a temporary **iddb.cfg** file in the log directory. The files record the configuration information for backup. The information is rotated with the log files by the Logging Service (idmlogsvc).
- If a server detects that one of the nodes has a different replication configuration, then a message opens to notify you of the need to propagate this replication configuration over to the other server.
- If you change which server is the primary server, you must update which server’s scheduler will run **psupdate**. For details on how to do this, see [Selecting a server to run a job on](#).
- Do **not** propagate changes while auto discovery (**psupdate**) is running.
- When propagation changes include queue sizing, the queues are adjusted dynamically. Queues files that still include data will remain until the data is either sent or committed, regardless of queue settings. When a new queue file is needed or a queue file is no longer needed, then the queue sizings are considered.

4.7.2.3 Adding an application node in shared schema

When adding an application node to an existing database all that is required is to install the node while selecting the existing database of a previously installed application node, as described in <https://docs.hitachi-id.net/#/home/5634/10/11>. This is the simplest way to create a shared schema node.

If a database has to be retired but its application nodes are still needed to provide web-based interface resources for users, this can also be done after node installation:

1. Have a fully installed database node (application node with its own database)
2. On the new application node, use the `iddbadm` utility to change the database connection details and point to the database used by the other node.
3. Delete the database used when creating the new application node. In this case it is not used anymore.

Note: If you allow `setup` to create the new database during installation, it will do so with the same name as the instance, and you *cannot* use the same backend database instance to install two databases with the same name.

Propagating without replication

In some cases, such as a shared-schema replication environment, you may need to propagate changes to the replication environment without initiating a full database resynchronization. On the **Database Replication** page, the option to **Propagate and reload replication configuration on all servers (without resynchronizing nodes)** is available for this purpose.

4.7.2.4 Adding an application node to a hybrid schema

There are two ways to add nodes to a hybrid schema:

- Adding a database node with application nodes
- Adding one application node to each of the existing database nodes

Adding a database node with application nodes

To add failover, further geographical support, nodes in a new data center and/or complete 100% uptime requirements, add a database node with the same number of application nodes as the other existing database nodes; that means going from this topology:

DB/Schema A	Replication	DB/Schema B
Node A1	<— —>	Node B1
Node A2	<— —>	Node B2

... to this one:

DB/Schema A	Replication	DB/Schema B	Replication	DB/Schema C
Node A1	<— —>	Node B1	<— —>	Node C1
Node A2	<— —>	Node B2	<— —>	Node C2

To do this:

1. Consider A1 is the primary node of the instance
2. Create node C1 as a full database node pointing to Database C and using the instance's `idmsetup.inf` (see [Installing Hitachi ID Bravura Security Fabric using idmsetup.inf](#)).
3. Create node C2 as a shared schema node pointing to Database C (see [Adding an application node in shared schema](#)).
4. From A1's web-based interface click **Manage the system** → **Maintenance** → **Database replication**, add C1's server and port and propagate changes without replication (see [Propagating without replication](#)).
5. From A2's web-based interface click **Manage the system** → **Maintenance** → **Database replication**, add C2's server and port and propagate changes without replication (see [Propagating without replication](#)).

Adding one application node to each of the existing database nodes

To add web-based interface and/or automation resources, add a new application node to each of the existing database nodes; that means going from this topology:

DB/Schema A	Replication	DB/Schema B
Node A1	<— —>	Node B1
Node A2	<— —>	Node B2

... to this one:

DB/Schema A	Replication	DB/Schema B
Node A1	<— —>	Node B1
Node A2	<— —>	Node B2
Node A3	<— —>	Node B3


To do this:

1. Consider A1 is the primary node of the instance
2. Create node A3 as a shared schema node pointing to Database A (see [Adding an application node in shared schema](#)).

3. Create node B3 as a shared schema node pointing to Database B (see [Adding an application node in shared schema](#)).
 4. From A3's web-based interface click **Manage the system** → **Maintenance** → **Database replication**, add B3's server and port and propagate changes without replication (see [Propagating without replication](#)).
-

4.7.3 Checking replication status as an Administrator

To view status information about a replication node:

1. Click **Manage the system** → **Maintenance** → **Database replication**.
2. Select  a node.
3. Select the **Status** tab.

The status page includes information about:

- **General**
 - Product version
 - Database version
 - Commits to the database have been suspended
 - Replication from this node has been suspended
 - Unique identifier for this server
- **Send queue to <node><port>**
 - Minimum size of the queue
 - Queue backlog
 - Maximum size of the queue (% of disk space)
 - Disk space used (%)
 - Queue has reached minimum free space threshold
 - Queue is empty
 - Time since last queue item was processed (seconds)
 - Last item completely processed
 - Queue delay of last item (seconds)

- Size of failed procedure log
- Current queue sequence number
- Generation ID of this queue
- **Receive queue from <node><port>**
 - Minimum size of the queue
 - Queue backlog
 - Maximum size of the queue (% of disk space)
 - Disk space used (%)
 - Queue has reached minimum free space threshold
 - Queue is empty
 - Time since last queue item was processed (seconds)
 - Last item completely processed
 - Queue delay of last item (seconds)
 - Size of failed procedure log (bytes)
 - Generation ID of this queue
- **Receive queue usage statistics for the last 5 minutes**
 - Command
 - Total time spent processing (ms)
 - Number of executions

Send queue and **Receive queue** will show the queues for each connected node where:

- <node> is the address of the replication node.
- <port> is the port number for the Database Service.

Configuration notes

- This database status refers to *Hitachi ID Bravura Security Fabric's* Database Service, not to the actual database status, which must be monitored separately.
- In shared schema (either hybrid or simple - only one database node with more than one application node), the local status and settings have to be configured separately on each application node that connect to the same database.
- The settings for each node can be found in the application node's file system in <instance>\service\iddb.cfg; as the previous note implies, after settings propagation these files should contain the same data in all nodes in direct replication (other than the local-node setting), but different on the nodes in shared

schema. Using the example in [Configuring replication nodes within a shared schema environment](#), iddb.cfg in A1 and B1 will contain settings from A1 and B1 only, the one in A2 and B2 will contain settings from A2 and B2 only, and so on.

4.7.4 Automated node check when using a load balancer

In addition to the manual process of checking the replication status, *Hitachi ID Bravura Security Fabric* also presents a web-based interface that can be used to determine the status of each application node.

The `loadbalancerstatus.exe` utility adds an API at `http://<domainname>/<instance>/api/nodestatus` that runs all configured tests when called and provides either an `HTTP OK 200` response if all configured tests passed, or `HTTP Service Unavailable 503` if any test failed.

The list of tests to run is configured via `nodestatus.cfg`, located at `C:\Program Files\Hitachi ID\IDM Suite\<instance>\psconfig`. Most tests are enabled by default, and can be enabled or disabled individually by marking the item as a comment.

Bravura Security Fabric also monitors the performance of Internet Information Services (IIS). When a request takes longer than a threshold, it may mean that IIS has timed out waiting for result and returned an error to the caller. If this happens a lot, that means this server is overloaded. API `nodestatus` will return 503 and this server will be taken out of the load balancer.

Timeout threshold is defined by a DWORD key in `ajaxsvc` registry: `timeout_threshold_ms`.

`ajax_max_timeouts` and `ajax_timeout_minutes` are defined in `nodestatus.cfg`. These let you configure how many timeouts in the last certain number of minutes are allowed before `nodestat` starts complaining.

Table 4.1: `nodestatus.cfg` test options

Variable	Description
\$Services	<p>A group of tests that confirm that the given service is running. Note that <code>iddb</code> and <code>ajaxsvc</code> must be running for any node status test to succeed.</p> <ul style="list-style-type: none"> • <code>iddb</code> • <code>ajaxsvc</code> • <code>idarch</code> • <code>idwfm</code> • <code>idpm</code> • <code>msgsvc</code>
Disk	Returns success if the node's replication queues have not hit the high water mark.
Ping	Calls the ping stored procedure to confirm that the database can be contacted.
db_commit_suspend	Returns success unless this node has suspended database commit operations, which typically occur during resynchronization or when a replication queue has become full.


... continued on next page

Table 4.1: nodestatus.cfg test options (Continued)

Variable	Description
Record = nodestat.db	Enables recording test results to a database flatfile, and defines the name of that file. This database is automatically written to the <i><instance>\db\nodestatus</i> directory.
ajax_max_timeouts = 10	Number of ajax timeouts to watch for
ajax_timeout_minutes = 5	How many minutes to observe ajax timeouts in
Plugin = "loadbalancerstatus.py"	Calls the plugin located at <i><instance>\plugin</i> to evaluate this node's status. Returns success if the plugin succeeds.

4.7.5 Modifying node settings

To modify a node:

1. Click **Manage the system** → **Maintenance** → **Database replication** .
2. Select  a node.
3. Select the **Configuration** tab.
4. Modify the node's settings.

You can change the:

- **Description**
- **External address**

Any URLs returned to client workstations can use a separate, externally addressable name, such as when downloading packages from session monitor or redeeming one-time use tokens. This address can only be modified for the replication node it is accessed from, otherwise the option will be disabled.

- **Mode**
- **Queue size limit** settings

If the replication is suspended on a node, then changes made on other replication nodes will not be propagated to this node.

WARNING!: Do *not* make any changes to a replication node while it is suspended.

5. Click **Update**

If the **External address** is modified, the Database Service needs to be restarted in order for the changes to take effect.

To delete a node, check the box next to the node on the **Database replication** page, then click **Delete**. You can also delete the node from its information page.

If the current node's server clock is different from that local node's clock by 60 seconds, a warning message is displayed, noting the time difference.

Next:

Propagate changes (p39)

4.7.5.1 Configuring file synchronization

Each *Hitachi ID Bravura Security Fabric* instance can synchronize the files they use; for example, scripts, connectors, and plugin programs. File synchronization is controlled by the File Replication Service (idfilerep) running on each server, and the process is activated by the **updinst** program. The **updinst** program is run during auto discovery when the **Manage the system** → **Maintenance** → **Options** → **PSUPDATE FILE REPLICATION** setting is enabled. This is the default setting.

The File Replication Service (idfilerep) archives existing files before overwriting them. By default, the archived files are stored in the Logs directory for the instance (<Program Files path>\Hitachi ID\IDM Suite\Logs\<instance>\). You can change the archive directory by using the **Manage the system** → **Maintenance** → **Options** → **FILE REPLICATION ARCHIVE DIR** setting. This directory will be automatically created on the other instances during file replication if it does not already exist.

The **Manage the system** → **Maintenance** → **Options** → **FILE REPLICATION TIMEOUT** setting is used to specify a timeout value (in seconds) before the File Replication Service disconnects. The default value is 300 seconds. This timeout only applies if servers lose their connection while backing up or deleting files; an error occurs immediately if the servers are unable to maintain a connection while replicating files.

To manually perform file synchronization:

1. Click **Manage the system** → **Maintenance** → **File synchronization**.
2. Select all file replication servers that you want to synchronize. You can choose file replication servers and proxy servers.
3. Click **Synchronize**.

If any nodes are missing from the **File synchronization** page (**Manage the system** → **Maintenance** → **File synchronization**), verify that the missing nodes have network connectivity, then restart their File Replication Services. Reload the **File synchronization** page. The missing nodes should be displayed after restarting their File Replication Services.

By default, **updinst** replicates all files and registry settings in *Bravura Security Fabric* instance. You can write an **updinst.cfg** file to provide additional configuration, including a white list and black list of files and settings to replicate.

A sample of **updinst.cfg** is located in the samples\ directory. This configuration file must be placed in the \<instance>\psconfig\ directory before it can be used by the File Replication Service. Use this configuration file to control which files and registry settings are replicated to other instances (white list) and which are not replicated (black list). The white list settings override black list settings.

WARNING!: All file and configuration modifications should be done on the same server (the primary). When attempting to run **updst** from a node other than the primary, an error will occur, and the operation will be aborted. In extreme circumstances there is an option to force external data store replication (**-extdb -forcerun**) from a secondary node; however that should be done only when that database was corrupted on the primary (and its backups that are created every time the external data store is updated, were also corrupted) but the database, or a backup, survived on a secondary node. If **updst** is run from more than one server, or if file or registry changes are made on secondary nodes, it is possible for it to overwrite newer files or settings that exist on secondary nodes. If a server with missing files runs **updst**, that will remove those same files on all other instances.

Updating database service files

CAUTION: Do *not* attempt to replace Database Service files using **updst** or the File Replication Service. Updating the Database Service and related files (such as **idbmssql.dll**) must be done manually on all instances. This only applies to the Database Service service. All other services can be updated using the File Replication Service. To update the Database Service files manually, shut down all services on the instance, back up all services, and then replace the Database Service files.

Sending an alternative address

If the server on which you are running the File Replication Service cannot access the other replication servers using the hostname (that is, database replication has to use the node's IP address to connect with other nodes), you can set the "serveraddress" string value in the instance's registry to broadcast the node's IP address to other replication nodes. This address can be used to set the file replication information.

See also:

- [idfilerep](#) for details on **idfilerep**.
- [updst](#) for details on **updst**.

4.7.5.2 Configuring replication event actions

Replication-related events can trigger external notification programs. Click **Manage the system** → **Maintenance** → **Options** to set event options, listed in [Replication events that launch interface programs](#).

Table 4.2: Replication events that launch interface programs

Option	Description
DB COMMIT RESUME	<p>Database commits have been resumed on this node after a DB COMMIT SUSPEND event. Either the queue limit has been adjusted, space has been freed on the disk, or activity on the server has slowed to allow the queue to flush.</p> <p>This event is preconfigured to run pxnull with the configuration file pxnull-replication.cfg.</p>
DB COMMIT SUSPEND	<p>A server has failed to send data to another server too many times, causing the send queue to be full. Database commits have been suspended on this node.</p> <p>This event is preconfigured to run pxnull with the configuration file pxnull-replication.cfg.</p>
DB FAILED PROC RECORDED	<p>A server has failed to insert data into the sending queue, or has failed to run procedures from the receive queue during replication.</p> <p>This event is preconfigured to run pxnull with the configuration file pxnull-replication.cfg.</p>
DB QUEUE INSERT FAILURE	<p>Data could not be inserted into the send queue or receive queue of a remote replication node.</p> <p>This event is preconfigured to run pxnull with the configuration file pxnull-replication.cfg.</p>
DB REPLICATION CONN FAILURE	<p>A local replication node lost connectivity to a remote server during a periodic check of availability. The system retries connecting to the disconnected node once every 30 seconds. This exit trap will run once every 10 minutes.</p> <p>This event is preconfigured to run pxnull with the configuration file pxnull-replication.cfg.</p>
DB REPLICATION CONN RESTORED	<p>A replication node successfully connects to a remote server that it had previously failed to contact.</p> <p>This event is preconfigured to run pxnull with the configuration file pxnull-replication.cfg.</p>
DB REPLICATION TRANS FAILURE	<p>The database service on a replication node failed to transmit data to a connected remote node. The transmission will be retried.</p> <p>This event is preconfigured to run pxnull with the configuration file pxnull-replication.cfg.</p>
DB REPLICATION QUEUE DELAY PAST THRESHOLD	<p>The queue delay of receive queues is larger than the value of DB REPLICATION QUEUE DELAY THRESHOLD.</p>
DB REPLICATION WATERMARK WARN	<p>The amount of data in a replication queue (to the local database or associated with a remote node) has passed a threshold. The threshold is set by Warn when percentage of maximum queue size used is, in terms of percent of disk or MB, on the Manage the system → Maintenance → Database replication menu.</p> <p>This event is preconfigured to run pxnull with the configuration file pxnull-replication.cfg.</p>

... continued on next page

Table 4.2: Replication events that launch interface programs (Continued)

Option	Description
FILE REPLICATION FAILURE	An error occurs during the file replication process.

4.7.5.3 Replacing the *Bravura Security Fabric* license

Hitachi ID Bravura Security Fabric license files occasionally need to be replaced. There are considerations to make when doing this in a replication environment. The license file is read by the Database Service (iddb) when it starts up.

When manually replacing the *Bravura Security Fabric* license files in a replication environment, you must:

1. Manually replace the license file in the `\<instance>\license\` directory of each node.
2. Restart the Database Service service on every node.

The downtime should be minimal. If access to the nodes is slow or otherwise inconvenient, but you have administrative access to the nodes, you *could* put the license on the primary node and synchronize files (**Maintenance** → **File synchronization**) wait for it to end (look at `idmsuite.log` until `updinst` ends), and then start the Database Service remotely from the primary on all other nodes.

Alternatively, you can use Windows' built-in `sc` command to stop and start `iddb` services, and all other services dependent on it:

```
sc \\server stop iddb_<instance_name>
sc \\server start iddb_<instance_name>
sc \\server start idpm_<instance_name>
sc \\server start idfilerep_<instance_name>
```

and so on, to start all Hitachi ID Systems services for that instance, which were stopped when Database Service stopped.

Or, use SysInternals' PsService to restart `iddb`:

```
psservice \\server restart iddb_<instance_name>
```

Using PsService is the simplest and fastest solution, although it requires placing the PsService utility on the primary node, from Windows' SysInternals. See <https://docs.microsoft.com/en-us/sysinternals/downloads/psservice>.

4.7.6 Removing a node from replication

In some cases, a replicated node cannot be recovered quickly enough; that is, before other servers' queues fill up, as described in [Time available to fix problems](#).

In these cases, the configuration of still-healthy servers should be modified to stop replication with unavailable application nodes, as described below.

After finishing any changes remember to verify the replication configuration. See [Cleaning up replication configuration](#).

4.7.6.1 Configuring replacement servers

Any settings that the removed server was responsible for must be configured to run on a surviving server, including:

- Service IDs in use by managed system policies.
- Proxy server settings on target systems.
- Reconfigure email settings. i
- If required, update links, load balancers, or network settings to direct users to log in using the new front-end server
- Server responsibilities:
 - Use the `servicemove` program to move the failed server's workload to the new server. See [servicemove](#) for more information about using this program.

4.7.6.2 Removing a database node

To remove an application node if administrators still log into the *Bravura Security Fabric* interface:

1. Log into any one of the working servers as a product administrator with the **Maintain servers** administrative privilege.
2. Click **Manage the system** → **Maintenance** → **Database replication**.
3. Select the node that is causing the problem and click **Delete**.
4. If at least one server is still available to replicate to, click **Propagate and reload replication configuration on all servers (without resynchronizing nodes)** to copy the changes to all remaining *Bravura Security Fabric* servers and restart the database.

To remove a shared schema application node, after decommissioning the node see [Cleaning up replication configuration](#) on cleaning up replication configuration.

If the shared schema node application services ever come back up and that node can connect to the database, the node will re-register itself and will show up in shared schema with the node that replaced it.

To remove an application node from a hybrid schema topology without replacing it, all application nodes that replicate to it from the other database nodes have to be removed as well. In effect, the reverse of [Adding an application node in shared schema](#) has to be applied.

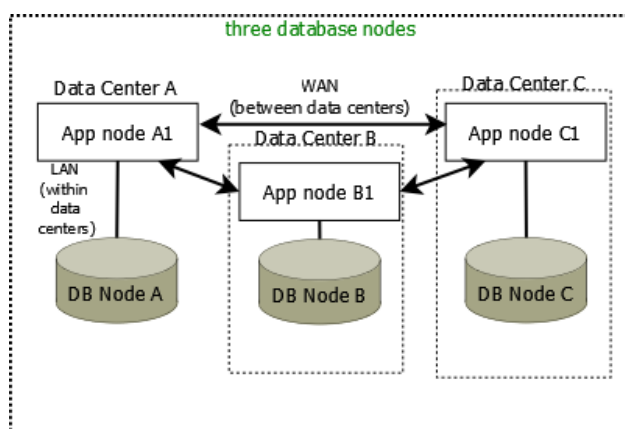


Figure 4.6: database node removed

4.7.7 Cleaning up replication configuration

Hitachi ID Bravura Security Fabric records the details of application nodes that are part of an instance, in its backend database in the "servicelist" table; each time a service starts (or stops gracefully) on a node, it updates its record in that table.

Scheduled jobs which are configured to run on specific application nodes are stored in the "schedrun" table and refer to those nodes by their "guid", an alphanumeric string like "e6bf36ad-2cb6-4347-9d8c-52154d8234e6" which can be found in the "servicelist" table mentioned above.

The configuration for the database service (**idddb**) is stored in each application node in the instance's service\idddb.cfg file.

After changing replication configuration it is recommended to verify in these three places that the change was as expected:

- "servicelist" table should contain only the nodes currently in replication.
- "This table can be emptied (on all databases involved), and then the Database Service has to be restarted on all remaining application nodes, to re-populate the table.
- "schedrun" table should contain rows for only current node guides.
- If there are rows with guides for any old nodes, they have to be deleted.
- Before deleting rows, make sure that scheduled jobs assigned only to those old nodes are reassigned to other nodes (by updating the "guid" field on those rows).
- All direct-to-DB changes like the ones above must be done separately on all databases.
- idddb.cfg files on all app nodes have to be checked to make sure the correct nodes are set to data-replicate.

Any incorrect configuration has to be corrected.

Recovering from Network Faults

5

This chapter explains what happens when data replication fails, describes the impact of each mode of failure and offers guidance regarding how to recover from each one. It shows you how to remove a server from the replication setup, and synchronize a new server with an existing setup.

To learn how to initially set up replicated servers, see [Configuring database replication](#).

5.1 Data Replication Queue

Whenever an event, such as a password change, login, or request submission, occurs on a single *Hitachi ID Bravura Security Fabric* server, information about the event is stored on the local database instance dedicated to that *Bravura Security Fabric* server, and is replicated to all other *Bravura Security Fabric* servers, each with their own physically distinct database instance.

This replication happens in real time so long as there is connectivity between the *Bravura Security Fabric* database services on the respective *Bravura Security Fabric* servers.

In the event that a given *Bravura Security Fabric* server (henceforth called *originator*) cannot contact the database service on another *Bravura Security Fabric* server (henceforth called *replica*), the update is queued on the originator's file system in queue files.

In the event that real-time replication stops and the queue starts to fill, the originator *Bravura Security Fabric* server will stop functioning when the queue is full. This is because the alternative would be to write updates to the originator's database without replicating them to other servers – in effect, creating a single point of failure in the system.

5.1.1 Queue settings

When database replication is unavailable, replication data is queued until the queue file has reached its size limit, or the node comes back online. You can modify:

- **Receive queue configuration for this node**
- **Send queue configuration for all nodes replicating to this node**

You can set the queue limit to a fixed size, using the following settings:

- **Minimum queue size** - default is 100MB

- **Maximum queue size** - for fixed size limit
- **Queue usage warning threshold (%)**: - default is 60%

You can set the queue limit to a percentage of disk, using the following parameters:

- **Minimum queue size** - default is 100MB
- **Maximum usage before queue stops growing (%)**: - default is 90%
- **Disk usage warning threshold (%)**: - default is 60%

It is best practice to set these variables the same on all *Bravura Security Fabric* servers. After making changes, you must restart the Database Service (iddb) or click **Propagate and reload replication configuration on all servers** on the replication configuration page.

During the replication process, the replicating servers attempt to send data to another server. If they fail to connect, they retry after 30 seconds until they succeed, or the queue reaches its limit. Once the limit is reached, the sending server suspends database commits, and stops accepting requests.

5.1.2 DB COMMIT SUSPEND mode

The server state when the queue is full is called *DB COMMIT SUSPEND mode*. One symptom of this mode is that every *Bravura Security Fabric* UI page shows the following message:

```
Changes to this instance are temporarily disallowed. Please contact the
Hitachi ID Bravura Security Fabric administrator. Due to a problem in
the
replication environment all pages except Database replication and System
logs are temporarily disabled.
```

When the server is in DB COMMIT SUSPEND mode, only superusers can log into *Bravura Security Fabric*. They will only have access to the systems log and database replication pages.

This state triggers the **DB COMMIT SUSPEND** exit trap. If you adjust the queue limit, or free some disk space when the limit is set as a percentage of disk, to allow the server to come back to normal replication mode. This triggers the **DB COMMIT RESUME** exit trap.

CAUTION: In the situation where the replication queue files have exceeded their limit, the logging of the superuser's actions to fix the problem may only be recorded on the local system, leading to potential security problems and unsynchronization. You may have to manually update other systems with the session logs created at the time.

5.1.3 Event actions (exit traps)

You can configure *Hitachi ID Bravura Security Fabric* to send email warnings, or some other notification, to administrators when replication failures occur.

Replication options, listed in [Replication events that launch interface programs](#), can be accessed by navigating to **Manage the system** → **Maintenance** → **System variables** or **Manage the system** → **Maintenance** → **Options**.

By default some events are configured to run the `pxnull` interface program with the `pxnull-replication` script, located in the `\<instance>\script\` directory, to notify users when the server has attempted to reconnect to the downed server a certain number of times. The script is set to use the settings set in the **Email configuration** page and contact the account in the **RECIPIENT EMAIL** field. You must install the `pxnull` program with the *Hitachi ID Connector Pack*.

You can also send a warning email to administrators in the event of a short-lived replication problem, by configuring the **DB REPLICATION CONN FAILURE** event action (**Manage the system** → **Maintenance** → **Options**). The advantage of detecting short-lived replication problems is that administrators can react more quickly to problems. The disadvantage is that there may be many spurious replication problems, due to a busy server rather than any underlying problem in the environment, and these may generate too many emails. A carefully configured exit trap can check for a threshold number of problems before sending an email. See the sample script in [Sample Replication Warning Event Configuration](#).

5.1.3.1 Sample Replication Warning Event Configuration

```

1 function REPLICATION_FAILURE
2 {
3     var $retryKey;
4     var $retryVal;
5     var $retryThreshold = 50;  #Default limit is 256 tries every minute before database
6                                 is suspended.
7
8     for( var $i = 0; $i < size( $sessdat ); $i = $i + 1 )
9     {
10         $retryKey = keyAt($sessdat, $i);
11         $retryVal = valAt($sessdat, $i);
12
13         if (strcmp($retryKey, "retry")==0)
14         {
15             if ($retryVal > $retryThreshold)
16             {
17                 # Add code to generate and send email notifications, and open
18                 # monitoring tickets here
19                 log ("replication-failure-pxnull.cfg: Send emails about failure: " +
20                     $retryVal);
21             }
22             else
23             {
24                 # Threshold has not been reached. Do nothing.
25             }
26         }
27     }
28 }

```

```

    return 0;
27 }

```

5.1.4 Time available to fix problems

The time interval between when replication stops – for example, due to network outage or hardware problem – and when the originator server becomes non-functional, depends on the rate of data updates:

Event	Size in queue	Capacity of 100MB queue
Interactive login to <i>Bravura Security Fabric</i>	5850 Bytes	17000
Checkout a single password	3228 Bytes	31000
A single password update to a target system:	6400 Bytes	15600

The time available to resolve a problem before *Bravura Security Fabric* functionality fails depends on the frequency of logins, password changes on target systems, and passwords being checked out.

Note: Hitachi ID Systems recommends that you periodically estimate the time available to resolve problems based on current metrics and the current queue size.

5.2 Modes of Failure

The following scenarios illustrate ways to recover from possible failures in a replicated environment.

- User loses connectivity
- Single *Bravura Security Fabric* server fails
- Link between single *Bravura Security Fabric* server and its database goes offline
- Single *Bravura Security Fabric* database goes offline
- Link between two *Bravura Security Fabric* servers goes offline
- Link between *Bravura Security Fabric* servers and proxy servers goes offline
- A single proxy server fails
- A link to a site where there are target systems fails
- A site where one or more *Bravura Security Fabric* servers is installed is offline
- A site where one or more *Bravura Security Fabric* proxy servers is installed is offline
- A site where one or more target systems is installed is offline

These scenarios are based on Figure 5.1, which illustrates a basic *Hitachi ID Bravura Security Fabric* configuration, including:

- Three replicated and load balanced *Bravura Security Fabric* servers.
- Access to target systems which is routed via *Bravura Security Fabric* proxy servers, which are themselves firewalled and load-balanced.

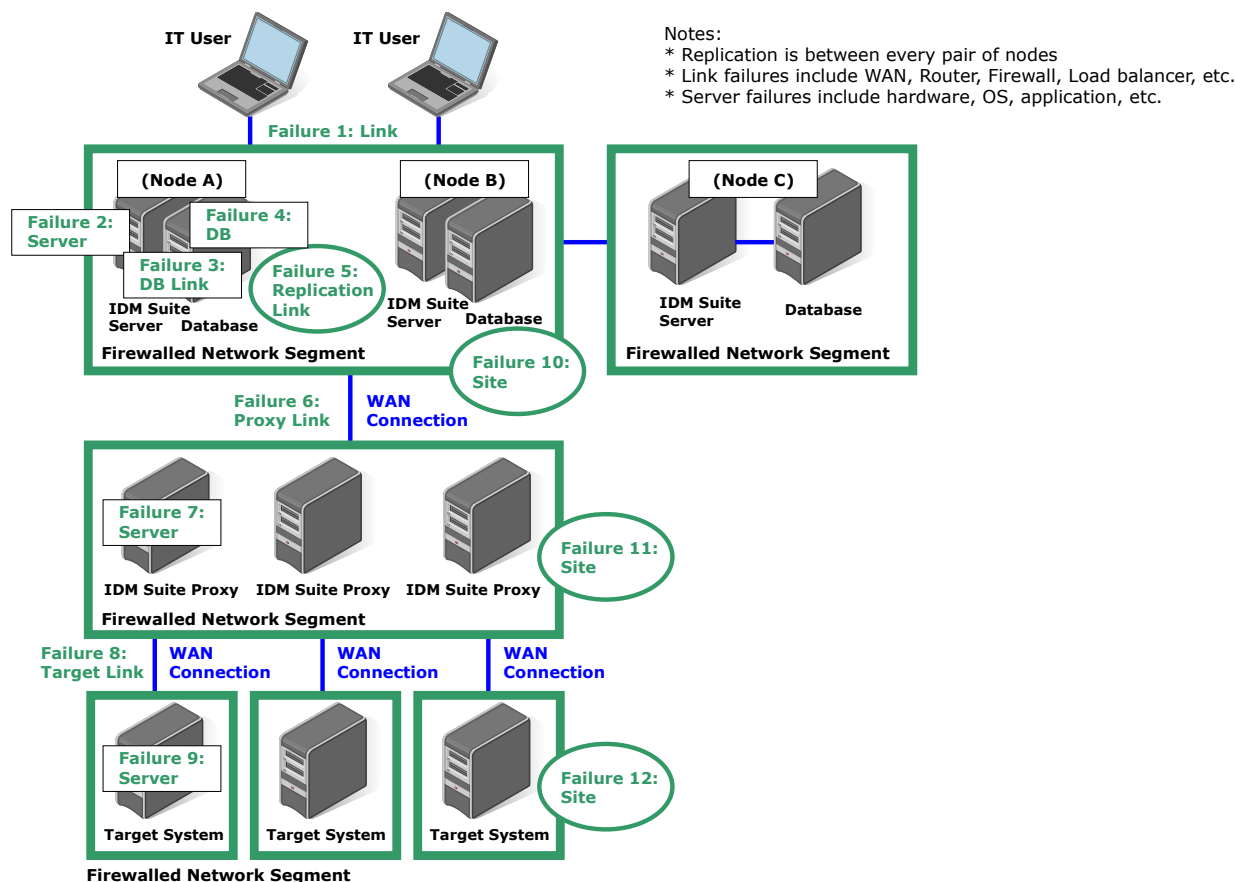
The figure illustrates the major modes of failure which can take place.

5.2.1 User loses connectivity

5.2.1.1 Scenario

The user loses connectivity which is required to access *Hitachi ID Bravura Security Fabric*. This may be for a variety of reasons, including:

- His ISP goes offline.
- The firewall protecting the *Bravura Security Fabric* servers malfunctions.
- The load-balancer distributing connections across multiple *Bravura Security Fabric* servers malfunctions.

Figure 5.1: Conceptual *Bravura Security Fabric* network architecture

5.2.1.2 What stops working

The user is unable to manage users or resources that are controlled by *Bravura Security Fabric*.

This is probably an inconsequential loss, since the user probably cannot connect to the resources either – that is, loss of connectivity to *Bravura Security Fabric* is likely accompanied by loss of connectivity to the resources which the user needs to manage.

5.2.1.3 What still works

Bravura Security Fabric continues to manage passwords on target systems and no data is lost. It is the ability of certain users to do work using *Bravura Security Fabric* which is impaired.

5.2.1.4 Data loss

None.

5.2.1.5 How to detect

Users complain that they cannot connect to *Bravura Security Fabric*.

5.2.1.6 How to recover

Resolve the connectivity problem. This is not a problem with *Bravura Security Fabric* itself.

5.2.2 Single *Bravura Security Fabric* server fails

5.2.2.1 Scenario

A problem occurs on a single *Bravura Security Fabric* server. This may be for a variety of reasons, including:

- Hardware problem, such as a disk crash.
- Operating system problem, such as a bug, or full disk.
- Application problem, such as a bug, or misconfiguration.

5.2.2.2 What stops working

- Users can no longer log into this server.
- Users can no longer retrieve passwords from this server.
- This server can no longer push password updates to target systems for which it is responsible.
- Other servers detect that replication is impossible to this server, so start queuing updates to this server and displaying alarm messages, indicating that when the queue fills, they will stop functioning normally.
- If the queue is allowed to fill – which could take several hours to several days, depending on activity level and queue size – other servers will suspend services; users will be unable to log in (since logins are logged in a replicated fashion) and will be unable to checkout passwords.

Effectively, the entire system will go into an alert state until the dead server is repaired or removed from replication. The entire system will eventually switch to a DB COMMIT SUSPEND state if a repair is not made before replication queues on the other servers fill.

5.2.2.3 What still works

Other servers continue to function normally, unless their replication queues reach their limit.

In the event that the queue is full on other servers, they switch to DB COMMIT SUSPEND mode. In that case the only possible action is to remove the non-functional server from replication.

5.2.2.4 Data loss

No data loss or – due to an unavoidable race condition – minimal data loss if updates on target systems were not yet committed to the database when the damaged server went offline

5.2.2.5 How to detect

Users may complain that they cannot log into this server at all (no login page) and that other servers display the following error message:

```
Changes to this instance are temporarily disallowed. Please contact the
Hitachi ID Bravura Security Fabric administrator. Due to a problem in
the
replication environment all pages except Database replication and System
logs are temporarily disabled.
```

The **DB COMMIT SUSPEND** event (**Manage the system** → **Maintenance** → **Options**) is configured by default to send an email as soon as the *Bravura Security Fabric* server has entered this state.

5.2.2.6 How to recover

Fix the failed server if it can be done in time (see [Time available to fix problems](#)). Other servers will continue to function in the meanwhile. See [Troubleshooting Bravura Security Fabric server failures](#) for fixes to some possible failures.

If the server cannot be fixed quickly or is permanently damaged, remove it from the replication configuration on other servers promptly, as described in [Removing a node from replication](#).

If the failed server can be recovered (for example, by installing new hardware), synchronize the node with the already-running replicated nodes, using the process described in [Synchronizing a new node with an existing set of Bravura Security Fabric replicas](#).

If the failing server was acting as the primary, then it may be necessary to promote one of the secondary nodes in order to allow it to initiate resynchronization. Update the list of scheduled jobs so the most up-to-date replica is acting as the primary, then resynchronize the new replacement node. Once the replacement has been confirmed as functional, it can be promoted to the primary node in the same manner.

5.2.3 Link between single *Bravura Security Fabric* server and its database goes offline

5.2.3.1 Scenario

A problem occurs on the network connecting a single *Bravura Security Fabric* server to its database server; this presumes that the two are not on shared hardware. This may be caused by DNS problems, router, switch or cabling problems, a failed NIC, or something else.

5.2.3.2 What stops working

The impact is identical to that described in [What stops working for Single *Bravura Security Fabric* server fails](#).

5.2.3.3 What still works

Other servers continue to function normally, unless their replication queues reach their limit.

In the event that the queue is full on other servers, they switch to DB COMMIT SUSPEND mode. In that case the only possible action is to remove the non-functional server from replication.

5.2.3.4 Data loss

No data loss or – due to an unavoidable race condition – minimal data loss if updates on target systems were not yet committed to the database when the damaged server went offline

5.2.3.5 How to detect

Users may be given warning messages that refer to the module not being licensed when users attempt to start a new login session or their existing session is terminated. This error is generated because the database service is unable to authenticate, log, or confirm requests.

The most likely errors that users will see are:

```
"Invalid session key! Please re-log in."
```

or

```
"This module is not enabled for use! Please call your help desk."
```

The logs may include messages such as the following:

```
iddb.exe [948,2296] Error: Failed to initialize the SQL Server OLE DB provider, ensure
it is installed [0x80004005]
iddb.exe [948,2296] Error: Got error [0x80004005], [2], [0x0], [0x4005]
```

```

idbdb.exe [948,2296] Error: Provider error [HRESULT:      0X80004005
SQLSTATE:      HYT00
Native Error:   0
Source:         Microsoft SQL Native Client
Error message:  Login timeout expired
HRESULT:        0X80004005
SQLSTATE:       08001
Native Error:   10061
Source:         Microsoft SQL Native Client
Error message:  An error has occurred while establishing a connection to the server.
                When connecting to SQL Server 2005, this failure may be caused by the fact that under
                the default settings SQL Server does not allow remote connections.
HRESULT:        0X80004005
SQLSTATE:       08001
Native Error:   10061
Error state:    1
Severity:       16
Source:         Microsoft SQL Native Client
Error message:  TCP Provider: No connection could be made because the target machine
                actively refused it.
]

```

Any system monitoring system that is tracking the health of the database should also alarm at this point.

5.2.3.6 How to recover

Network links and DNS problems should be diagnosed and repaired quickly (see [Time available to fix problems](#)).

If the server/database link cannot be fixed quickly, the affected *Bravura Security Fabric* server should be removed from the replication configuration on other *Bravura Security Fabric* servers promptly. Instructions for this are in [Removing a node from replication](#).

At a later date, the server should be returned to the replicating set using instructions from [Synchronizing a new node with an existing set of *Bravura Security Fabric* replicas](#).

5.2.4 Single *Bravura Security Fabric* database goes offline

5.2.4.1 Scenario

A problem occurs on the database server used as a back end for a single *Bravura Security Fabric* server. This takes the database offline and incapacitates the *Bravura Security Fabric* server in question.

5.2.4.2 What stops working

The impact is identical to that described in [What stops working for Single *Bravura Security Fabric* server fails](#).

5.2.4.3 What still works

Other servers continue to function normally, unless their replication queues reach their limit.

In the event that the queue is full on other servers, they switch to DB COMMIT SUSPEND mode. In that case the only possible action is to remove the non-functional server from replication.

5.2.4.4 Data loss

No data loss, or minimal data loss if updates on target systems were not yet committed to the database when the damaged server went offline.

5.2.4.5 How to detect

From the point of view of the *Bravura Security Fabric* server, this is indistinguishable from a failed link to the database. It is possible that system monitoring logs will discern whether the problem is connectivity to the database or the database itself.

5.2.4.6 How to recover

Database problems may be due to hardware or OS on the database server (assuming that it is separate from the *Bravura Security Fabric* server). They may be as simple as a full file system or may be more complex.

Diagnostics of database problems are outside the scope of this document. Repair the database if possible (see [Time available to fix problems](#)).

If the database link cannot be fixed in time, remove the affected *Bravura Security Fabric* server from the replication configuration on other *Bravura Security Fabric* servers promptly. Instructions for this are in [Removing a node from replication](#). At a later date, the server should be returned to the replicating set using instructions from [Synchronizing a new node with an existing set of *Bravura Security Fabric* replicas](#).

5.2.5 Link between two *Bravura Security Fabric* servers goes offline

5.2.5.1 Scenario

The link between two *Bravura Security Fabric* servers becomes non-functional. This may be due to a bad NIC, network cable, network switch, router, WAN link, or something else. The result is that the two servers cannot communicate and consequently cannot replicate updates.

5.2.5.2 What stops working

Attempted replication events, including sending a record of user logins from one server to another, will cause the sending server to detect the outage automatically. Other (still functioning) servers will start displaying a warning about replication problems and queuing updates until the unavailable server comes back on-line.

If the queue fills on replicated servers, these servers enter a **DB COMMIT SUSPEND mode** (p54). At that time, the only available option is to remove the failed server from the functional servers' replication configuration.

5.2.5.3 What still works

Each server continues to function, and queues updates to its peers until the link comes back up. Functionality is suspended if a configured retry-value has been reached, or if the queue fills.

5.2.5.4 Data loss

No data loss or – due to an unavoidable race condition – minimal data loss if updates on target systems were not yet committed to the database when the damaged server went offline

5.2.5.5 How to detect

Users logged into the individual *Bravura Security Fabric* servers may not notice this problem at all. You can send a warning email to administrators in the event of a short-lived replication problem, by configuring the **DB REPLICATION CONN FAILURE** event action (**Manage the system** → **Maintenance** → **Options**).

5.2.5.6 How to recover

Restore connectivity quickly if possible (see [Time available to fix problems](#)).

Depending on when the failure occurs while the replicating data is being sent to the other servers, there may be some discrepancies between the nodes. If possible, check that the database backend is still up, and consolidate the databases.

If the network link cannot be fixed quickly, the affected *Bravura Security Fabric* server should be removed from the replication configuration on other *Bravura Security Fabric* servers promptly. Instructions for this are in [Removing a node from replication](#). At a later date, the server should be returned to the replicating set using instructions from [Synchronizing a new node with an existing set of *Bravura Security Fabric* replicas](#).

5.2.6 Link between *Bravura Security Fabric* servers and proxy servers goes offline

5.2.6.1 Scenario

The link between at least one *Bravura Security Fabric* server and at least one proxy server becomes non-functional. This may be due to a bad NIC, network cable, network switch, router, WAN link, or something else.

5.2.6.2 What stops working

The *Bravura Security Fabric* server in question will be unable to scramble passwords on devices supported by that proxy server.

5.2.6.3 What still works

Passwords can still be checked out by users. Check-ins will obviously not trigger a successful password change until the problem is resolved.

5.2.6.4 Data loss

None, unless an extremely unlikely race condition takes place:

1. A password update is initiated on a target system.
2. The password update succeeds.
3. Before the “success” result code is returned to the *Bravura Security Fabric* server, the link is severed.
4. As a result, the *Bravura Security Fabric* server “thinks” the password change failed and records a failure, rather than the new password.

5.2.6.5 How to detect

The effect is the same as a target server being disconnected. Users may see that requested operations, such as password changes, have not happened yet. You can run system operation reports; for example to see which password have been expired and not changed recently. You can also review logs to detect failed password reset attempts in the form of connectors errors.

A recommended way to detect this condition and related problems is to configure events to trigger email or update tickets in the event of a failed operation.

5.2.6.6 How to recover

Repair connectivity as soon as possible.

5.2.7 A single proxy server fails

5.2.7.1 Scenario

A problem occurs on a single *Bravura Security Fabric* proxy server. This may be for a variety of reasons, including:

- Hardware problem, such as a disk crash.
- Operating system problem, such as a bug, or full disk.
- Application problem, such as a bug or misconfiguration.

5.2.7.2 What stops working

The main *Bravura Security Fabric* servers will be unable to scramble passwords on devices supported by that proxy server. The *Bravura Security Fabric* server is unable to connect to any device that requires that proxy server. *Bravura Security Fabric* will be unable to carry out operations involving a target system, including listing information, or updating data such as passwords on a machine.

5.2.7.3 What still works

Users can still log into *Bravura Security Fabric*, assuming they are not authenticating against a target through the damaged proxy. They can access their profiles and carry out tasks that do not require a connection to affected target systems.

5.2.7.4 Data loss

None, unless an extremely unlikely race condition takes place:

1. A password update is initiated on a target system.
2. The password update succeeds.
3. Before the “success” result code is returned to the *Bravura Security Fabric* server, the proxy server in question fails.
4. As a result, the *Bravura Security Fabric* server “thinks” the password change failed and records a failure, rather than the new password.

5.2.7.5 How to detect

Similar to Link between *Bravura Security Fabric* servers and proxy servers goes offline.

The effect is the same as a target server being disconnected. Users may see that requested operations, such as password changes, have not happened yet. You can run system operation reports; for example to see which password have been expired and not changed recently. You can also review logs to detect failed password reset attempts in the form of connectors errors.

A recommended way to detect this condition and related problems is to configure events to trigger email or update tickets in the event of a failed operation.

5.2.7.6 How to recover

Repair/replace the failed proxy server as soon as possible. *Bravura Security Fabric* proxy servers are essentially stateless so a fresh installation on a new physical server or virtual machine is all that is required.

5.2.8 A link to a site where there are target systems fails

5.2.8.1 Scenario

A site where there are target systems becomes inaccessible to either the main *Bravura Security Fabric* server or a proxy server which would normally be responsible for making updates to systems at that site.

This may be caused by DNS problems, router, switch or cabling problems, a failed NIC, or something else.

5.2.8.2 What stops working

The main *Bravura Security Fabric* servers will be unable to carry out operations on devices at that site.

5.2.8.3 What still works

Users can still log into *Bravura Security Fabric*, assuming they are not authenticating against a target on the offline site. They can access their profiles and carry out tasks that do not require a connection to affected target systems.

Moreover, it is likely that users who access the site via *Bravura Security Fabric* cannot connect to that site themselves, but this is beyond the scope of *Bravura Security Fabric's* functionality or responsibility.

5.2.8.4 Data loss

None, unless an extremely unlikely race condition takes place:

1. A password update is initiated on a target system.
2. The password update succeeds.

3. Before the “success” result code is returned to the *Bravura Security Fabric* server, the site in question goes offline.
4. As a result, the *Bravura Security Fabric* server “thinks” the password change failed and records a failure, rather than the new password.

5.2.8.5 How to detect

The symptoms here are identical to those in [Link between *Bravura Security Fabric* servers and proxy servers goes offline](#). A server monitoring system may be able to localize the connectivity problem.

5.2.8.6 How to recover

Repair the failed network link as soon as possible.

5.2.9 A single target system goes offline

5.2.9.1 Scenario

A system where *Bravura Security Fabric* performs operations goes offline. This may be for a variety of reasons, including:

1. Hardware problem, such as a disk crash.
2. Operating system problem, such as a bug, or full disk full.
3. Application problem, such as a bug, or misconfiguration.

5.2.9.2 What stops working

The main *Bravura Security Fabric* servers will be unable to scramble passwords on the device which is offline.

5.2.9.3 What still works

Users can still log into *Bravura Security Fabric*, assuming they are not authenticating against the offline target through. They can access their profiles and carry out tasks that do not require a connection to affected target systems.

Moreover, users who access the system via *Bravura Security Fabric* will be unable to connect to the system itself, or at least not over the network. It is possible that a checked out password will be used to assist in system recovery at the failed device’s console, however.

5.2.9.4 Data loss

None, unless an extremely unlikely race condition takes place:

1. A password update is initiated on a target system.
2. The password update succeeds.
3. Before the “success” result code is returned to the *Bravura Security Fabric* server, the server in question goes offline.
4. As a result, the *Bravura Security Fabric* server “thinks” the password change failed and records a failure, rather than the new password.

5.2.9.5 How to detect

The symptoms here are identical to those in [Link between *Bravura Security Fabric* servers and proxy servers goes offline](#). A server monitoring system may be able to localize the problem and identify a failed server as opposed to a failed network link.

5.2.9.6 How to recover

Repair the failed device as soon as possible.

5.2.10 A site where one or more *Bravura Security Fabric* servers is installed is offline

5.2.10.1 Scenario

A complete data center goes offline. This may be due to power outage, fire, flood, earthquake, hurricane, tornado, or something else. The site may be permanently damaged or simply taken offline for a while; for example, digging equipment cuts all network links to the site.

5.2.10.2 What stops working

If the site is destroyed, assuming that there are *Bravura Security Fabric* servers running in at least one other physical site, this is equivalent to [What stops working in Single *Bravura Security Fabric* server fails](#).

If the site is just knocked offline (for example, all connectivity temporarily lost), this is equivalent to a combination of [User loses connectivity](#), [Link between two *Bravura Security Fabric* servers goes offline](#) or [Link between *Bravura Security Fabric* servers and proxy servers goes offline](#).

5.2.10.3 What still works

Users at other sites can still access data from *Bravura Security Fabric* servers at other sites, as described in [DB COMMIT SUSPEND mode](#).

Operations on some target systems will continue, from servers at other sites to target systems for which those servers are responsible.

Users at an off-line site can still access local systems at their local site. They can even access data for systems at other sites, but obviously cannot connect to those other sites.

5.2.10.4 Data loss

None, unless an extremely unlikely race condition takes place:

1. A password update is initiated by a *Bravura Security Fabric* server at the soon-to-be-destroyed site.
2. The password update succeeds.
3. Before the “success” result code is returned to the *Bravura Security Fabric* server and replicated to other *Bravura Security Fabric* servers at other sites, the site hosting the *Bravura Security Fabric* server in question is destroyed.
4. As a result, *Bravura Security Fabric* servers at other sites never received the new password.

5.2.10.5 How to detect

A destroyed or offline site should be immediately detected by network monitoring systems in general. Detecting disasters is not a function of *Bravura Security Fabric* (although symptoms will be evident).

5.2.10.6 How to recover

If the problem is site-wide connectivity and it can be resolved quickly (see [Time available to fix problems](#)), wait for the site to come back on-line.

If the site was physically destroyed, see [Removing a node from replication](#) for instructions on removing affected servers from replication and ultimately building and initializing replacement servers.

5.2.11 A site where one or more *Bravura Security Fabric* proxy servers is installed is offline

A site is either destroyed or taken offline. The site houses at least one *Bravura Security Fabric* proxy server. Site disasters should be detected via other means, before *Bravura Security Fabric* is involved.

The symptoms and recovery steps in *Bravura Security Fabric* are identical to the situation described in [A single proxy server fails](#).

5.2.12 A site where one or more target systems is installed is offline

This scenario is identical to the situation described in [A link to a site where there are target systems fails](#).

5.3 Troubleshooting *Bravura Security Fabric* server failures

This section describes ways to diagnose and recover from faults that may occur on a *Hitachi ID Bravura Security Fabric* server.

5.3.1 Replication errors in log files

The following log file messages appearing in the `idmsuite.log` file indicate replication problems:

Log message	Explanation
Failed to insert data block into receive queue ...	Likely means that the local queue – which holds transactions that will be committed to the local database – is either corrupt or full.
Failed to deserialize record for ...	Something in the receive queue could not be parsed. Likely due to disk corruption, queue file corruption or a truncated transmission inbound to the local node.
Failed to execute replication ...	Received a transaction from a remote node which should be committed to the local database. Failed to execute the relevant stored procedure on the local database. See the remainder of the message for details about which procedure was called and with what arguments. Please refer to the failed procedures log for the data that could not be posted locally.
Retried proc ... times. Aborting.	Unknown (database-related) errors encountered while running stored procedure on the local database. Review contents of the failed procedures log to see what procedure was attempted.
Failed to initialize the send queue for node.	Something is wrong with setup of a new queue file which will hold transactions pending transmission to a remote node. The queue file may be corrupted or the disk full.
Stopping replication to ..., as it is sharing schema with this site.	The local database has a unique schema ID. The remote database also has a unique schema ID. Replication has been configured between the two nodes, but the replication services on the two endpoints of communication noticed that the two schema IDs are the same. This implies that the two databases are one and the same (a shared schema) so replication should not be performed.

5.3.2 Databases contain unsynchronized data

Generally, during replication:

1. A stored procedure runs on server A.
2. Server A puts that stored procedure into its queue.
3. When it is able to talk to server B, it will attempt to transmit the contents of the queue to server B.
4. If server B verifies that the transmission occurred correctly, server B inserts the data into its receive queue.
5. Server B then responds to server A stating that it received the data correctly.
6. Server A then removes the data from its queue.
7. Server B in the mean time starts processing the data in its receive queue.

Problems can occur when:

- The queue is deleted or corrupted.
- A server is taken out of replication for a while and as a result, updates to it are not queued.
- Something goes wrong and the stored procedure is not executed properly on the secondary server. In this case the problem may fix itself the next time the data is updated.

The `\<instance>\db\` directory contains two files that can be used to investigate why certain nodes do not have the same set of data as other nodes.

The file `iddb-failed-procs-receivequeue.log` keeps track of all failed replication procedures that were successfully received from the sending node, but were not processed on the local instance. These include (but not limited to) procedure errors such as:

- Invalid key constraints
- Accessing none existent records
- Full databases

The files `iddb-failed-procs-<node ID>.log` are used to track replication procedures that failed on the sender side, mainly because the procedures were not added to the queue to be sent to the receiving server. These kinds of errors include (but not limited to) :

- Corrupt queue files
- Corrupt header files

You can use these files to identify the procedure causing a problem, then determine how to correct it, either by adding/correcting the failed record, or by re-playing the stored procedure. The appropriate values needed for the failed stored procedure can be found in the **iddb-failed-procs-*** files.

The following is an example of the contents of these files:

```
2011-07-06\  
  22:21:52,\  
  PslStoreAdd,\  
  "namespace" "key" "S" "value" "4e14e000",\  
  constraint violation
```

The format of this file is:

1. Date
2. Time
3. Procedure name
4. Arguments to the procedure
5. Database error message

These logs are not automatically scrubbed, so if the database experiences many problems, they could grow without bound.

5.3.3 Missing iddb queue files

If one or more queue files are not properly generated, then a failure may occur once the queue is full and begins searching for the missing files. Files may be missing as a result of being deleted, or if the hard drive is full and new files cannot be written.

The missing file failure causes the failed-retry value to begin counting as it attempts to find the missing queue files. Once the counter reaches the failed-retry value it triggers the DB COMMIT SUSPEND state.

To recover from this suspension, restart the **iddb** service. This regenerates all missing queue files.

CAUTION: Any data that was waiting in the queue will not be replicated. A Database Administrator must manually consolidate each node's database.

5.3.4 Missing psupdate replication batch files

If one or more **psupdate** replication batch files are not properly generated during the **psupdate** process, then a replication failure can occur. Files may be missing as a result of being deleted, or if the hard drive is full and new files cannot be written.

The missing file failure causes replication to fail and possibly suspend the database.

To recover from this suspension:

1. Stop the **iddb** service.
2. Delete all replication batch files, located in:

```
C:\Program Files\Hitachi ID\IDM Suite\<instance>\db\replication\
```

Replication batch files follow this naming scheme: **<guid>_<node ID>**

Do not delete files that do not follow this naming scheme.

Note: The replication folder is only created when replication files are created.

CAUTION: Any data that was waiting in the queue will not be replicated. A Database Administrator must manually consolidate each node's database.

3. Start the **iddb** service.
4. Re-run **psupdate**. You may need to re-list users on targets.

5.3.5 Unreadable passwords

When you try to view the password for a managed system and the password does not appear as plain text, it is possible that replicated servers are out of synch. You can check and resynchronize passwords by navigating to **Manage the system** → **Privileged access** → **Managed system policies**.

5.3.6 Server clocks unsynchronized

Server clocks on replication nodes must be synchronized. Clock times should not differ by more than 60 seconds. Use the **Database replication** page to view the status of the replication nodes (**Manage the system** → **Maintenance** → **Database replication**).

If the server clocks of the replication nodes start to drift out of synchronization by too many seconds, the time status on the **Database replication** page displays this issue. You can then view the status of each node, note the time difference, and see if the server can be contacted.

You must update the time settings manually to resynchronize server clocks.

5.4 Synchronizing a new node with an existing set of *Bravura Security Fabric* replicas

You may need to add a new replica to an already-running set of replicated *Hitachi ID Bravura Security Fabric* servers; for example, if a node failed because of a disk crash, and a new server had to be built to replace it.

To add a new node to an existing replication environment includes:

1. Prepare the node by installing the operating system, database and *Bravura Security Fabric* software on a new server.

See [Preparation](#) for details.

For nodes with a Microsoft SQL Server database, see <https://docs.hitachi-id.net/#/home/2209/10/11> to learn how to create the appropriate database instance and user. See [Installing Bravura Security Fabric using idmsetup.inf](#) to learn how to install *Bravura Security Fabric* using `idmsetup.inf` file to ensure that settings are identical.

2. Add the replication node according to [Adding a node details](#).
3. Propagate replication changes according to [Propagating replication changes](#).
4. Synchronize files and registry settings from the *primary* server.

You can do this by navigating to **Manage the system** → **Maintenance** → **File synchronization**, or running `updinstr` from the command line.

See [Configuring file synchronization](#) for details.

5. Configure the new server and test replication, as described on page [77](#).

5.4.1 Resynchronizing databases

You can resynchronize the primary node's database to any other node if they become unsynchronized.

To determine if (or how much) the databases are out of sync, you can use an SQL query (ran at the same time on all databases), to count the rows of each table in the database. The following is a suggested query (run under an account with the sysadmin role):

```
SELECT OBJECT_NAME(id) AS [TableName], rowcnt AS [RowCount]
FROM sysindexes
WHERE indid IN (1,0) AND OBJECT_NAME(id) not like '%_stg' AND OBJECT_NAME(id) not like
'%_cache'
AND OBJECTPROPERTY(id, 'IsUserTable') = 1 order by rowcnt desc;
```

The primary node will have data in staging tables (*_stg) that the secondaries never receive. Except for those tables, row counts should match (with a margin of 5-10 rows; current operations on one server may not have replicated yet, or the query on one database ran slightly earlier than on another database). If counts don't match, send the row counts to support@Hitachi-ID.com to determine if the differences are important enough to require a resynchronization.

Preparation

Before resynchronizing, take the following steps:

1. Check the size of the database files (<instance>.mdf) on the primary (sending) backend database; make sure that both the sending application node and the receiving nodes have at least *twice* that much space. In Microsoft SQL Studio you can check file size by expanding **Databases**, right-clicking the database, then selecting **Properties** → **Files**.

The space available at the location of the temp files (.ldf) on the backend database since the resynchronization process will send most of the large tables over.

Hitachi ID Bravura Security Fabric saves the database to disk on the sending side and loads it from disk on the receiving side where it has to be loaded in the temp/transaction file (.ldf) then flushed to the main db file (.mdf). When the database has been saved, there has to be enough space left for the database engine and the Database Service (iddb) to function properly. The Database Service requires by default to have at least 10% free space to be able to replicate. When both the backend database and the *Bravura Security Fabric* node are storing their files on the same disk/partition/share, the free space on that disk has to be at least:

- On the sending node, 10% of the disk size plus twice the size of the sending database's .mdf files
 - On the receiving node(s), 10% of the disk size plus three times the size of the .mdf files.
2. Identify the application nodes that will have to go down and exclude them from any load balancer that end users connect through, so that started sessions are not interrupted by the Database Service going down. When the Database Service is down on a node, anyone attempting to login to that node's Web UI will get a "product not available" error.
 3. If your replication architecture is hybrid (more than one application node using the same database), stop the Database Service on any nodes that share the source and destination's databases and also remove those nodes from the load balancer.

Stopping the Database Service will stop all product services dependent on it. After that happens, start only the database service, so it can perform the resynchronization operation.

Note:

- Reduce logged errors by stopping the IIS service and the Health check task in Windows Task Scheduler on all the nodes that should not contact their databases.
- Remember to start up all *Bravura Security Fabric* services, IIS and Health check after the resynchronization process ends successfully.

Resynchronizing

Once you have completed the preparatory steps listed above, start the resynchronization:

1. Log onto the Web UI of your primary node (or the node that contains the most data) as a product administrator.

To identify the current primary node, go to **Manage the system** → **Maintenance** → **Scheduled jobs** → **PSUPDATE**, and see the node in the **Run task on this node** drop-down list. Disable the PSUP-

DATE job until you verify the resynchronization succeeded. Login to the primary application node as the server administrator and disable the Windows Task Scheduler's External database replicator task.

Also disable the Health check task if it was enabled so it does not generate logged errors while the Database Service is down.

2. Verify in the browser address field (or in the middle of the web-based interface footer) that you are logged into the node that contains the data you want to send to the other nodes.

Note: If you log into the web-based interface of a new node that you are just adding to replication and click **Resynchronize**, the data from the old nodes will be returned to defaults, as the initial data from the new node will be copied over to all other nodes selected for resynchronization, overwriting any data there.

3. Click **Manage the system** → **Maintenance** → **Database replication**.
4. Select the destination node or nodes that need to be resynchronized on the **Database replication** page.
5. Click **Resynchronize**.

This creates a database export of the source node database and sends it to the destination node. The source node will be temporarily offline during the export process. This button is only enabled on the primary node.

6. Wait while the destination node imports the data.

Depending on many variables, primarily the size of the dataset being resynchronized and the network speed, the import operation may take several hours.

The destination node will be offline during the import process.

7. In idmsuite.log on the sending node, check that all tables that start to be written to disk are successfully written. On the receiving node, check that all tables that are received are also loaded to the backend database (the largest tables will finish loading last). If there is anything preventing some the tables from loading, the resynchronization process failed; depending on the table that failed to load, contact support@Hitachi-ID.com immediately.
8. Decide which node will remain the primary node, and enable the PSUPDATE task on that node (see step 1). Enable the Windows Task Scheduler's External database replicator on the primary node and make sure it is disabled on all secondary nodes.

If you have configured the Health check task and are normally acting on its warning emails, re-enable the Health check task on all application nodes' Windows Task Scheduler.

WARNING!: When resynchronization is initiated, the data on the destination node is dropped and replaced with the data from the primary node.

5.4.2 Configuring the new application node and testing replication

After you have added the application node to the replication configuration, propagated changes, and synchronized files, complete the configuration and testing:

1. Clean up replication configuration as outlined in [Cleaning up replication configuration](#).
2. Log into the new *Bravura Security Fabric* server's administration console and set all server-specific variables, including:
 - (a) Base URL, if a load balancer is not used (**Manage the system** → **Workflow** → **Email configuration**).
 - (b) Scheduled tasks, if required.

Note: Ensure that the **psupdate** task is scheduled to run on only one node in each *Bravura Security Fabric* instance.

- (c) Registry entries.

In general, registry entries are copied when you run **updst**. Verify that any special or custom settings are correct.

CAUTION: Do not copy the database access registry keys, which are located in the *<instance>\MSSQL* directories. The credentials set here must be able to access the schema that was designated for this new instance.

3. Test that the backup/restore has completed successfully:
 - (a) Log into two different *Bravura Security Fabric* application nodes as the same product administrator.
 - (b) Ensure that the data on all configuration pages is consistent across all replication nodes.
 - (c) Make some changes from one node that propagates data to other replication nodes.
 - (d) Compare the data on all replication nodes again and ensure all changes have been propagated between them.

You can now resume normal operation.

5.5 Recommended maintenance plan

Analytics (SSRS) report synchronization

6

Support for integrating Microsoft's SQL Server Reporting Services (SSRS) was added in *Hitachi ID Bravura Security Fabric* 12.0.0. *Analytics* is an optional Hitachi ID Systems report feature that organizes and displays SSRS reports.

For more details about the analytics feature see the Reports User Guide ([reports.pdf](#)).

Application nodes may share an SSRS report server, or connect to their own SSRS report server. In either case, each node has its own report folder on the report server. A report file that exists on the report server can be viewed on that node's *Analytics* app. If users need to access the same reports from each replication node, a copy of the report file must also exist on the *Bravura Security Fabric* application server.

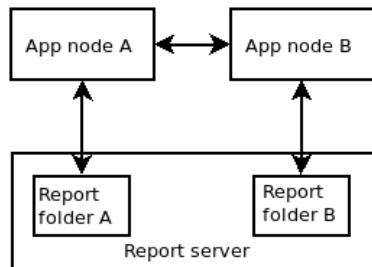


Figure 6.1: Application servers sharing same SSRS report server

Report files that exist on the SSRS report server can be downloaded to the application node in the `<instance>\analytics\ReportItems\` directory in order to synchronize them between application nodes.

The following use case demonstrates how to synchronize a report between application nodes:

1. Application node A connects to SSRS report server A, and application node B connects to SSRS report server B.
2. An administrator creates a custom report on an SSRS server that is accessed via the *Analytics* app on Bravura Security Fabric application node A.
3. The administrator uses the `configureanalytics` utility to copy report files from the report server A to the application node A, using the command:

```
configureanalytics.exe -publish
```

```

c:\Program Files\Hitachi ID\IDM Suite\default16_24028\util>configureanalytics.exe -publish
=====
Choose a file to action on:
  1. ReportItems\ANALYTICS_CATEGORY_AUDIT\Dog_sql2014 | Report | New on file system
  2. ReportItems\ANALYTICS_CATEGORY_AUDIT\horse_sql2014 | Report | New on report server
  3. ReportItems\ANALYTICS_CATEGORY_AUDIT\sql2012_sunset | Report | New on file system
---- <q>: Quit <Enter>: Make selection ----
Your choice =>

```

4. The administrator types the number corresponding to the report that is "New on report server" and presses **[Enter]**.

The report file is downloaded to the \<instance>\analytics\ReportItems\ on application node A.

5. The administrator synchronizes the files with Bravura Security Fabric application node B using the administrative interface: **Manage the system** → **Maintenance** → **File synchronization**.

During file synchronization, the report files are copied from application node A to application node B, then automatically uploaded to report server B.

The same report can now be accessed from the *Analytics* app on application node B.

Part II

TOOLS

Description

The **updinst** program synchronizes files and registry settings between servers in a multiple-instance environment, or a replication environment. The program is run during auto discovery when the **Maintenance** → **Options** → **PSUPDATE FILE REPLICATION** setting is enabled. This is the default setting.

This program also collects and synchronizes proxy log files onto the instance server.

This program is used in conjunction with the File Replication Service (idfilerep). See [idfilerep](#) for more information about this service.

By default, **updinst** replicates all files and registry settings in *Bravura Security Fabric* instance. You can write an **updinst.cfg** file to provide additional configuration, including a white list and black list of files and settings to replicate.

A sample of **updinst.cfg** is located in the `samples\` directory. This configuration file must be placed in the `\<instance>\psconfig\` directory before it can be used by the File Replication Service. Use this configuration file to control which files and registry settings are replicated to other instances (white list) and which are not replicated (black list). The white list settings override black list settings.

WARNING!: All file and configuration modifications should be done on the same server (the primary). When attempting to run **updinst** from a node other than the primary, an error will occur, and the operation will be aborted. In extreme circumstances there is an option to force external data store replication (`-extddb -forcerun`) from a secondary node; however that should be done only when that database was corrupted on the primary (and its backups that are created every time the external data store is updated, were also corrupted) but the database, or a backup, survived on a secondary node. If **updinst** is run from more than one server, or if file or registry changes are made on secondary nodes, it is possible for it to overwrite newer files or settings that exist on secondary nodes. If a server with missing files runs **updinst**, that will remove those same files on all other instances.

CAUTION: Do *not* attempt to replace Database Service files using **updinst** or the File Replication Service. Updating the Database Service and related files (such as `iddbmssql.dll`) must be done manually on all instances. This only applies to the Database Service service. All other services can be updated using the File Replication Service. To update the Database Service files manually, shut down all services on the instance, back up all services, and then replace the Database Service files.

If a problem occurs during file replication then a notification email is sent to the administrator, and the FILE REPLICATION FAILURE event is triggered.

Usage

```
updst.exe [-list] | [-showconfig] | [-synchfile] [-syncreg]
[-globalcp]
→ [-serverid <serverid>...] [-dry-run] |
[-extdb[-forcerun]] | [-getlogs] [-proxyfile <proxylist.csv>] [-removelogs]
```

Table 7.1: updst arguments

Argument	Description
-dry-run	Only show what would be done without making changes.
-getlogs	Retrieve and synchronize all logs from all proxies configured onto the instance server. Logs will be under their respective \Logs\proxy_<proxy_IP> directory.
-globalcp	Synchronize the global connector pack in addition to the current instance. Used in conjunction with the -synchfile argument.
-list	List file replication services involved and exit. Display the status of each service, node ID, address, and port. If the service is unreachable, check the connection of the Database Service for each server.
-proxyfile <proxylist.csv>	Retrieve and synchronize all logs from proxies listed in the \<instance>\psconfig\proxylist.csv file. Logs will be under their respective \Logs\proxy_<proxy_IP> directory. Where proxylist.csv contains a list of proxy IP addresses and their ports. This is only used in conjunction with -getlogs.
-serverid <serverid> ...	Synchronize to specified servers; can be used with multiple server IDs. The server ID can be found using the -list argument. If no server is specified, all servers are updated.
-showconfig	Show built-in configuration and exit.
-synchfile	Synchronize files to other servers.
-syncreg	Synchronize registry settings to other servers.
-extdb	Synchronize the extdb tables.
-forcerun	Forces the current node to synchronize the extdb tables.
-removelogs	Remove all previously fetched logs from all proxies. Can be used in conjunction with -getlogs to remove logs that no longer exist on the remote server.
-filedir <filedir>	Specific file path from which files are synchronized to other servers. This is the relative path under the instance's main file folder.
-registrydir <registrydir>	Specific registry path from which settings are synchronized to other servers. This is the relative path under the instance's main registry folder.

Examples

1. To show the default configuration file:

```
updst -showconfig
```

2. To show a list of all file replication services that are running:

```
updst -list
```

3. To synchronize all files and registry settings on all servers:

```
updst -syncfile -syncreg
```

4. To synchronize a specific file directory on all servers:

```
updst -syncfile -filedir <filedir>
```

5. To synchronize multiple file directories on all servers:

```
updst -syncfile -filedir <filedir1> -filedir <filedir2> ...
```

6. To synchronize a specific registry setting on all servers:

```
updst -syncreg -registrydir <registrydir>
```

7. To run a dry-run on a specific server:

```
updst -serverid <server id (guid)> -syncfile -syncreg
```

8. To update global *Connector Pack* files:

```
updst.exe -globalcp -syncfile
```

9. To collect and synchronize logs from all proxies:

```
updst.exe -getlogs
```

10. To collect and synchronize logs from all proxies listed in the proxylist file:

```
updst.exe -getlogs -proxyfile <proxylist.csv>
```

Where the proxylist.csv contains:

```
::proxy, port::
10.0.39.111, 3344
10.0.39.121, 3344
```

11. To synchronize the **extdb** tables:

```
updst.exe -extdb
```

12. To remove all previously fetched logs:

```
updst.exe -removelogs
```

13. To remove logs that no longer exist on the remote server:

```
updst.exe -removelogs -getlogs
```

7.0.1 Use Case: updinst config file

Most of these changes are system variables listed in the registry section at **Manage the system** → **Maintenance** → **System variables**.

In the updinst config file:

- All file paths are rooted not at the instance directory, but at the backslash immediately before it: (instead of script\somefile.py it has to be:

```
^\\\\script\\\\somefile.py.
```

- Here is an example, that adds three blacklisted registry entries (meaning that they will not be sent from the primary to secondary application nodes):

This example shows that blacklisting these entries ensures that the SENDER_EMAIL and ServerAddress are not propagated from the primary to the secondaries. These two entries will be unique to each server.

The IDAPISOAP entry is an example of an entry that already exists and is hard coded in the configuration file.

```
# KVGROUP-V2.0
"updinst" "cfg" = {
}

# Registry path blacklist. Registry paths that match the regular
# expressions here are not considered for replication, unless they
# appear in regWhitelist. All paths are relative to the instance
# registry root. Registry keys (and value names) are separated by
# double-escaped backslashes (\\\\).
"regBlacklist" = { "^IDAPISOAP\\\\endpoints",
                  "SENDER_EMAIL",
                  "ServerAddress" };
```

Description

The File Replication Service (idfilerep) receives data from a master instance in a replication environment, and is used in conjunction with the **updinst** utility to synchronize files and registry keys between multiple instances.

When **updinst** is used from the master instance, the File Replication Service (idfilerep) adds, modifies, and removes files and registry settings on the server. Configuration options for file replication are explained further in this section.

CAUTION: Do *not* attempt to replace Database Service files using **updinst** or the File Replication Service. Updating the Database Service and related files (such as **iddbmssql.dll**) must be done manually on all instances. This only applies to the Database Service service. All other services can be updated using the File Replication Service. To update the Database Service files manually, shut down all services on the instance, back up all services, and then replace the Database Service files.

Requirements

You must set up a database replication environment in order for the File Replication Service to identify replication servers with which to synchronize files. See Replication and Recovery (**replication.pdf**).

The File Replication Service uses the **updinst** utility to initiate the file replication process. See **updinst** for more information about this utility.

Configuration

The File Replication Service is automatically installed and started on the *Hitachi ID Bravura Security Fabric* server during setup. You can also modify the following parameters related to this service on the **Service information** page:

Table 8.1: idfilerep service options

Option	Description
Port number this service is running on	Specifies the port or the shared memory ID to listen on. The default is 2380.

The File Replication Service archives existing files before overwriting them. By default, the archived files are stored in the Logs directory for the instance (<Program Files path>\Hitachi ID\IDM Suite\Logs\→<instance>\). You can change the archive directory by using the **Manage the system** → **Maintenance** → **Options** → **FILE REPLICATION ARCHIVE DIR** setting. This directory will be automatically created on the other instances during file replication if it does not already exist.

The **Manage the system** → **Maintenance** → **Options** → **FILE REPLICATION TIMEOUT** setting is used to specify a timeout value (in seconds) before the File Replication Service disconnects. The default value is 300 seconds. This timeout only applies if servers lose their connection while backing up or deleting files; an error occurs immediately if the servers are unable to maintain a connection while replicating files.

To manually perform file synchronization:

1. Click **Manage the system** → **Maintenance** → **File synchronization**.
2. Select all file replication servers that you want to synchronize. You can choose file replication servers and proxy servers.
3. Click **Synchronize**.

If any nodes are missing from the **File synchronization** page (**Manage the system** → **Maintenance** → **File synchronization**), verify that the missing nodes have network connectivity, then restart their File Replication Services. Reload the **File synchronization** page. The missing nodes should be displayed after restarting their File Replication Services.

If the server on which you are running the File Replication Service cannot access the other replication servers using the hostname (that is, database replication has to use the node's IP address to connect with other nodes), you can set the "serveraddress" string value in the instance's registry to broadcast the node's IP address to other replication nodes. This address can be used to set the file replication information.

You can control whether or not to archive existing files by adding the following registry entry in:

HKLM\SOFTWARE\Hitachi ID\IDM Suite\<instance>\IDFileRep

Entry name	backups
Value	0 1 Set to 0 to disable backups
Data type	DWORD
Default	1

The File Replication Service is used in conjunction with **updstinst**. By default, **updstinst** replicates all files and registry settings in *Bravura Security Fabric* instance. You can write an **updstinst.cfg** file to provide additional configuration, including a white list and black list of files and settings to replicate.

A sample of **updstinst.cfg** is located in the `samples\` directory. This configuration file must be placed in the `\<instance>\psconfig\` directory before it can be used by the File Replication Service. Use this configuration file to control which files and registry settings are replicated to other instances (white list) and which are not replicated (black list). The white list settings override black list settings.

WARNING!: All file and configuration modifications should be done on the same server (the primary). When attempting to run **updstinst** from a node other than the primary, an error will occur, and the operation will be aborted. In extreme circumstances there is an option to force external data store replication (`-extddb -forcerun`) from a secondary node; however that should be done only when that database was corrupted on the primary (and its backups that are created every time the external data store is updated, were also corrupted) but the database, or a backup, survived on a secondary node. If **updstinst** is run from more than one server, or if file or registry changes are made on secondary nodes, it is possible for it to overwrite newer files or settings that exist on secondary nodes. If a server with missing files runs **updstinst**, that will remove those same files on all other instances.

If a problem occurs during file replication then a notification email is sent to the administrator, and the FILE REPLICATION FAILURE event is triggered.

See also:

- [updstinst](#) for information about **updstinst** usage.

Description

The **updproxy** program, installed on the *Hitachi ID Bravura Security Fabric* server, works in conjunction with the **psproxy** service. It is used to update the list of proxy servers registered on the *Bravura Security Fabric* server, and to push any files necessary to run connectors and list utilities to the proxy servers. This includes the Logging Service (idmlogsvc) being used on the proxy servers. Normally, **updproxy** is invoked as one of the first steps in the auto discovery batch file.

The **UPDATE SYSTEM AGENT COUNT** option controls how many proxy servers can be updated simultaneously. The default value is 50. This option can be set at **Manage the system** → **Maintenance** → **Options**.

Usage

```
updproxy.exe -list | -refresh | -rotatelog <N> | -proxy <IP address>/<port>
```

Table 9.1: updproxy arguments

Argument	Description
-list	Lists all proxy servers registered with the <i>Hitachi ID Bravura Security Fabric</i> server.
-refresh	Updates all proxy servers.
-rotatelog <N>	Rotate the log folder for at most <N> rounds.
-proxy <IP address>/<port>	Specifies which proxy servers to refresh.

Note: Unlike **updst**, **updproxy** cannot be controlled using a configuration file. Since proxies are not full instances, they have different requirements compared to idmsuite instances. Currently **updproxy** works based on a hardcoded list of things that are relevant to proxy instances:

- *Connector Pack* (instance or global)
- Some utilities (**updsvcpass**, **pscp**, **logutil**)
- \<instance>\script\
- \<instance>\lib\
- \<instance>\idmlib\
- \<instance>\license\
- \<instance>\component\
- Services (**idmllogsvc**, **psproxy**)

Examples

1. To show a list of currently registered proxy servers, type:

```
updproxy -list
```

2. To update all the proxy servers, type:

```
updproxy -refresh
```

3. To update a specific server, type:

```
updproxy -refresh -proxy <IP address>/<port>
```

See also:

Proxy Service (psproxy) in the Bravura Security Fabric *Reference Manual* provides details about the Proxy Service (psproxy)

Description

Use the **dbperfchk** program to provide metrics to test database performance in a replication environment.

Usage

```
dbperfchk.exe <options>
```

Table 10.1: dbperfchk arguments

Argument	Description
-a, --all	Run all tests
-ert, --expected-replication-time <N>	The amount of time (in minutes) to wait for replication to complete before timing out; default 10 minutes.
-xt, --maxThreads <N>	Maximum number of threads to test (default 5)
-mt, --minThreads <N>	Minimum number of threads to test (default 1)
-mcl, --multi-commit-latency	Run concurrent commit latency test
-mcr, --multi-commit-replication	Run a multi-threaded test of replication throughput using only one replicated server
-mcrt, --multi-commit-replication-thorough	Run a multi-threaded test of replication throughput using all replicated servers (auto-detect)
-port, --port <port>	Replication secondary server port
-secondary, --secondary <address>	Replication secondary server address
-scl, --single-commit-latency	Run single-threaded commit latency test
-sro, --single-read-ops	Run a single-threaded read-only operations-per-second test
-t --time <N>	Number of milliseconds to run for

Examples

1. To test all replication partners running a multi-threaded test:

```
dbperfchk.exe -mcrt -mt 2 -xt 3
```

2. To test the local instance database latency:

```
dbperfchk.exe -scl
```

or,

```
dbperfchk.exe -mcl
```

3. To check database performance on 10.0.xx.xx server with port 5555 in 60 seconds with a minimum thread of 4 and maximum thread of 6:

```
dbperfchk.exe -a -xt 6 -mt 4 -port 5555 -secondary 10.0.xx.xx -t 60000
```

4. To check database performance on 10.0.xx.xx server with port 5555 in 60 seconds with single thread read-only test:

```
dbperfchk.exe -sro -port 5555 -secondary 10.0.xx.xx -t 60000
```

Description

Use the **iddbadm** program to modify and configure the credentials used by **iddb** to connect to the database backend.

Usage

Run **iddbadm** with the following arguments:

```
iddbadm.exe [-database <database>] [-dbserver <dbserver>]
[-iddbport <iddbport>]
-dbtype <dbtype> -dbuser <dbuser> -instance <instance> -password <password>
[-servicename <servicename>] [-dbversion <dbversion>]
```

Table 11.1: iddbadm arguments

Argument	Description
-database <database>	The database name (required for MSSQL database only)
-dbserver <dbserver>	The database server (required for MSSQL database only)
-iddbport <iddbport>	The database service TCP port
-dbtype <dbtype>	The database type (MSSQL)
-dbuser <dbuser>	The database server user ID
-instance <instance>	The <i>Bravura Security Fabric</i> instance name
-password <password>	The database server user password
-dbversion <dbversion>	The version of the database (required for MSSQL database only)
-showconfig <showconfig>	Show current DBMS backend configuration

Examples

1. To change the DBMS credentials for a MSSQL server:

```
iddbadm -dbtype MSSQL -dbuser mssqluser -instance idminstance -password
dbuserpassword -database dbname -dbserver dbserver.com -dbversion 2008
```

Description

The **servicemove** failover program helps transfer a failed server's workload to a secondary server. The program is run from the secondary server.

For example, upon installation, each Workflow Manager Service (idwfm) is assigned a unique ID on each server. Each Workflow Manager Service (idwfm) handles only the IDs assigned to it. A workflow request, when created, is bound to one of these IDs. So if one of the servers goes down, **servicemove** can be used to transfer the down Workflow Manager Service to an active server, so that the requests bound to this Workflow Manager Service service could be processed.

Usage

```
servicemove.exe [options] ...
```

Table 12.1: servicemove arguments

Argument	Description
-force	Forces a move to occur even when the remote service appears to be running.
-list <server ID>	Specify a server ID to list services for that server, or you can specify "all" to list services for all servers.
-move <service name>	Moves the specified service to this server.
-release <service name>	Releases the specified service back to its home server.
-server <serverID>	Specifies the home server of the service.

Examples

1. To list services on all servers:

```
servicemove.exe -list all
```

2. To make the secondary server take over running the Password Manager service (idpm) on behalf of a failed primary server, and put all the requests in queue:

```
servicemove.exe -server <primary server id> -move idpm
```

3. To force a secondary server to take over running the Password Manager service even though the primary server is currently running:

```
servicemove.exe -server <primary server id> -move idpm -force
```

4. To release the Password Manager service back to its home server:

```
servicemove.exe -server <primary server id> -release idpm
```

This chapter provides details of the location and purpose of files installed by:

- *Hitachi ID Bravura Security Fabric* (p97)
- *Hitachi ID Connector Pack* (p102)

When you install any Hitachi ID Systems product, the default path for program files is:

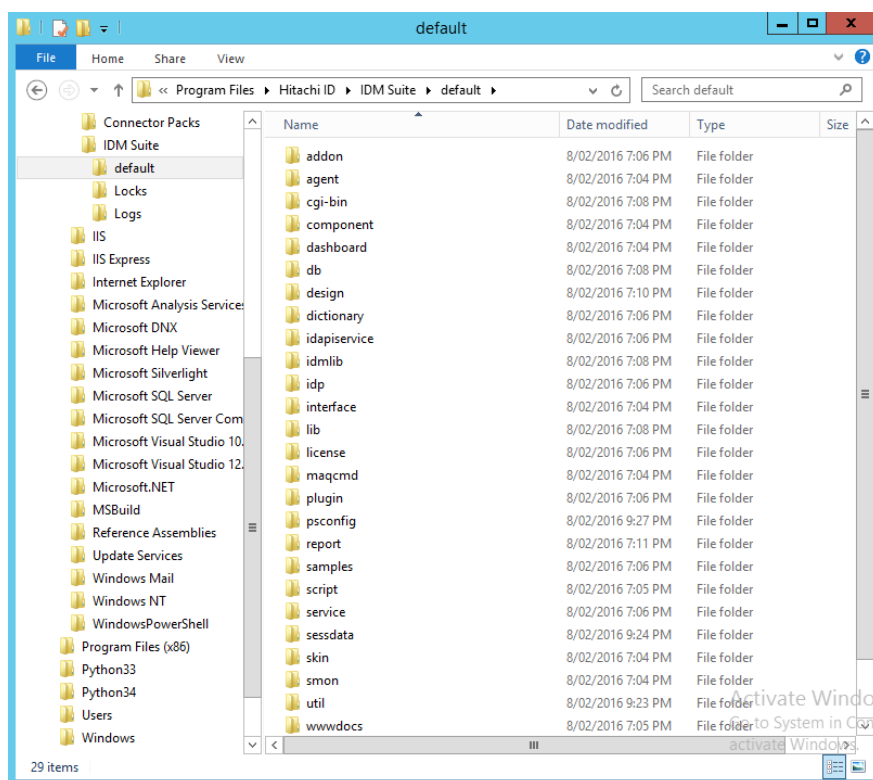
C:\Program Files\Hitachi ID\

13.1 *Bravura Security Fabric* directories and files

There are three main directories that are created when you install *Bravura Security Fabric* instance:

- <Program Files path>\Hitachi ID\IDM Suite\<instance>\
- <Program Files path>\Hitachi ID\IDM Suite\Logs\<instance>\
- <Program Files path>\Hitachi ID\IDM Suite\Locks\

The contents of those directories are detailed in the following subsections.



It is recommended that you do *not* change these directory locations during the setup process. You cannot install any of the directories required for *Bravura Security Fabric* on a mapped drive.

13.1.1 Instance directory

[Instance directory files](#) describes the function of directories that are created when an instance of *Bravura Security Fabric* is installed.

Note: Directories marked with ★ include files installed by *Connector Pack*.

Directories marked with ★★ include folders and files installed with the optional *Analytics* app.

Directories marked with † include optional files. They are only installed in a complete installation or if selected in a custom installation.

Table 13.1: Instance directory files

Directory	Contains
† * addon	Files required for add-on software, such as Password Manager Local Reset Extension and secure kiosk account (SKA). Some files, required to target Netegrity SiteMinder, are installed by <i>Connector Pack</i> . If you installed a global <i>Connector Pack</i> , these files are contained in the <i>Connector Pack</i> global directory.
* agent	Instance-specific user management connectors (agents). If you installed a global <i>Connector Pack</i> , user management connectors are contained in the <i>Connector Pack</i> global directory.
** analytics	<i>Analytics</i> app specific folders
** analytics\DataSets	Contains *.rsd files which are Shared Dataset Definitions. These files are only used by SQL Server versions higher than Express. They contain datasets that are shared between reports.
** analytics\Hidden	Contains *.rdl files which are Report Definitions. These files are the drillthrough reports used by other reports. They are not visible to the end-user.
** analytics\ReportItems	This folder contains other folders. Each folder in this folder is a category in the <i>Analytics</i> app. Within these folders are *.rdl files which are Report Definitions. The folders need to be added to the CUSTOM ANALYTIC CATEGORIES system variable to be visible. These reports are then visible to the end-users in the <i>Analytics</i> app.
cgi-bin	The user web interface modules (*.exe CGI programs).
db	The <i>Bravura Security Fabric</i> database SQL scripts.
db\cache	Search engine temporary search results. These files are cleaned up nightly by psupdate .
db\replication	Stored procedure replication queues, and temporary replicated batch data.
* design	Files necessary to make modifications to the GUI. Some files are installed by <i>Connector Pack</i> . If you install a global <i>Connector Pack</i> , files related to connectors are located in the global design directory. See the Bravura Security Fabric Documentation for details.
dictionary	A flat file, words.dat , that contains dictionary words. <i>Bravura Security Fabric</i> uses this file to determine if new passwords fail dictionary-based password-policy rules.
idapiservice	Files required to use the SOAP API.
* interface	Instance-specific ticket management connectors (exit trap programs). If you installed a global <i>Connector Pack</i> , ticket management connectors are contained in the <i>Connector Pack</i> global directory.
lib	Contains the pslangapi.dll .
license	The license file for <i>Bravura Security Fabric</i> .
plugin	Plugin programs executed by <i>Bravura Security Fabric</i> .

... continued on next page

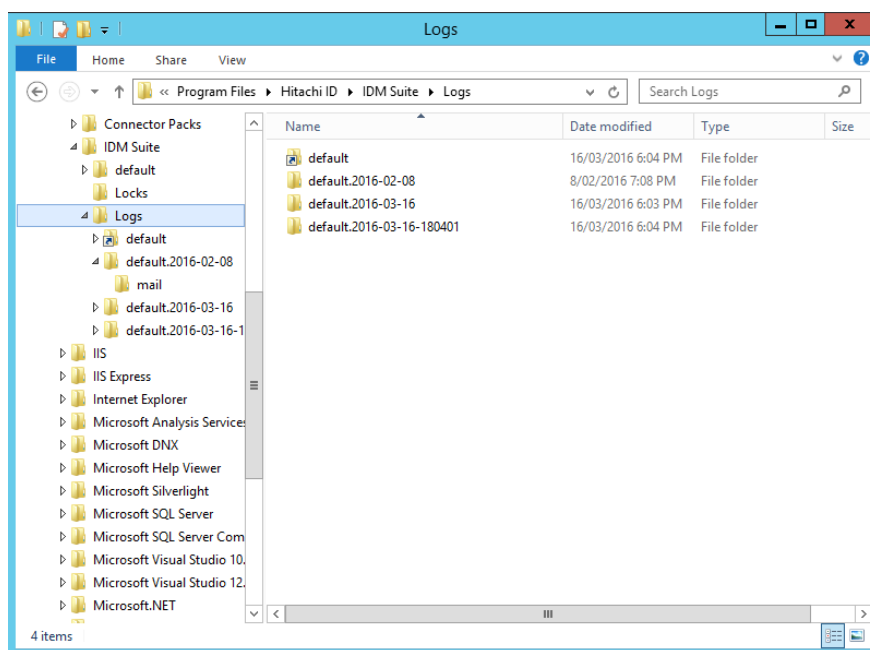
Table 13.1: Instance directory files (Continued)

Directory	Contains
psconfig	List files produced by auto discovery and the <code>idmsetup.inf</code> file.
report	Files and programs for report generation.
† ★ samples	Instance-specific sample scripts and configuration files. If you installed a global <i>Connector Pack</i> , connector-related sample files are contained in the <i>Connector Pack</i> global directory.
script	Configuration files and scripts used by connectors, <code>psupdate</code> , plugins and interface programs.
service	Service programs.
sessdata	Session data. A scheduled program removed old data files nightly.
skin	Compiled GUI files used at run-time (HTML and *.z).
smon	Monitored session data. This location can be changed by <i>Recorded session management</i> (SMON) module options.
★ util	Command-line programs and utilities. If you install a global <i>Connector Pack</i> , tools related to connector configuration are located in the global util directory.
★ unix	The <code>psunix</code> archive, which is required to install the Unix Listener and supporting files on a Unix-based target system. If you installed a global <i>Connector Pack</i> , this directory is created in the <i>Connector Pack</i> global directory.
wwwdocs	Images and static HTML pages used by <i>Bravura Security Fabric</i> .

13.1.2 Log directory

Any operation that is run by *Bravura Security Fabric* is logged. Those logs are invaluable when debugging an issue. The log directory by default is `C:\Program Files\Hitachi ID\IDM Suite\Logs\`. Each instance of *Bravura Security Fabric* that is installed will have at least one sub-directory within this directory.

The `rotatelog` scheduled job, which runs on a nightly basis, rotates the logs in to a new folder, to reduce disk space usage.



See the [Bravura Security Fabric Documentation](#) for more information.

13.1.3 Locks directory

Certain target systems can only be accessed serially, such as Lotus Notes. This is a limitation of the API used to access the target system. In these cases *Bravura Security Fabric* drops a *lock file* in the locks directory when an operation is being performed that should only be performed serially. For this reason the locks directory *must* be the same for all instances of *Bravura Security Fabric* that are installed on the same server.

See the [Bravura Security Fabric Documentation](#) for more information.

13.2 Connector pack directories and files

When you install *Hitachi ID Connector Pack*, files are placed in different locations depending on type of *Connector Pack*.

For an instance-specific connector pack, the installer, **connector-pack-x64.msi**, installs connectors and supporting files in:

<Program Files path>\Hitachi ID\IDM Suite\<instance>\

For a global connector pack, the installer, **connector-pack-x64.msi**, installs connectors and supporting files in:

<Program Files path>\Hitachi ID\Connector Packs\global\

[Connector Pack directory files](#) describes the function of directories that are created when a *Connector Pack* is installed:

Table 13.2: Connector Pack directory files

Directory	Contains
addon	Files required to target Netegrity SiteMinder systems
agent	User management connectors (agents)
design	<i>Connector Pack</i> -related files necessary to make modifications to the GUI; for example target system address help pages. See the Bravura Security Fabric Documentation for details.
interface	Ticket management connectors (exit trap programs)
samples	Sample scripts and configuration files
unix	The psunix archive, which is required to install the Unix Listener and supporting files on a Unix-based target system
util	Tools to support the configuration of various target systems

Part III

APPENDICES

Calculating the Number of Application Nodes Required

A

Hitachi ID Systems *strongly* recommends that you create at least two or three replicated *Hitachi ID Bravura Security Fabric* database nodes for fault tolerance and backup, depending on which products are licensed and which features are implemented. When *Hitachi ID Bravura Privilege* is installed, or when close to 100% uptime is required from the instance, the recommended minimum is three. This is because when backend database replication is performed, at least two database nodes have to go down at the same time.

On the other hand, avoid setting up too many replicated database nodes. It is likely that the additional overhead of any more than six fully replicated nodes would outweigh the advantage of having multiple nodes to do work. If you implement transparent password synchronization, it is recommended that you avoid having more than five concurrently active database nodes.

Calculate the number of database nodes you require with the following formula:

$$N = \text{ceil}(1 + [U/P * 7/5/Z * A]/[60 * 60 * C/T])$$

Where:

U = # of users

P = days between password expiry

Z = time zones

A = accounts/user

C = # of concurrently running connectors

T = seconds on average to change a password

For example, when:

U = 10000 people

P = 45 days

Z = 1 time zone

A = 10 accounts/user

C = 30 connectors at a time

T = 2 seconds

then:

$$N = \text{ceil}(1 + [10000/45 * 7/5/1 * 10]/[60 * 60 * 30/2])$$

$$N = \text{ceil}(1 + [3111]/[54000])$$

$$N = \text{ceil}(1.0576)$$

$$N = 2$$

See also:

- [Installing with a shared schema](#)
- [Configuring database replication](#)