IBM Informix Dynamic Server (IDS)

IDS 11.50 (Cheetah 2)

Techsales Training

MACH 11 (Phase 2)

Compiled By IDS-CTE
anupn@us.ibm.com

Contents

IDS 11 Availability
(Includes MACH-11 Current Functionality)

- **What is MACH-11?**
- **Supporting Infrastructure for MACH-11?**
- **High-Availability Data Replication (HDR).**
- **Shared Disk Secondary (SDS).**
- **Remote Standalone Secondary (RSS).**
- **Enterprise Replication (ER).**
- **Continuous Log Restore (CLR).**
- **Above Combination.**

This section covers the above points in brief to support the explanation of the features in the subsequent sections (those that are available in IDS 11.50). The above features have already been presented in detail during the release of IDS 11.10. Contact IDS-CTE, if a detailed presentation of the above features are needed.

What is MACH-11?

- Multi-Node Active Cluster for High Availability:
  - Extends HDR to support more than a single primary with a single secondary instance.
  - New types of secondary instances:
    - Shared Disk Secondary (SDS).
    - Remote Standalone Secondary (RSS).
- MACH-11 is not:
  - Just 1-to-N HDR.
  - “MACH-11” is the treating of all new forms of secondary as a multi-tiered availability solution.

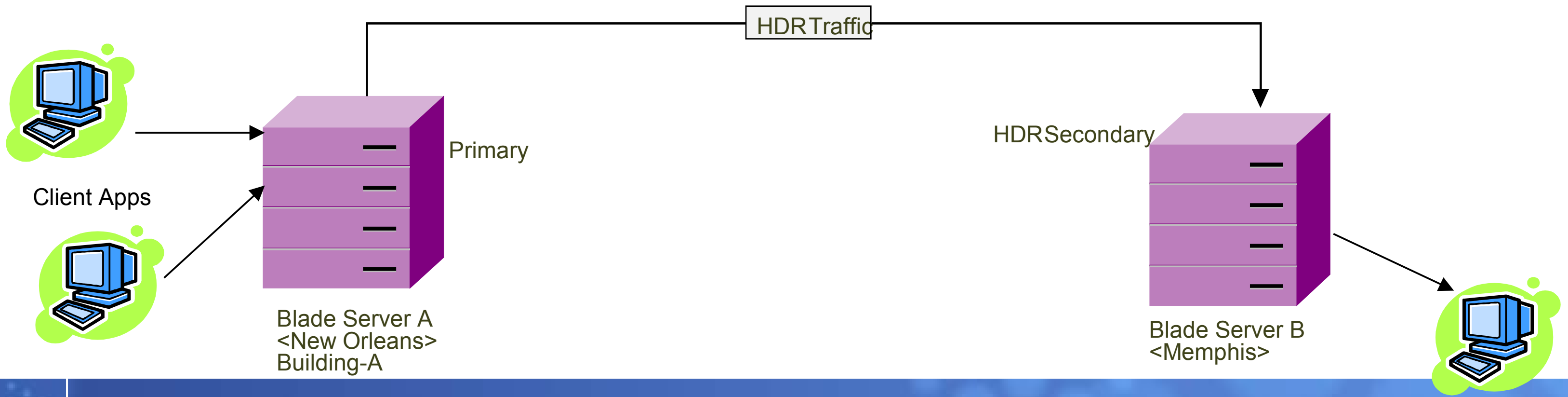
Supporting Infrastructure For MACH-11

- Two New Subcomponents Used to Support MACH-11:
  - Server Multiplexer (SMX):
    - Used internally to establish network connections between the instances.
    - Supports multiple logical connections over a single TCP connection.
  - Index Page Logging:
    - Used to log the pages created as part of an index build.
    - Requirement for RSS.

High-Availability Data Replication (HDR)

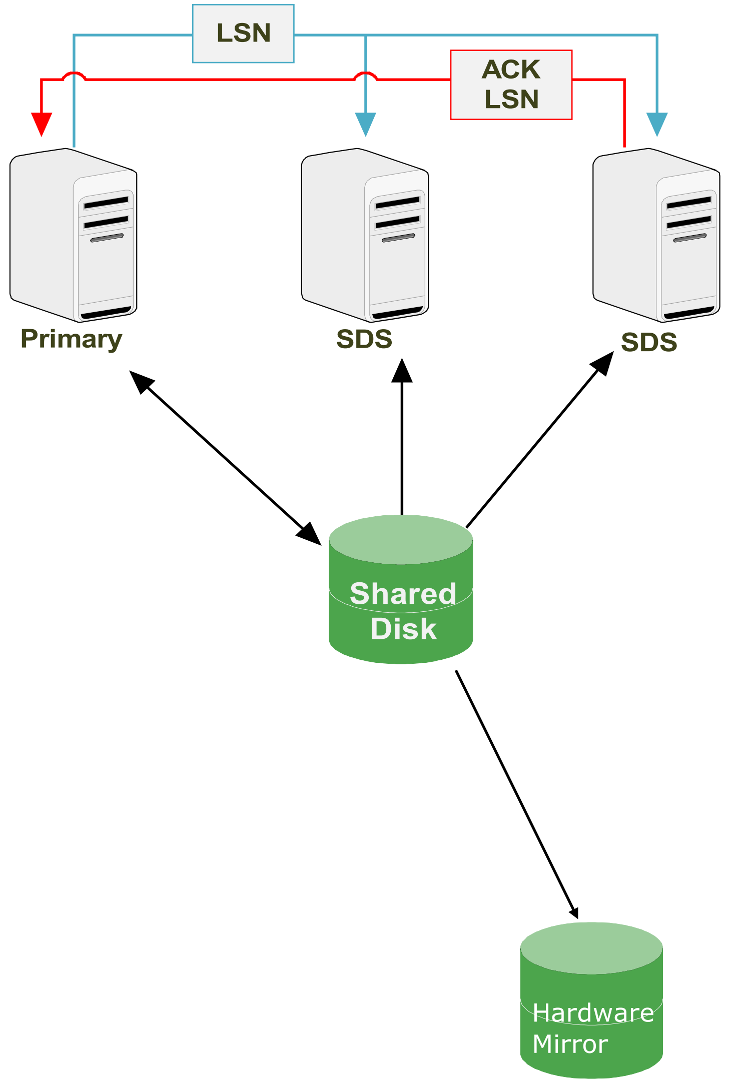
- Two identical servers on two identical machines:
  - Primary server.
  - Secondary server.
- Primary server:
  - Fully functional server.
  - All database activity – insert/update/deletes, are performed on this instance.
  - Sends logs to secondary server.
- Secondary server:
  - Read only server : allows read only query.
  - Always in recovery mode.
  - Receives logs from primary and replay them to keep in sync with primary.

When Primary server goes down, secondary server takes over as Standard server.

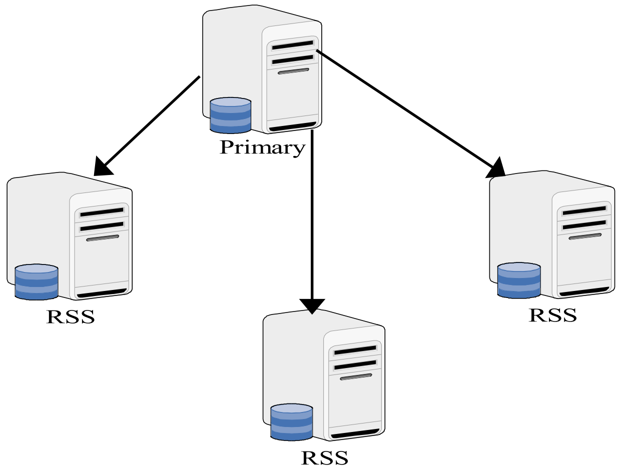


Shared Disk Secondary (SDS)

- HDR on top of a shared disk subsystem.
- Primary transmits the current Log Sequence Number (LSN) as it is flushing logs.
- SDS instance(s) receives the LSN from the primary and reads the logs from the shared disks.
- SDS instance(s) applies log changes to its buffer cache.
- SDS instance(s) resynch processed LSN to primary.



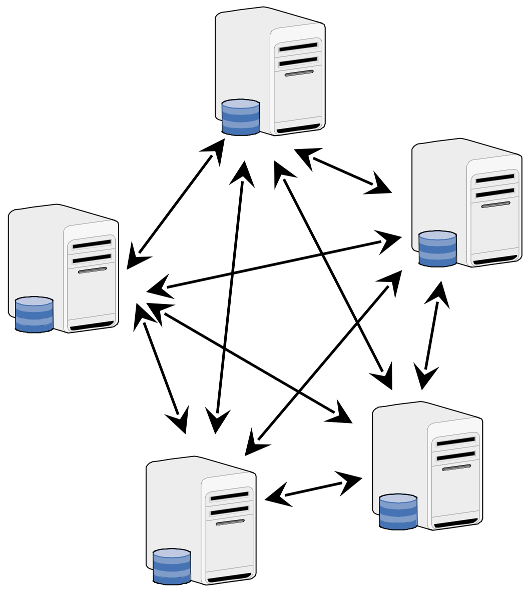
Remote Standalone Server (RSS)



- Similar to HDR:
  - Maintains a full disk copy of the database.
  - Created by performing a backup/restore of the instance
  - Can be used for: Additional Backup, Report processing, Load balancing.
- Distinct from HDR:
  - Uses full duplex communication (SMX)– better throughput over slower lines.
  - Does not support SYNC mode, not even for checkpoints.
  - Can not currently be 'promoted' to primary – but can be promoted to HDR secondary (Focus is on 24 X 7 Availability, Scalability, and Disaster Recovery).
  - There can be any number of RSS instances.
  - Requires Index Page Logging be turned on.
- RSS can be used in combination with HDR secondary:
  - RSS can be converted into HDR secondary.
  - HDR secondary can be converted into RSS.

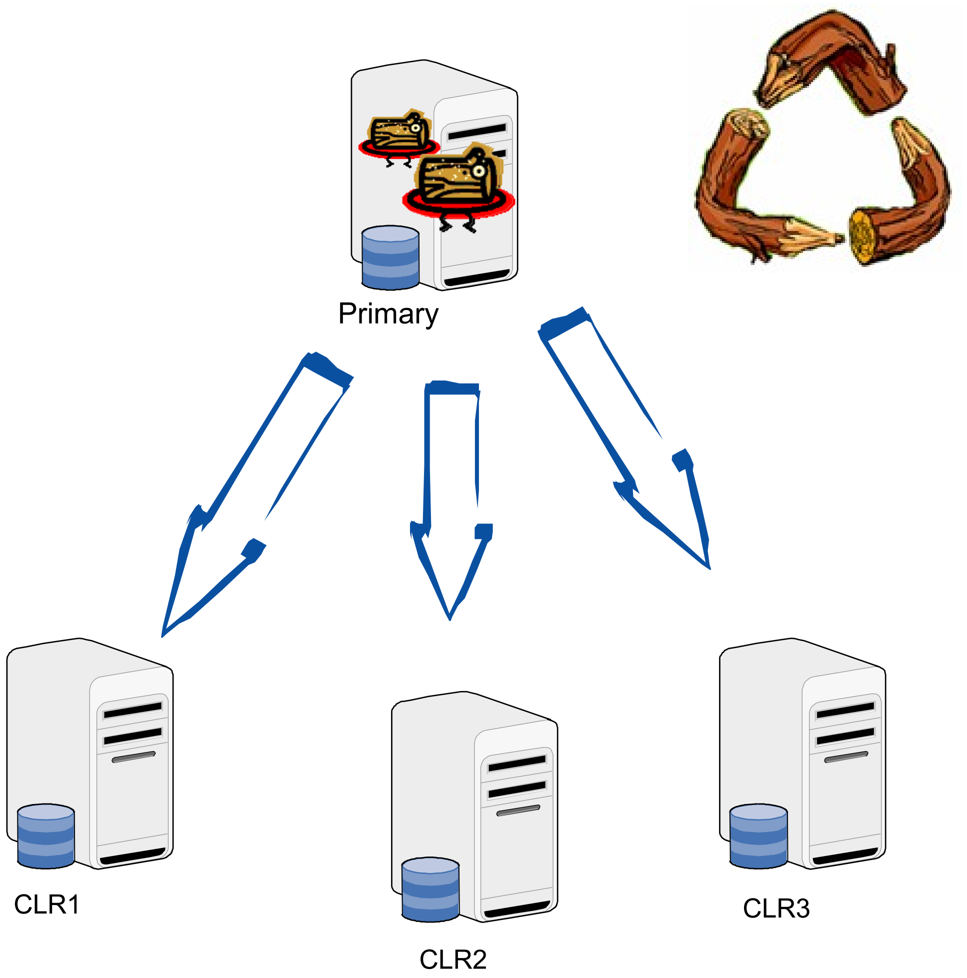
Enterprise Data Replication (ER)

- Uses Workload partitioning for capacity relief and active/active updates.
- Flexible and Scalable to use subset of the data.
- Supports update anywhere while Synchronizing local with global data with very low latency.
- Integrated to be compatible with all other IDS availability solutions with the option of secure data communications.
- Applies Parallelism that enables improved ability to update target tables in parallel which causes reduced response latency back to the source.
- Triggers During Synchronization on ER Servers.

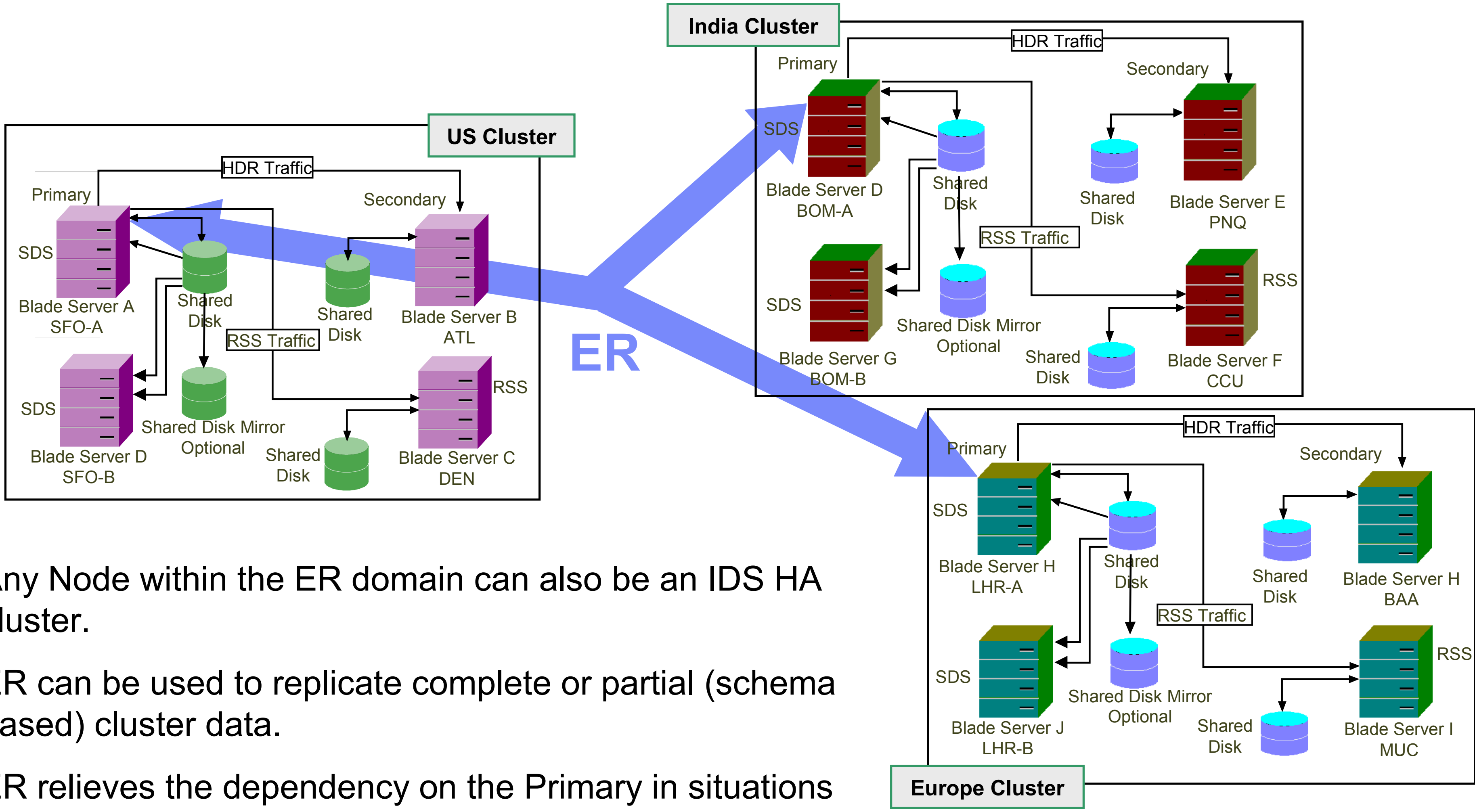


Continuous Log Restore (CLR)

- Also known as “Log Shipping”.
- Allows logical recovery to span multiple ‘ontape/onbar’ commands.
- Provides a secondary instance with ‘log file granularity’.
- Does not impact the primary server.
- Can co-exist with “the cluster” (HDR/RSS/SDS) as well as ER.
- Can be automated by scripting the log backup alarms.
- Useful when backup site is totally isolated (i.e. no network)



Combination Of The Availability Features



- Any Node within the ER domain can also be an IDS HA cluster.
- ER can be used to replicate complete or partial (schema based) cluster data.
- ER relieves the dependency on the Primary in situations such as network outages.

More Availability Features Via IDS 11.50
(MACH 11 Phase 2 New Functionality)

- **Redirected Writes**
- **Connection Manager**
- **Connection Manager Arbitrator**
- **Password Manager**
- **Redirected Writes of Smart BLOBs**

More Availability Features Via IDS 11.50
(MACH 11 Phase 2 New Functionality)

- **Redirected Writes**
- Connection Manager
- Connection Manager Arbitrator
- Password Manager
- Redirected Writes of Smart BLOBs

Redirected Writes

- Client applications can update data on secondary servers by using redirected writes.

Redirected writes gives the appearance that updates are occurring directly on the secondary server when in fact the transaction is transferred to the primary server and then the change is propagated to the secondary server.

- The secondary server is not updated directly.

Redirected Writes : Uses

- Redirects an attempted write (insert, update, and delete) operation from the secondary to the primary node.
- Uses optimistic concurrency to avoid updating a stale copy of the row.
- Works on HDR secondary, RSS nodes, and SDS nodes.
- Redirected writes work on the basic data types and the extended data types, such as: UDTs (those that store data in the server), logged smart BLOBs, and BLOBs stored in tablespaces.
- Only supports DML statements.
- Supports temp tables - both explicit and implicit.
- Works with ER.

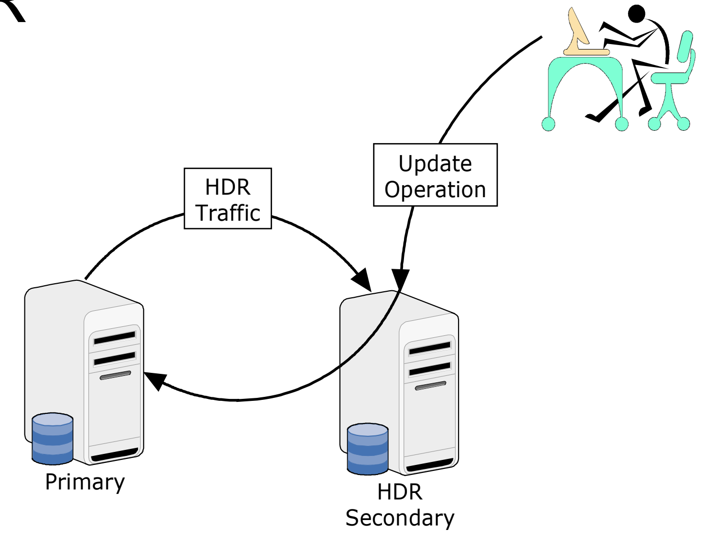
cont'd …

Redirected Writes : Features

- Directs and minimizes collisions on the primary server.
- Provides failover features (such as the ability to continue activity on secondary servers when the primary server fails without restarting).
- Provides support for certain opaque data types, i.e. “deep copy”.
- Only requires that the “REDIRECTED_WRITES” parameter be set in the ONCONFIG file.

How does Redirected Write Work?

- The lower level portion of the updating operation is transferred to a pool of proxy threads on the primary node for execution.
- Triggers and constraint checking is performed on the primary node.
- If the primary node fails and the HDR secondary/RSS/SDS is promoted to the new primary then the redirected writes are sent to the new primary.



Configuring Redirected Writes

- **`REDIRECTED_WRITES`**
  - Set in the onconfig file on the secondary node.
  - Specifies the number of communication pipes and proxy dispatchers which will be used to perform redirected writes.
  - A value of 0 disables redirected writes.
  - There will be one SMX send/receive pair for each of the pipes plus one for normal secondary usage on SDS/RSS.
    - **`REDIRECTED_WRITES`** 3 would result in 4 smxsnd/smxrcv threads on a SDS or RSS connection.

Value of REDIRECTED_WRITES should be set to no more than two times the number of CPU virtual processors.

Optimistic Concurrency

- A update technique used by many web applications to allow disconnected transactions.
- Basically an update is allowed to proceed if the row about to be updated has not been updated by some other activity.
- Uses either a comparison of the before image of the row or some form of row versioning.

Optimistic Concurrency and Redirected Writes

- If it is determined that the before image on the secondary is different than the current image on the primary, then the write operation is not allowed and an EVERCONFLICT (-7350) error is returned.
  - If the table is created with Column Versions, then only they are compared.
  - If the table does not have Column Versions, then the entire before image of the row from the secondary is compared with the primary.

Row Versioning

- Used for determining if a row was changed and to detect collisions.
- With row versioning enabled, each row of a table is configured to contain both a checksum (ifx_insert_checksum) and a version number (ifx_row_version).
- Data collisions can be detected by comparing ifx_insert_checksum and ifx_row_version between the secondary and the primary servers.
- Row versioning is not required to perform redirected writes, but enabling it will reduce network utilization and improve performance.

Creating Row Versioning

```
CREATE TABLE …. WITH VERCOLS
ALTER TABLE ….   [ADD]/[DROP] VERCOLS
```

- Creates a shadow column consisting of an insert checksum value (ifx_insert_checksum) and update version column (ifx_row_version).
- The ifx_insert_checksum remains constant for the life of the row.
- The ifx_row_version column is incremented with each update of the row.

Row Versioning examples

- To add row versioning to an existing table, use the following syntax:

```
ALTER TABLE tablename add VERCOLS;
```

- To delete row versioning from a table with the following syntax:

```
ALTER TABLE tablename drop VERCOLS;
```

- To create a new table with row versioning, use the following syntax:

```
CREATE TABLE tablename (
Column Name Datatype
Column Name Datatype
Column Name Datatype
) with VERCOLS;
```

ifx_row_id virtual column

- New hidden virtual column consisting of a character representation of the following separated by a colon:
  - partition Number.
  - rowid.
  - ifx_insert_checksum.
  - ifx_row_version.

  Example: `1048928:257:741480809:1`

- If table is not created with version columns, then ifx_insert_checksum and ifx_row_version is not present :

  Example: `7324334:258`

Example of using Row Versioning

Update using update cursor/pessimistic locking. Current mechanism used by applications.

```
/* declare and open a cursor  */
$declare cur1 cursor for select * from T1 for
update;

$open cur1;


/* fetch data */
$fetch cur1 into :my_id, :my_name;


/* process the data and make a decision */
do_update = get_user_input(my_id, my_name);


/* update the row if required */
if (do_update)
{
  $update T1 set name = :my_name where
current of cur1;
}
```

Optimistic locking using version shadow column

```
/* Alter table to add version columns */
$alter table T1 add vercols;

/* declare and open a cursor with row version */
$declare cur1 cursor for select *, ifx_row_id from T1;

$open cur1;

/* fetch data */
$fetch cur1 into :my_id, :my_name, :row_id;

/* process the data and make a decision */
do_update = get_user_input(my_id, my_name);

/* update the row if required */
if (do_update)
{
  /* compare current ifx_row_id with the one read earlier */
  $update T1 set name = :my_name where id = :my_id and
ifx_row_id = :row_id;

  /*check whether update succeeded, raise warning otherwise*/
  if (sqlca.sqlerrd[2] < 1 )
    raise_warning(“trying to update stale data”);
}
```

More Availability Features Via IDS 11.50

(MACH 11 Phase 2 New Functionality)

- Redirected Writes
- **Connection Manager**
- Connection Manager Arbitrator
- Password Manager
- Redirected Writes of Smart BLOBs

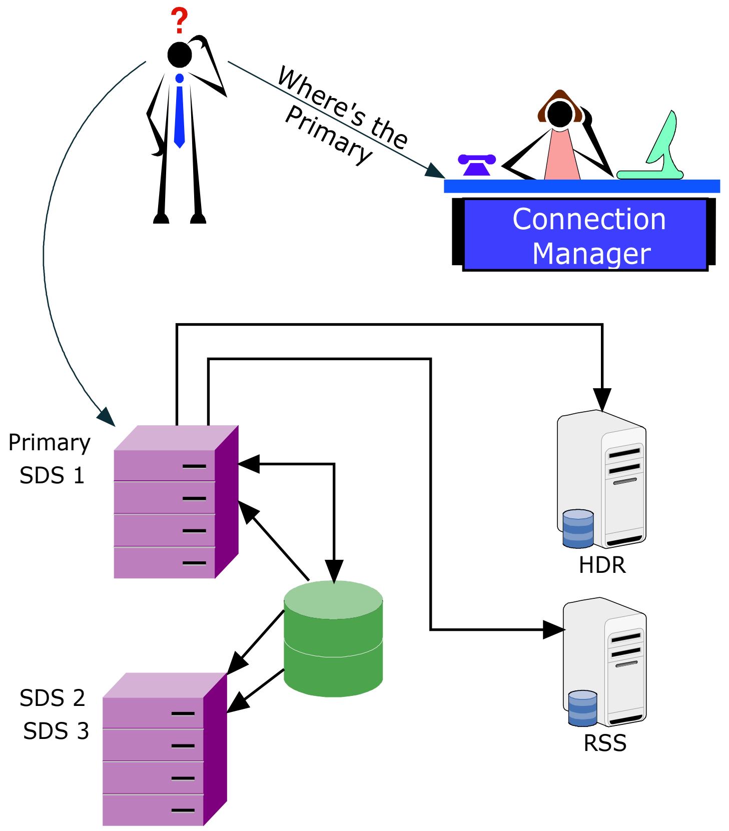
Connection Manager

- The Connection Manager, 'oncmsm' is a daemon program which accepts a client connection request and then re-routes that connection to one of the best fit nodes in the IDS cluster.

- oncmsm : **onl**ine **C**onnection **M**anager and **S**erver **M**onitor,

- Monitors all of the members of the Mach11 Cluster.

- For maximum availability, It is recommended that 'oncmsm' be run on a different machine than the server (not a requirement though).



Connection Manager

- Works with CSDK, JDBC and JCC:
  - DRDA/CLI will come post release.
- Resolves connection request based on Service Level Agreement (SLA):
  - Connect to primary.
  - Connect to best secondary.
  - Connect to primary or SDS based on workload.

Before you configure and use Connection Manager, you must install the Client SDK

Notations: SLA based Connection Rerouting

SLA: Service Level Agreement

It is a contract that exists between client applications and their service providers, IDS servers. The SLA is based on probability of response to performance and quality of data. That is, if the user requires that data be current as of NOW, then it must be connected to the primary node. If the user can accept some fuzziness, but still needs to view data that is very close to current, then it could be directed to one of the SDS. If the user can not accept dirty reads, but can accept some degree of latency, then it would be directed to an ER secondary that network services is aware of.

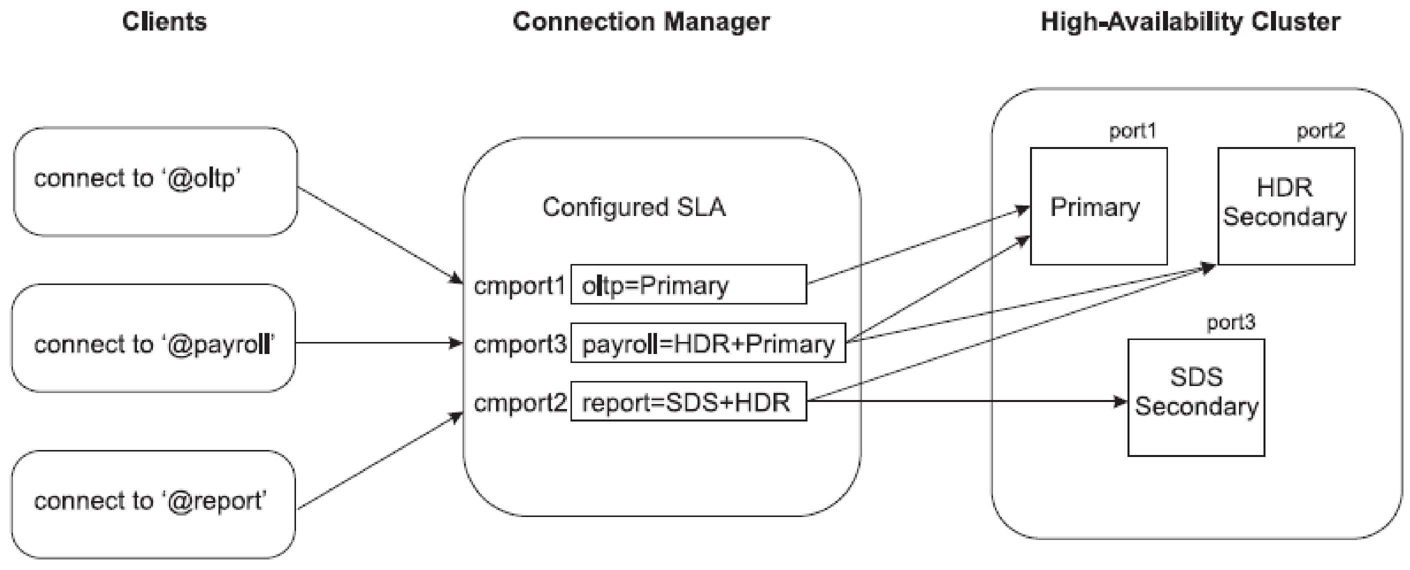
For example:

- **SLA OLTP=primary**
- **SLA payroll=primary+SDS**
- **SLA report=HDR+SDS**
- **SLA accounting=SDS+HDR+primary**
- **SLA HR=HDR+primary**

SLA based Connection Rerouting example

The following figure illustrates the configuration with three SLA as follows:

- SLA oltp=primary
- SLA payroll=HDR+primary
- SLA report=SDS+HDR



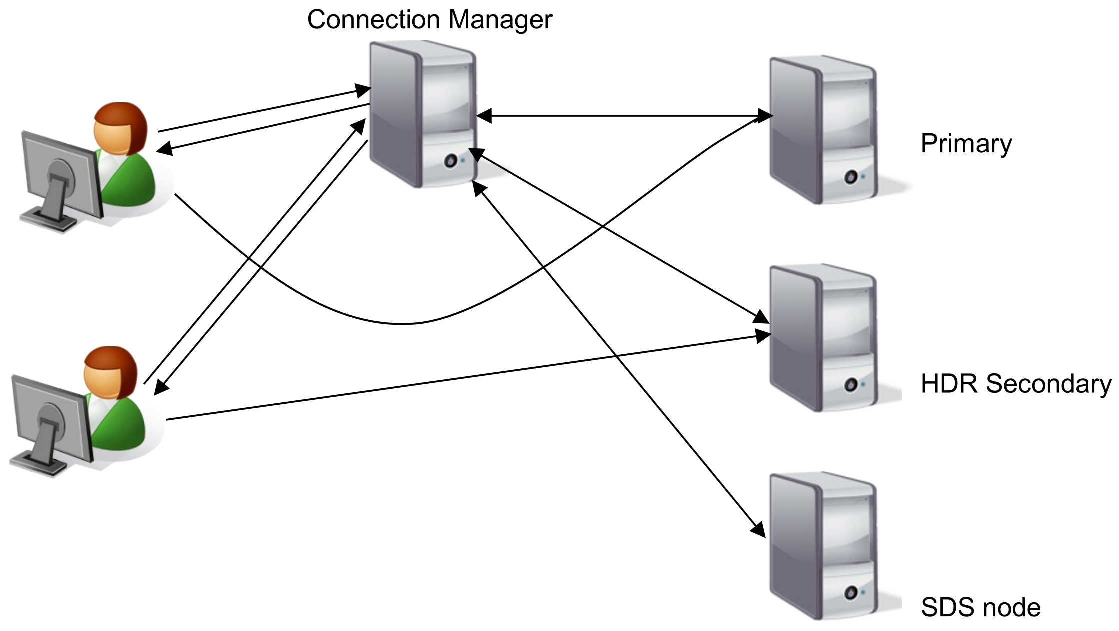
sqlhosts file on Connection Manager and Client machines:

```
ids     onsoctcp host1 port1
oltp    onsoctcp cmhost1 cmport1
report  onsoctcp cmhost1 cmport2
payroll onsoctcp cmhost1 cmport3
```

Cluster sqlhosts file:

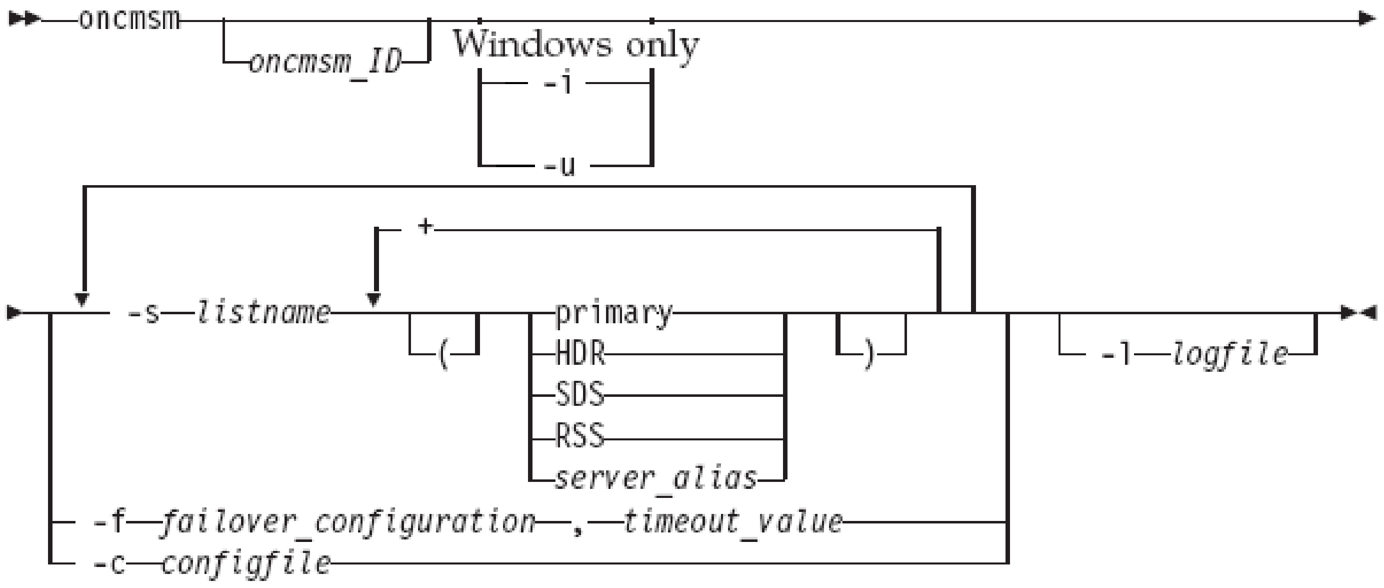
```
g_ha group - - i=10
ids onsoctcp host1 port1 g=g_ha
ids_hdr onsoctcp host2 port2 g=g_ha
ids_sds onsoctcp host3 port3 g=g_ha
```

Connection Manager reroute chart



- Clients have the option of connecting to the servers directly or via the connection manager.
- Clients connected via the connection manager will have the ability of automatic connection rerouting in case of failures.

Connection Manager Daemon Command



- *oncmsm_ID :* Name of the Connection Manager instance.
- *server_alias :* Used to specify a server alias name.
- **-s** *listname :* Used to identify the SLA of a given listener port.
- **-f** *failover_configuration :* Specifies the secondary servers for failover.
- *timeout_value :* Specifies the time (in seconds) to wait before performing failover.
- **-c** *configfile :* Optional configuration file that will contain the SLA requirements.
- **-l** *logfile :* Used to specify an optional log file.
- **-i/-u** *:* (Windows Only) Install/Uninstall oncmsm as windows service.

Connection Manager Daemon Command

Examples

```
% oncmsm
% oncmsm -c /path/to/config/file
% oncmsm  cm1 -s oltp_cm1=primary -s report_cm1=HDR+SDS
% oncmsm -s oltp=primary -s payroll=HDR+primary -s
  report=SDS+HDR -l cm.log
```

Connection Manager Setup

- The Connection Manager configuration file:
  - Default **`$INFORMIXDIR/etc/cmsm.cfg.`**
- SQLHOSTS
- User Password defined with '`onpassword`'.
- Environment Variables:
  - **`INFORMIXDIR`**
  - **`INFORMIXSERVER`**
  - **`INFORMIXSQLHOSTS`**

Connection Manager Setup: configure file

- The format of the configuration file is as follows:

```
NAME     ConnectionManagerName
SLA      name=value
SLA      name=value
FOC      failover_configuration,timeout_value
DEBUG    1/0
LOGFILE <path to log file>
```

- FOC is the Fail Over Configuration parameter.
- *value* can refer to a server name, a server type, or a server list.
- The default configuration file name and location is:
  `$INFORMIXDIR/etc/cmsm.cfg.`

SQLHOSTS Sample

- Connection Manager:

```
ids       onsoctcp    idshost    ids
oltp      onsoctcp    cmhost     oltp
Report    onsoctcp    cmhost     report
```

- IDS:

```
ids       onsoctcp    idshost     ids
sds       onsoctcp    idshost     sds
rss       onsoctcp    idshost     rss
```

- Client:

```
oltp      onsoctcp    cmhost     oltp
report    onsoctcp    cmhost     report
```

Failover of Connection Manager (sample configuration)

- Connection Manager 1:

```
ids            onsoctcp  idshost  ids
oltp1          onsoctcp  cmhost1  oltp
report1        onsoctcp  cmhost1  report
```

- Connection Manager 2 :

```
ids            onsoctcp  idshost  ids
oltp2          onsoctcp  cmhost2  oltp
report2        onsoctcp  cmhost2  report
```

- Client:

```
oltp_group     group     -        -       i=1
oltp1          onsoctcp  cmhost1  oltp    g=oltp_group
oltp2          onsoctcp  cmhost2  oltp    g=oltp_group
report_group   group     -        -       i=2
report1        onsoctcp  cmhost1  report  g=report_group
report2        onsoctcp  cmhost2  report  g=report_group
```

Client Application

- `Connect to '@oltp'`
  Always can connect to the primary server.

- `Connect to '@report'`
  Connect to any secondary node based on node latency and workload.

Connection Manager Statistics

`onstat -g cmsm`

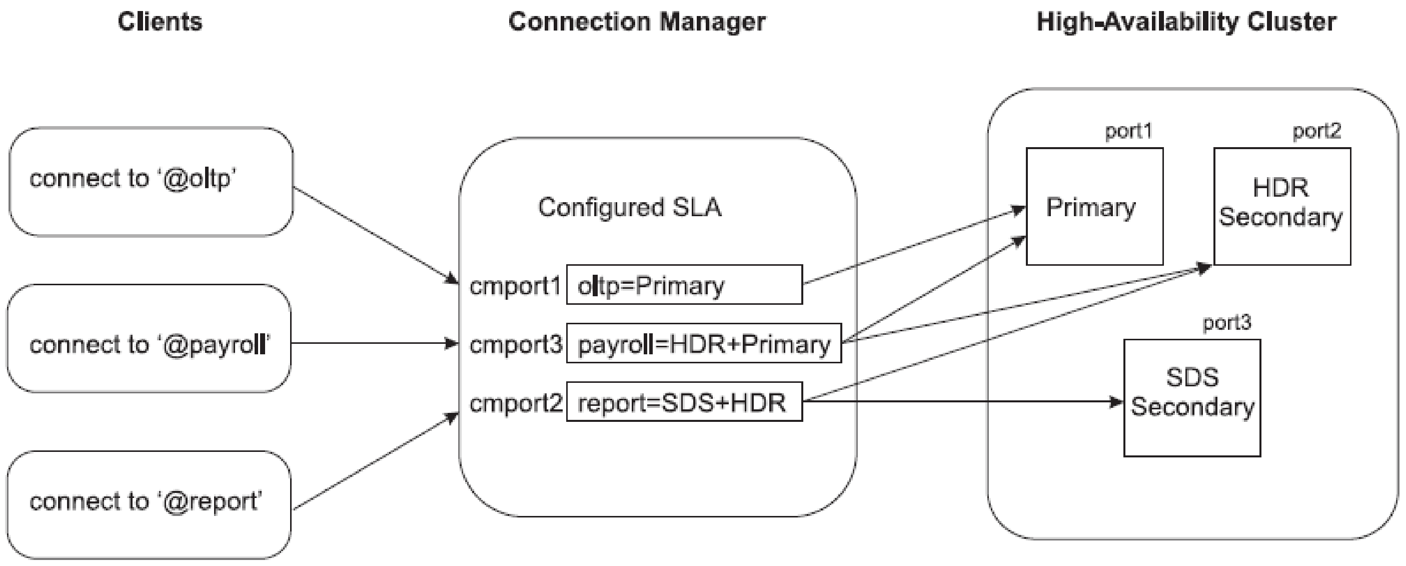
- Displays the various Connection Manager daemons and corresponding details that are attached to a server instance.
- Display contents:
  - All connection managers inside cluster.
  - Associated hosts.
  - SLA and corresponding defines.
  - Failover Configuration.
  - Flags & statistics.
- Sample Output:

```
CM name host sla     define     foc            flag  connections
Cm1     bia  oltp    primary    SDS+HDR+RSS,0  3     5
Cm1     bia  report  (SDS+RSS)  SDS+HDR+RSS,0  3     16
```

SLA based Connection Rerouting example walkthrough …

The following figure illustrates the configuration with three SLA as follows:

- SLA oltp=primary
- SLA payroll=HDR+primary
- SLA report=SDS+HDR



sqlhosts file on Connection Manager
and Client machines:

```
ids     onsoctcp host1 port1
oltp    onsoctcp cmhost1 cmport1
report  onsoctcp cmhost1 cmport2
payroll onsoctcp cmhost1 cmport3
```

Cluster sqlhosts file:

```
g_ha group - - i=10
ids onsoctcp host1 port1 g=g_ha
ids_hdr onsoctcp host2 port2 g=g_ha
ids_sds onsoctcp host3 port3 g=g_ha
```

cont'd …

SLA based Connection Rerouting example walkthrough …

To define the three SLAs and configure the system:

1. Configure the IDS Cluster server INFORMIXSQLHOSTS file :

```
ids          onsoctcp     host1       port1
ids_hdr      onsoctcp     host2       port2
ids_sds      onsoctcp     host3       port3
```

2. Configure the INFORMIXSQLHOSTS file on the Connection Manager machine and the Client machine:

```
ids          onsoctcp     host1       port1
oltp         onsoctcp     cmhost1     cmport1
payroll      onsoctcp     cmhost1     cmport2
report       onsoctcp     cmhost1     cmport3
```

3. Set the INFORMIXSERVER environment variable:

`setenv INFORMIXSERVER ids` (in csh).

cont'd …

SLA based Connection Rerouting example walkthrough …

4. Start Connection Manager from a command line to configure the SLA:

```
oncmsm -s oltp=primary -s payroll=HDR+primary -s
report=SDS+HDR -l cm.log
```

- Requests for **oltp** connections will be rerouted to the primary server only. Failover, if any, happens automatically. Rerouting to a new server is transparent to the client applications if connecting via: `connect to @oltp`

- Connection requests to **payroll** are rerouted to both the HDR secondary and to the primary servers: `connect to @payroll`

- Connection requests to **report** will be rerouted to both the HDR secondary and the SDS servers. Failover of HDR secondary goes to SDS server via: `connect to @report`

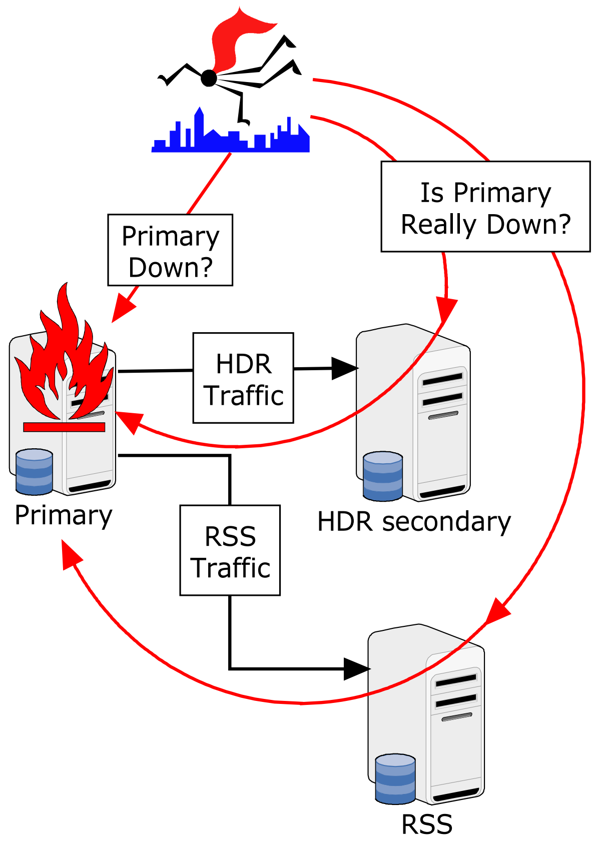
More Availability Features Via IDS 11.50

(MACH 11 Phase 2 New Functionality)

- Redirected Writes
- Connection Manager
- **Connection Manager Arbitrator**
- Password Manager
- Redirected Writes of Smart BLOBs

Connection Manager Arbitrator

- The Connection Manager Arbitrator provides automatic failover logic for high-availability clusters.
- Also known as the Failover Arbitrator.
- Monitors all nodes checking for the primary failover.
- Performs failover (i.e. promote secondary to primary) when it is confirmed that the primary is down.
- Released as part of the connection manager.
- Will support failover to an RSS node.



Fail Over Configuration (FOC) Parameter

- Order of failover defined in the parameter in the Connection Manager Configuration File `($INFORMIXDIR/etc/cmsm.cfg).`
- FOC parameter format:

  `FOC` ***`failover_configuration,timeout_value`***

  Where:

  failover_configuration: One or more of Primary, SDS, HDR, RSS, or a group of server types separated by a plus (+) enclosed in parentheses.

  `timeout_value`*:* *A*mount of time (in seconds) to wait before attempting a failover.

- Default of the FOC parameter:

  ```
  FOC SDS+HDR+RSS,0
  ```

FOC Parameter

- Generic Key Words: `primary, HDR, RSS, SDS`

  Example:

  ```
  FOC    serv1+(serv2+SDS)+HDR+RSS,10
  ```

  Failover first to serv1, then to the best choice of serv2 or any running SDS node, then to the HDR secondary, then to any RSS node. Allow 10 seconds before performing the failover process.

- The FOC can be passed to the connection manager (oncmsm) on the command line as "`-f serv1+(serv2+SDS)+HDR+RSS,10`".

More Availability Features Via IDS 11.50

(MACH 11 Phase 2 New Functionality)

- Redirected Writes
- Connection Manager
- Connection Manager Arbitrator
- **Password Manager**
- Redirected Writes of Smart BLOBs

Onpassword

- The “onpassword” utility is used to encrypt/decrypt the password file.
- These are the passwords that the connection manager uses to connect to the various nodes. It is not used by the client application.
- The password file contains the server names, user names and passwords used to establish connections.
- Due to the sensitive nature of the data in the password file, the file is encrypted.

Password file structure

- The password file is an ASCII text file with the following structure:

```
Servername Alternateserver Username Password
```

  - **`Servername`**: Name of the server to connect to.
  - **`Alternateserver`**: Name of the alternate server to connect to in case the connection to the primary server cannot be established.
  - **`Username`**: Name of the user to use for the connection.
  - **`Password`**: Password to be used for the connection.

Password file example

Examples of password file contents:

```
ServerName_1   AlternateServer_1   UserName_1  Password_1
ServerName_2   AlternateServer_2   UserName_2  Password_2
lx-rama        lx-rama             ravi        foobar
toru           toru                usr2        fivebar
seth           seth                fred        9ocheetah
cheetah        panther             anup        c0mpl1cate
```

The third row shows that “lx-rama” is both the **`Servername`** and the **`Alternateservername`**. The username is “ravi” and the password is “foobar”.

Onpassword syntax

```
onpassword –k Key –e/-d Filename
```

- `Key` : access key used for encryption or decryption.
- `Filename:`
  - –e option : file that needs to be encrypted.
  - –d option : output of the decrypted file.
- The –d option is used to decrypt the file.
- The –e option is used to encrypt the file.
- The encrypted file is copied to `$INFORMIXDIR/etc/passwd_file.`
- During decryption, `$INFORMIXDIR/etc/passwd_file` is decrypted and placed into `Filename.`

Prerequisites

- The following prerequisites have to be satisfied before running onpassword:
  - The $INFORMIXDIR environment variable has to be set to a valid value.
  - Only user “informix” can execute the onpassword utility.
  - The access key length cannot be greater than 24 characters.

More Availability Features Via IDS 11.50
(MACH 11 Phase 2 New Functionality)

- Redirected Writes
- Connection Manager
- Connection Manager Arbitrator
- Password Manager
- **Redirected Writes of Smart BLOBs**

Redirected Writes Of Smart BLOBs (SLOBs)

- Redirected writes are supported only for logged SLOBs.
- To log SLOBs:
  - `onspaces -c -S …. -Df "logging=on"`
  - Or log when you create each object.
- Without logging, updates will not go back to secondary node – secondary reads return 0 length.

Redirected Writes Of SLOBs

- Replication makes the secondary writes go to the primary.
- Same lock wait mode is observed: no wait, wait, or time value.
- Each operation on the primary transmits an “ack” and completion code back to the secondary.
- Updates sent to the primary from the secondary will fail if the secondary reads a stale data value for the same object in the same transaction on the secondary.
- Get back “optimistic concurrency error” – retry the read/write operation again in new transaction.
- Reads are resolved on the secondary if possible.

Redirected Writes Of SLOBs

- Check the last log record written for this secondary node and SLOB.
- If this log record has not been replayed then send the read request to the primary.
- Note that the log record is the last log record written by all transactions on the secondary node.
- In most scenarios, most SLOB reads are done on the secondary node – not sent to the primary.
- SLOBs does not wait for the log record to come back to the secondary before performing additional operations on a SLOB – greatly reduces elapse time.

Thank You