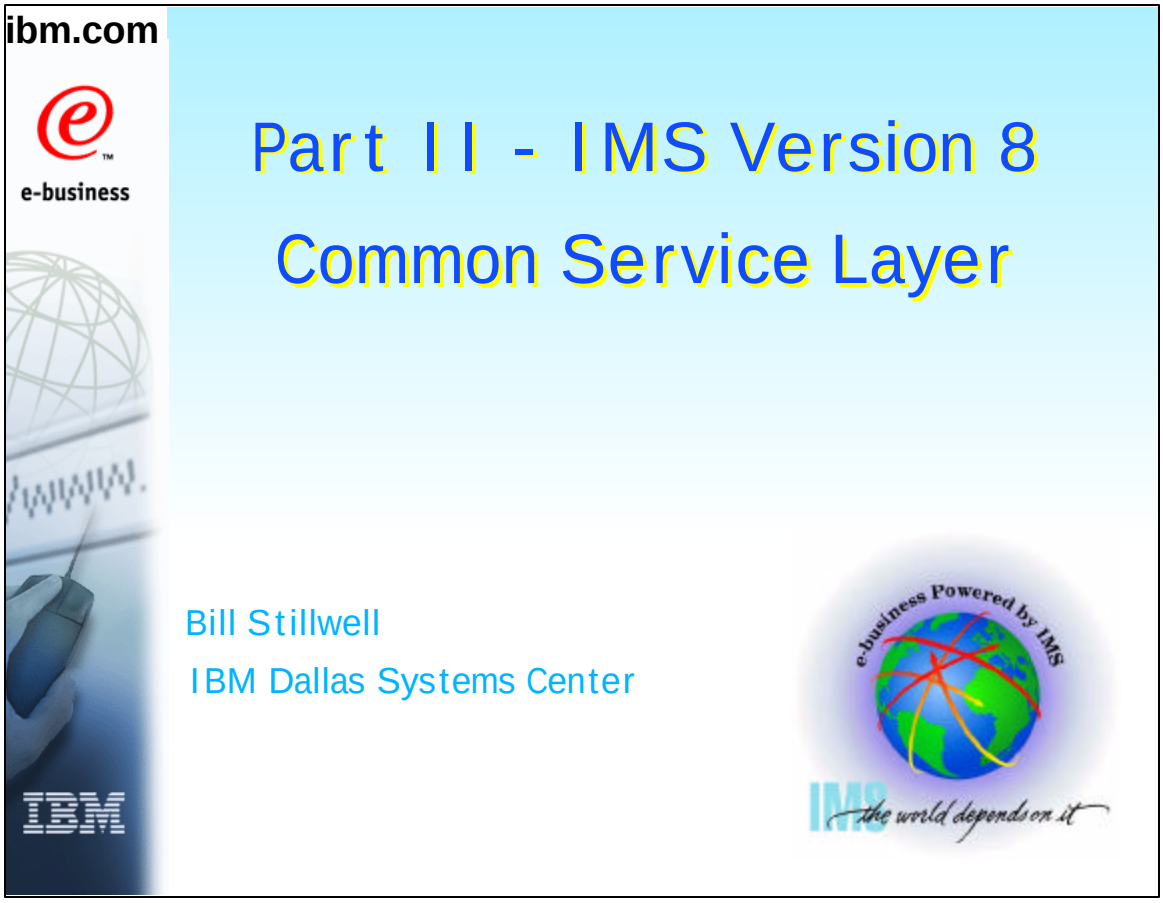


This part addresses a new architecture and systems management functionality in IMS V8. The architecture is called the Common Service Layer, or CSL.

IMS V8 Highlights

Enhancements to ...

- Database Manager
- Transaction Manager
- Systems
- Applications

Enhancements to ...

- Parallel Sysplex
- Common Service Layer

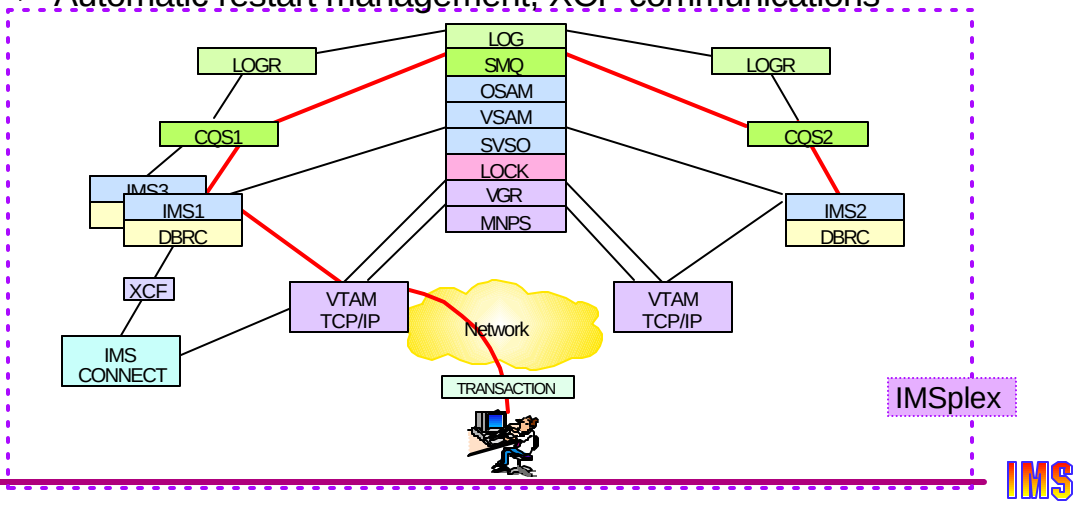
IMS

This topic addresses the Common Service Layer. The CSL represents the next step in IMS architectural evolution.

By the End of IMS V7

IMS had exploited many parallel sysplex functions to share resources in an **IMSplex**

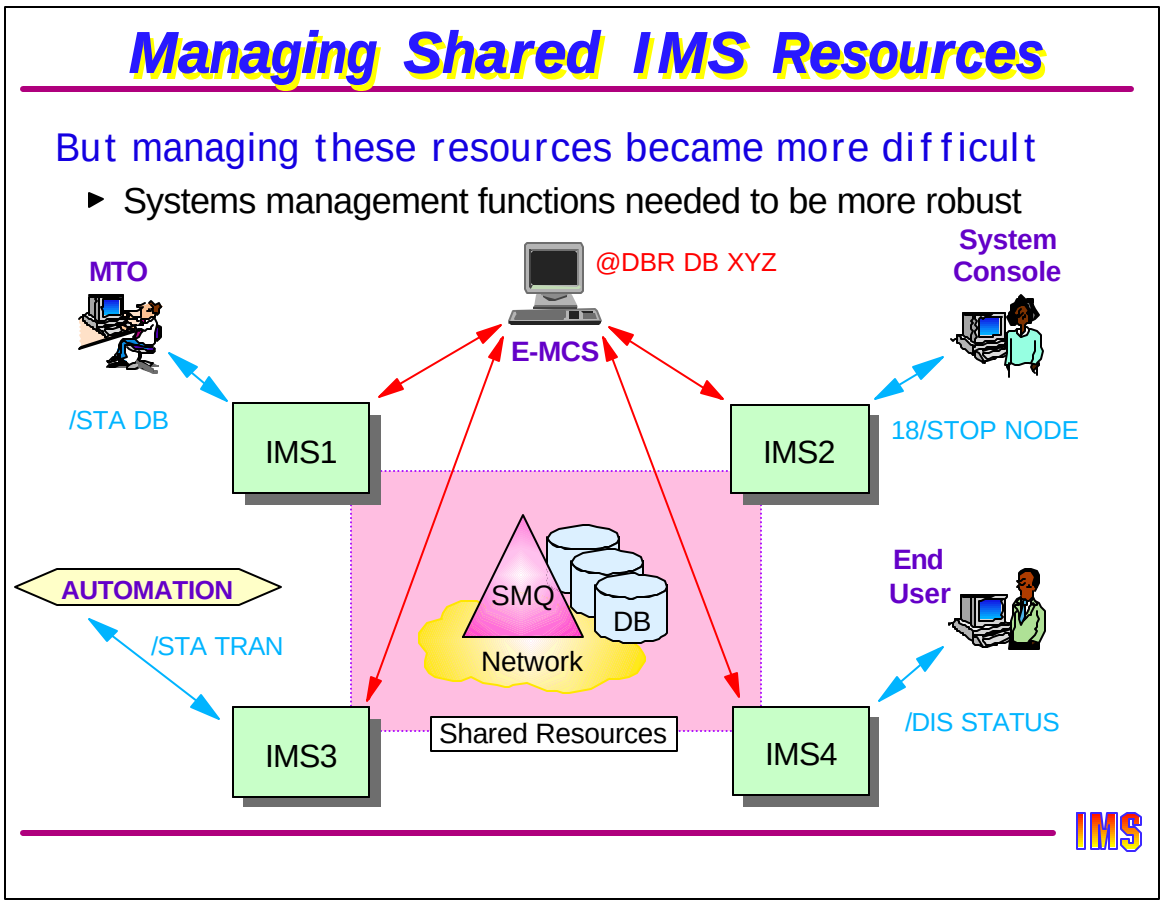
- Data sharing, shared queues
- VTAM generic resources, multinode persistent sessions
- Automatic restart management, XCF communications



By the end of Version 7, IMS had exploited the parallel sysplex for a multiple data sharing and connectivity functions. These include:

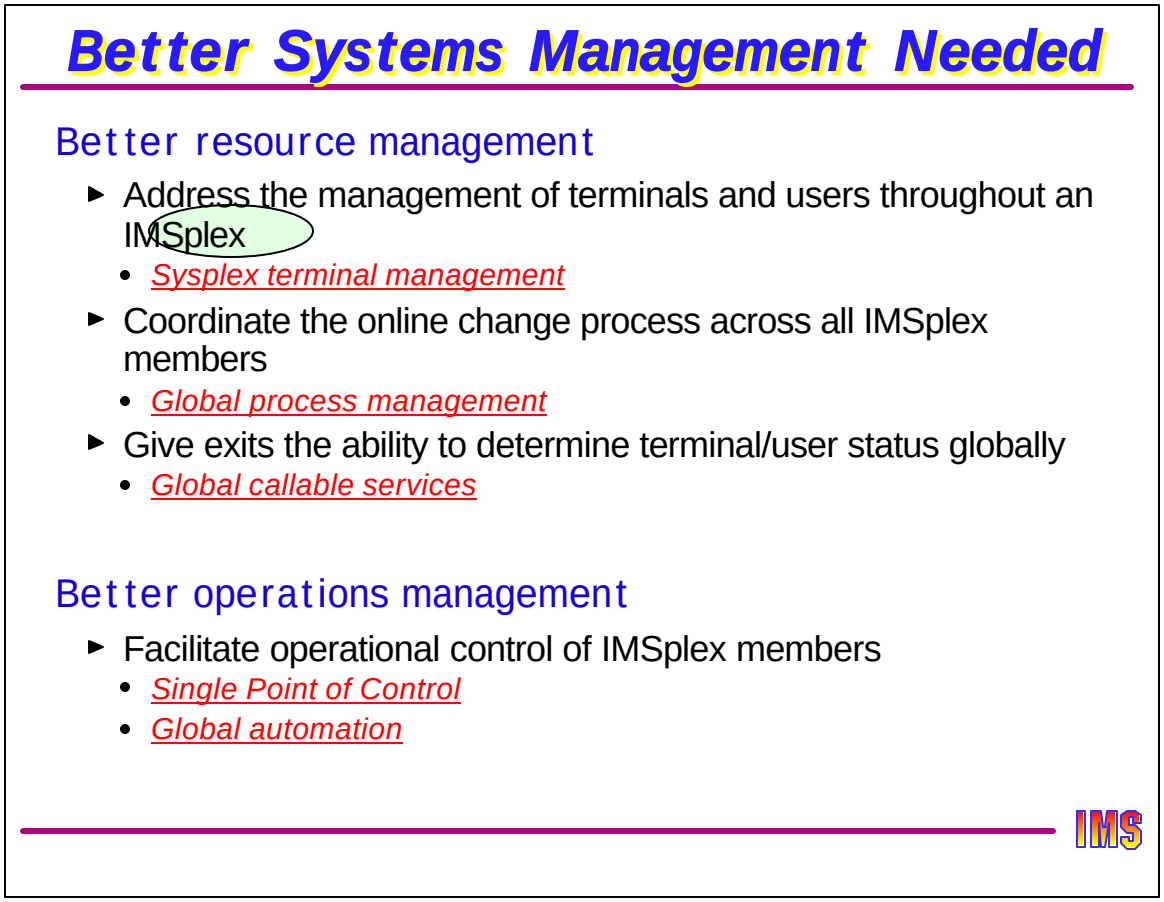
- Data sharing
- Shared queues, which also uses the MVS System Logger
- VTAM generic resources
- VTAM multinode persistent sessions (IMS's implementation is called Rapid Network Reconnect)
- XCF communications (IMS Connect and intra-IMSplex communications)
- Plus a variety of XCF services (group services, signalling services, monitoring services)

These functions evolved over many years, from IMS V1R2 when data sharing was introduced, right up through IMS V7



The reason for all this is, of course, is to share resources across multiple IMSs in a parallel sysplex, or IMSplex. But while this has improved availability, capacity, and performance, it has at the same time made the task of managing these resources more difficult. For example, IMS commands used to control the resources must be entered to each IMS individually and, with a few exceptions for databases, take effect only on the IMS where the command is entered. IMS V6 began support of the Command Recognition Character so that a command entered from an MSC or E-MSC console can be "routed" to all IMSs, but this is really not part of IMS, and is not particularly user friendly.

As the number of IMSs in the IMSplex grows larger, management of these resources becomes more difficult.



Better capabilities for managing the operations of the IMSplex and its resources were needed. Version 8 identifies two areas for improvement.

Resource Management
- Sysplex terminal management applies to the management of terminals and users within the IMSplex.
- Global process management is the coordination of processes across all members of the IMSplex. An example is Global Online Change.
- Global callable services allows user exits to determine the status of terminal, lterm, and user resources IMSplex-wide.

Operations Management
- Single point of control is one or more entry points into the IMSplex where commands can be entered to any or all IMSs and consolicated responses from those IMSs can be received.
- Global automation is an interface which allows user- or vendor-written automation software to issue commands and receive consolidated responses.

The IMSplex

Definition of an IMSplex

- An IMSplex is a set of IMS address spaces that are ***working together as a unit*** and are most likely running in a parallel sysplex with a *common service layer (CSL)*
  - Note: The IMSplex is not new, we're just now formalizing the term
- Examples of an **IMSplex** include ...
  - A set of IMS control regions at the V6 and/or V7 and/or V8 level without a CSL that are data sharing or message queue sharing
  - A set of IMS control regions at the V6 and/or V7 level (no CSL) that are data sharing or message queue sharing with V8 with a CSL
  - A set of IMS control regions at the V8 level with a CSL that are data sharing or message queue sharing
  - A single IMS control region at the V8 level with a CSL

IMS

The IMSplex is not a new concept, nor even a new term. But it has been formalized in IMS V8. It is what you have always thought it was - a group of address spaces (related to IMS, of course) that work together to perform a set of services for end users. Although the term is introduced formally in IMS V8, the IMSplex may consist of IMS V6 and IMS V7 IMS address spaces. And although we normally think of the IMSplex running on a parallel sysplex, it is technically not a requirement.

Common Service Layer (CSL)

The next step in IMS architectural evolution

- New *address spaces* built on Base Primitive Environment
  - *Structured Call Interface (SCI)*
    - IMSplex member registration
    - Communications between IMSplex members
  - *Operations Manager (OM)*
    - IMSplex-wide command entry and response
  - *Resource Manager (RM)*
    - Global resource and process management
    - VTAM terminal/user status recovery
- Enables new systems management *functions* in IMSplex
  - *Sysplex Terminal Management (STM)*
    - Uses SCI and RM
  - *Single point of control (SPOC) and user-provided automation (AOP)*
    - Uses SCI and OM
  - *Coordinated Online Change (Global Online Change)*
    - Uses SCI, OM, and RM

IMS

The existing IMS architecture did not provide a good means of providing these new systems management functions. So in IMS V8, the IMS architecture has taken another step in its evolution. The architecture is called the Common Service Layer, or CSL, and consists of three new address spaces.

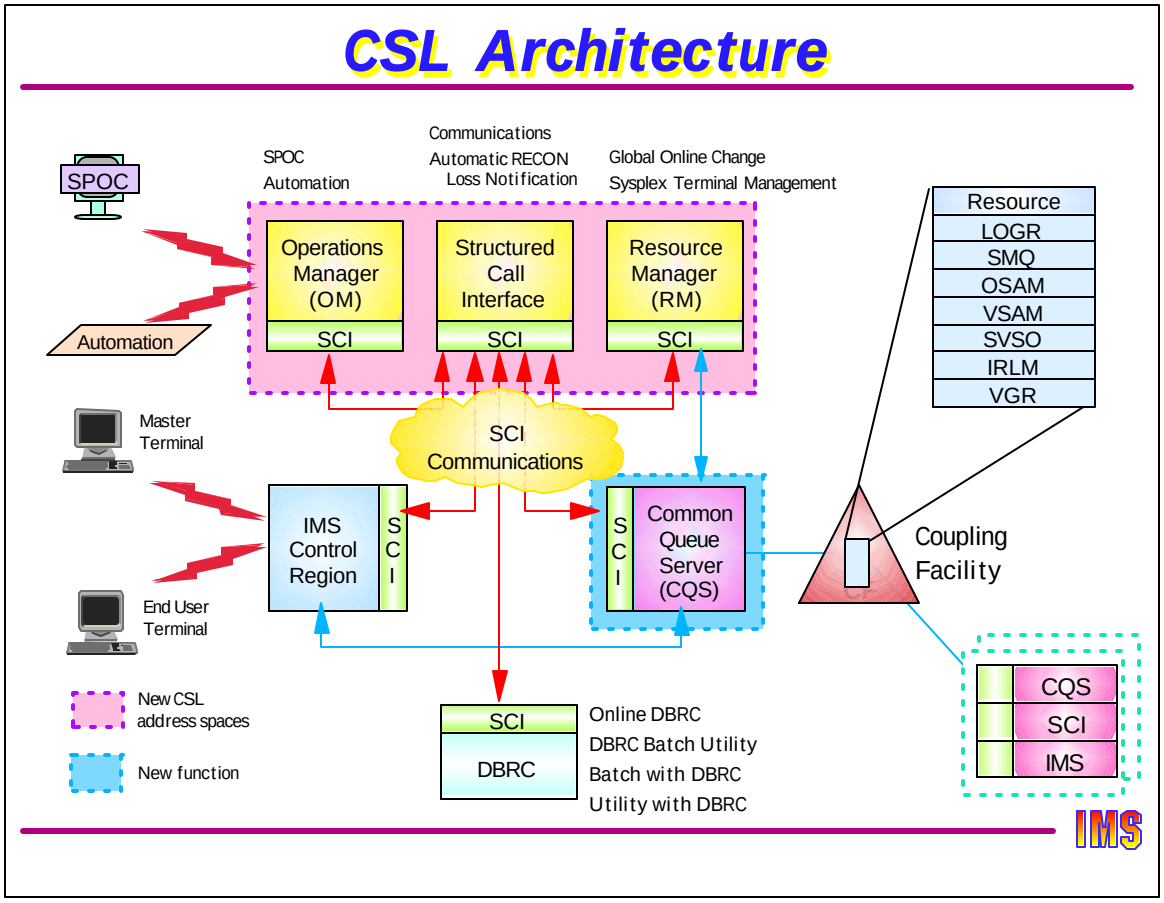
Structured Call Interface
- SCI allows for members of the IMSplex to "register" as members. Members include not only the IMS control regions, but may also include CQS, DBRC, SPOC and Automation programs, and of course the new CSL address spaces.
- SCI provides for communications between members. Registered members may uses SCI services to communicate with other members of the IMSplex.

Operations Manager
- OM provides the API for a SPOC or Automation Program to gain access to the IMSplex, enter commands, and receive responses.

Resource Manager
- RM provides the infrastructure for global resource and process management
- RM also supports terminal and user status recovery between IMSs.

These address spaces provide the infrastructure which enables members of the IMSplex to implement systems management functions. We will spend most of our time discussing these SM functions.
- Sysplex terminal management (STM)
- Single point of control (SPOC)
- Coordinated online change, also called global online change (G-OLC)

Each of these functions uses one of more of the CSL address space services.



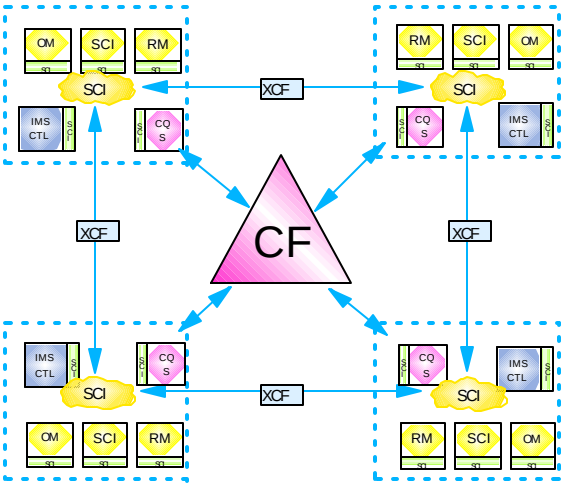
This diagram shows a minimum configuration for IMS in a CSL environment utilizing all functions. The three new address spaces are shown with the function they support written above them. For example, OM supports SPOC and Automation.

Sysplex terminal management requires a new structure in the coupling facility called the Resource Structure. CQS, the Common Queue Server, is used by the RM address space to access this structure. The same CQS can also be used to access the Shared Queues structures. If STM is not desired, there is no need for a Resource Structure. If a Resource Structure is used, then all RMs in the IMSplex must use the same one.

The interface between IMS and CQS for SQ and between RM and CQS for STM continues to use the CQS interface introduced with IMS V6 - it does not use SCI. All other communications uses SCI.

All forms of DBRC can register with SCI to get Automatic RECON Loss Notification (ARLN). This is a function which allows all DBRCs to be notified when a RECON is lost and the SPARE is used. More on this later.

IMSplex Configuration



Resource List Structure

LOGR List Structures

SMQ List Structures

OSAM Cache Structure

VSAM Cache Structure

Shared VSO Cache Structures

IRLM Lock Structure

VGR List Structure

- In an IMSplex
  - All members share the same CF structures
  - Intra-IMSplex communications is implemented by SCI using XCF across OS images

IMS

When there are multiple IMSs within the IMSplex, each IMS (actually each MVS image) must have a copy of the SCI address space. Only one copy of OM and RM is needed, although for availability and performance, it is recommended that several, or one per MVS image, be started. Wherever there is an IMS with shared queues, or an RM with a Resource Structure, there needs to be a CQS. Remember, one CQS can serve both the Shared Queue structures and the Resource structure.

When it is necessary to communicate between members of the IMSplex (for example, between IMS and RM), SCI is used. If the address spaces are not on the same MVS image, XCF communications services are used.

IMS V8 Highlights

CSL Components

- ★ Structured Call Interface
- ★ Operations Manager
- ★ Resource Manager
- ★ Resource Structure

IMS

This part deals with the CSL components - three address spaces and one CF structure. The first topic is the Structured Call Interface.

CSL Components (SCI)

SCI address space

- Provides for standardized intra-IMSplex *communications* between members of an IMSplex
- Provides *security authorization* for IMSplex membership
- Provides SCI *services to registered members*

Stuctured call interface services

- Used by SCI clients to
  - Register/deregister as member of IMSplex
  - Communicate with other members
- SCI client issues CSL macros to request SCI services
  - Documented in ***CSL Guide and Reference*** manual

SCI configuration

- One SCI address space is *required on each OS/390 or z/OS image* with IMSplex members

IMS

SCI is a "server" in the IMSplex, and provides several services the other members of the IMSplex. Unlike the other CSL address spaces, there must be a local SCI on every MVS image where an IMSplex member is running.

1. Accepts registration requests from other IMSplex members. When registering with SCI, the member also joins an XCF group, giving it access to XCF services, such as status monitoring and signaling services.

2. If the user defines the IMSplex resource in the RACF FACILITY class, then SCI will check each registration attempt by IMSplex members for authorization. Note that the user requires UPDATE access.

   ```
   RDEFINE FACILITY CSL.CSLPLX0 UACC(NONE)
   PERMIT CSL.CSLPLX0 CLASS(FACILITY) ID(PLXGRP0) ACCESS(UPDATE)
   ```

3. The main service provided by SCI to IMSplex members is communication services between members; for example, between IMS and the Operations Manager. All SCI services are provided using CSLSxxx macros. These macros are documented in the CSL Guide and Reference manual.

Structured Call Interface (SCI)

IMSplex address spaces register with SCI

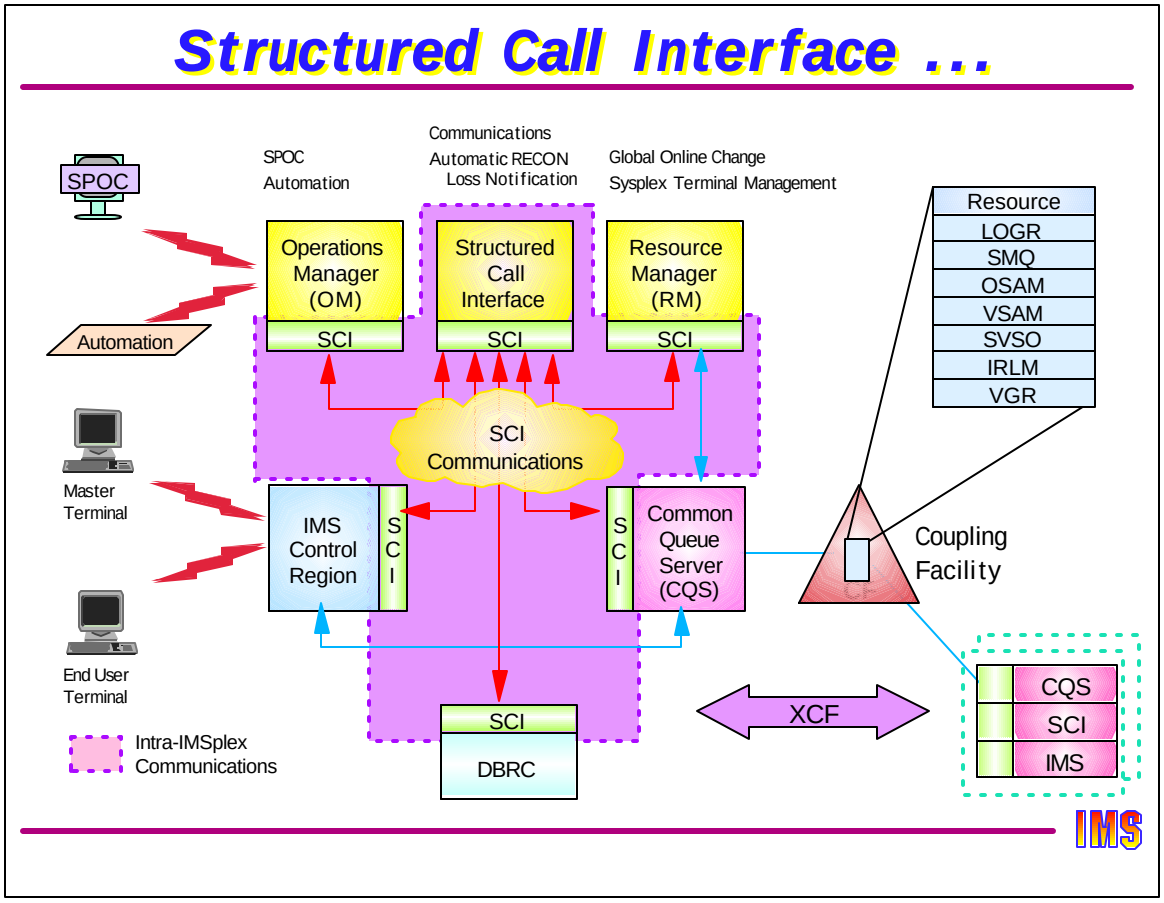
- CSL address spaces
  - Operations Manager (OM)
  - Resource Manager (RM)
- Common Queue Server (CQS)
- IMS
  - DB/DC, DBCTL, DCCTL, FDBR
- Automated Operator Programs (AOP)
- DBRC
  - Online DBRC address space
  - DBRC utility (DSPURX00)
  - Batch with DBRC=Y
  - DLI Utilities with DBRC=Y
- Other
  - CSL (SCI) interface is documented
  - May be accessed by user or vendor programs

Registrants may abend if SCI not available when required.

IMS

Other (non-SCI) address spaces which may register as part of the IMSplex include those shown on this graphic.
- The other two CSL address space types register with SCI.
- The Common Queue Server (CQS) has been enhanced in IMS V8 to support not only shared queues, but also the resource manager (discussed later).
- All IMS control region types are supported.
- Automated operator programs are those user- or vendor- or IBM-written programs which submit commands to IMS through the new operations manager interface (also discussed later).
- Each DBRC may register for the sole purpose of enabling Automatice RECON Loss Notification (discussed earlier in Part I and also later in this part).
- And, since the interface is documented, any other user- or vendor-written authorized program may join the IMSplex by registering with SCI.

It is important that SCI be started before any of the other members try to register. IMS control regions will wait, while other address spaces may abend if SCI is not available when the address space registers.



This shart just shows that SCI is involved with every member. Each individual member has an SCI interface which it invokes to register or communicate with other members. When communications is with a member on another MVS image, then XCF signalling services are used to send the message.

The top of the chart shows the major functions supported by the CSL address spaces. Each of this will be discussed shortly. Note also that there is a new Resource Structure in the coupling facility. This is optional - it is required only to enable Sysplex Terminal Management.

The interface between IMS and CQS was developed for IMS V6 shared queues and is also used in V8, even when CQS is used for access to the Resource Structure. SCI is not used for communications between IMS and CQS or between the Resource Manager and CQS.

IMS V8 Highlights

CSL Components

- ★ Structured Call Interface
- ★ Operations Manager
- ★ Resource Manager
- ★ Resource Structure

IMS

We continue with the next CSL address space - the Operations Manager.

CSL Components (OM)

Operations Manager (OM)

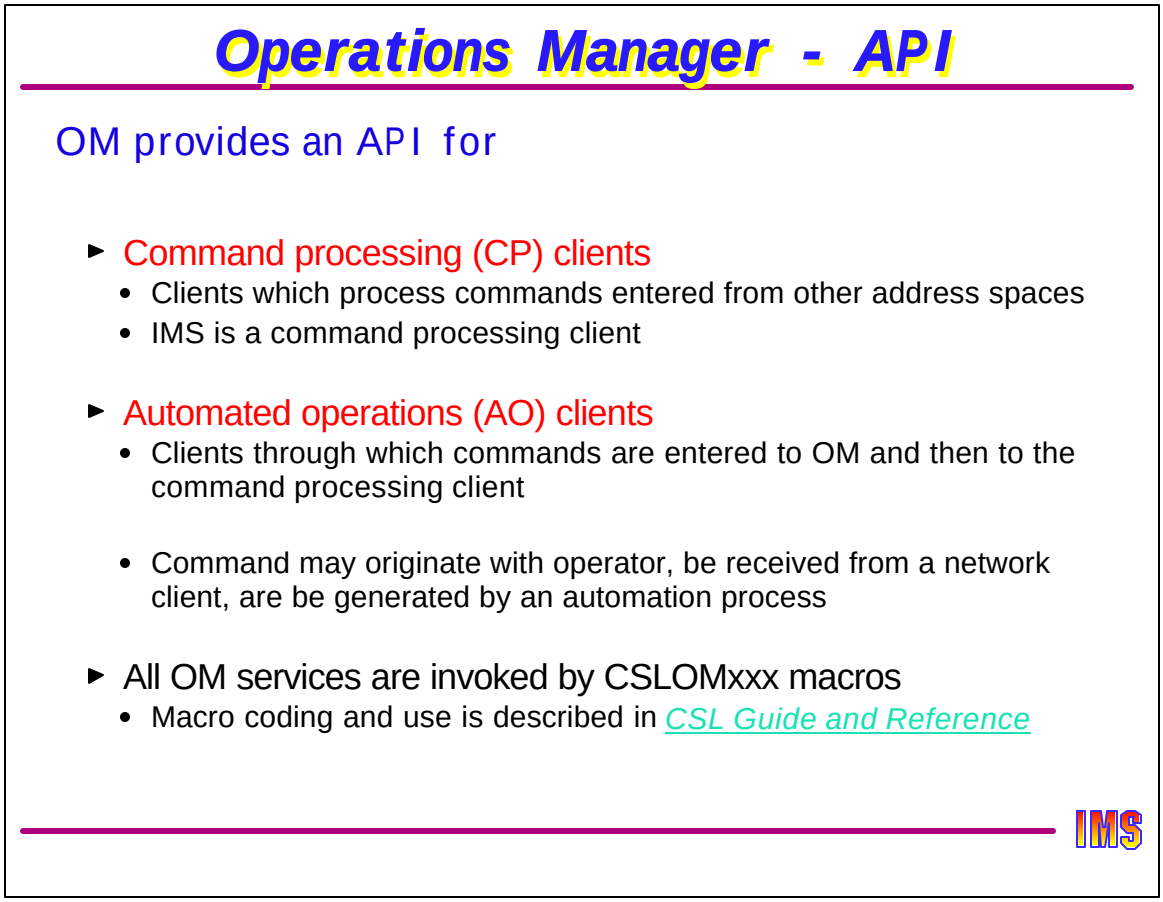
- Provides an API supporting *common point of command entry*
  - Focal point for operations management and automation
  - Command responses from multiple IMSs are consolidated
- Provides the following services to members and clients of an IMSplex
  - *Provide an API* for IMS commands submitted from outside IMS
    - Classic IMS commands (/cmd ...)
    - *New IMSplex commands (QRY, INIT, TERM, DEL, UPD)*
  - *Command registration* to support any command processing client
    - Clients tell OM which commands it can process
  - *Command security*
    - Perform authorization within OM - before sending to IMS
    - RACF or user-written command security exit
  - *Route commands* to IMSplex members registered for the command
  - *Consolidate command responses* from individual IMSplex members into a single response to present to the command originator

IMS

The Operations Manager (OM) is a server address space in the IMSplex. It acts as a common point of command entry into the IMSplex. That is, a command can be entered through the operations manager interface to any IMS in the IMSplex. The responses to those commands will be consolidated by OM and returned to the client (service requestor). OM, with its clients, can become the focal point for operations management and automation within the IMSplex [note that there may be multiple OM clients].

The major services provided by OM are listed:

- OM provides a "documented" API for IMS commands to be entered by an OM client to any or all IMSs in the IMSplex and to retrieve consolidated responses. The API is documented in the *CSL Command and Reference* manual. Any type of IMS command can be entered. This includes the classic commands (those starting with a '/') as well as the five new IMSplex commands (we'll discuss these later).
- OM can support any command process client that conforms to the API - not just IMS. In an IMSplex, the Resource Manager (RM) is also a command processing client, although there is only one command it can process.
- OM can provide authorization processing for IMS commands before that command is submitted to IMS. Authorization can be done using RACF, a user exit, or both. Or the user can opt not to perform command authorization in OM. There is a new parameter in IMS which indicates whether IMS itself should perform authorization processing on commands entered through OM. Presumably, if OM does command authorization, it would not be necessary for IMS to also do it.
- An OM client can instruct OM to route a command to any or all registered IMSs. IMS, during initialization, will inform OM what commands it wants to process by "registering" that command with OM. If a client enters an invalid command (for example, /DBX), then OM would reject it because it was not registered. It would not be passed on to IMS.
- And finally, OM will consolidate command responses before returning them to the client.



Two types of OM clients are supported.

1. Command processing (CP) clients are those to which commands entered through the OM interface are delivered for execution. IMS is obviously the most prominent CP client, although RM may also process one command (QUERY STRUCTURE). However, because this is an open API, any user or vendor could write its own command processing client. OM doesn't know or care, as long as the CP client joins the IMSplex and registers its commands with OM.
2. Automated operations (AO) clients are those OM clients who enter commands and retrieve the responses. OM forwards these command to a CP client, and then returns the reponses to the AO client. AO clients may be people sitting at a terminal, or they may be automation programs such as NetView EXECs.

The OM interface is documented in the *CSL Guide and Reference* as CSLOMxxx macros.

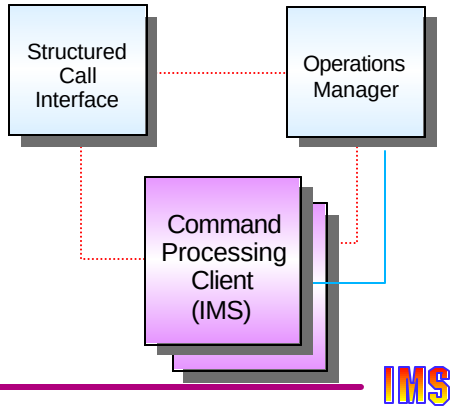
Command Processing Client

Command processing client

- OM client that processes commands
  - IMS and RM are command processing clients of OM
- CP client
  - Registers with SCI
    - Must be on same OS image
  - Registers with OM
    - Identifies commands that it can process
    - Any OM in IMSplex

- Processes commands received from OM
- Sends command response back to OM

- Deregisters from OM
- Deregisters from SCI



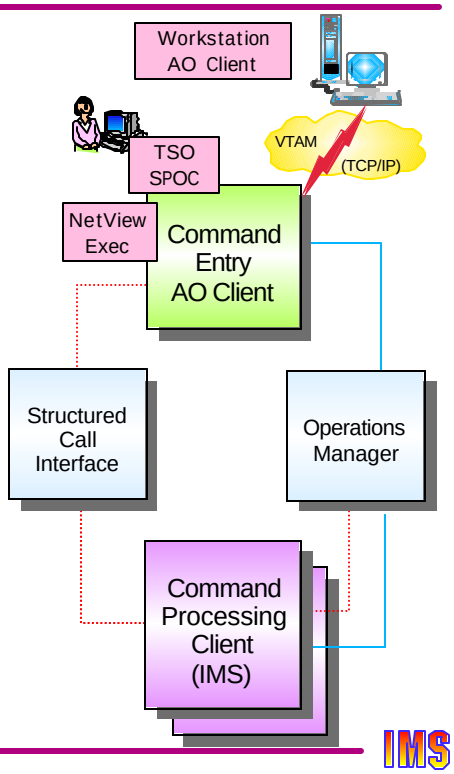
This chart shows where a CP client (IMS) fits in the IMSplex. IMS must register with SCI to become a member of the IMSplex and be eligible to receive commands from OM. OM must also register with SCI. IMS then registers commands with OM. These are commands which IMS is willing to accept from AO clients.

IMSs responsibility is to receive the command, process it, and respond to OM. IMS does not communicate directly with any AO client. A parameter in new proclib member DFSCGxxx (CMDSEC) determines whether or not IMS should perform command authorization for commands from OM. As mentioned before, command authorization can (and probably should be) performed by OM prior to sending the command to IMS.

AO Client

S/390 address space

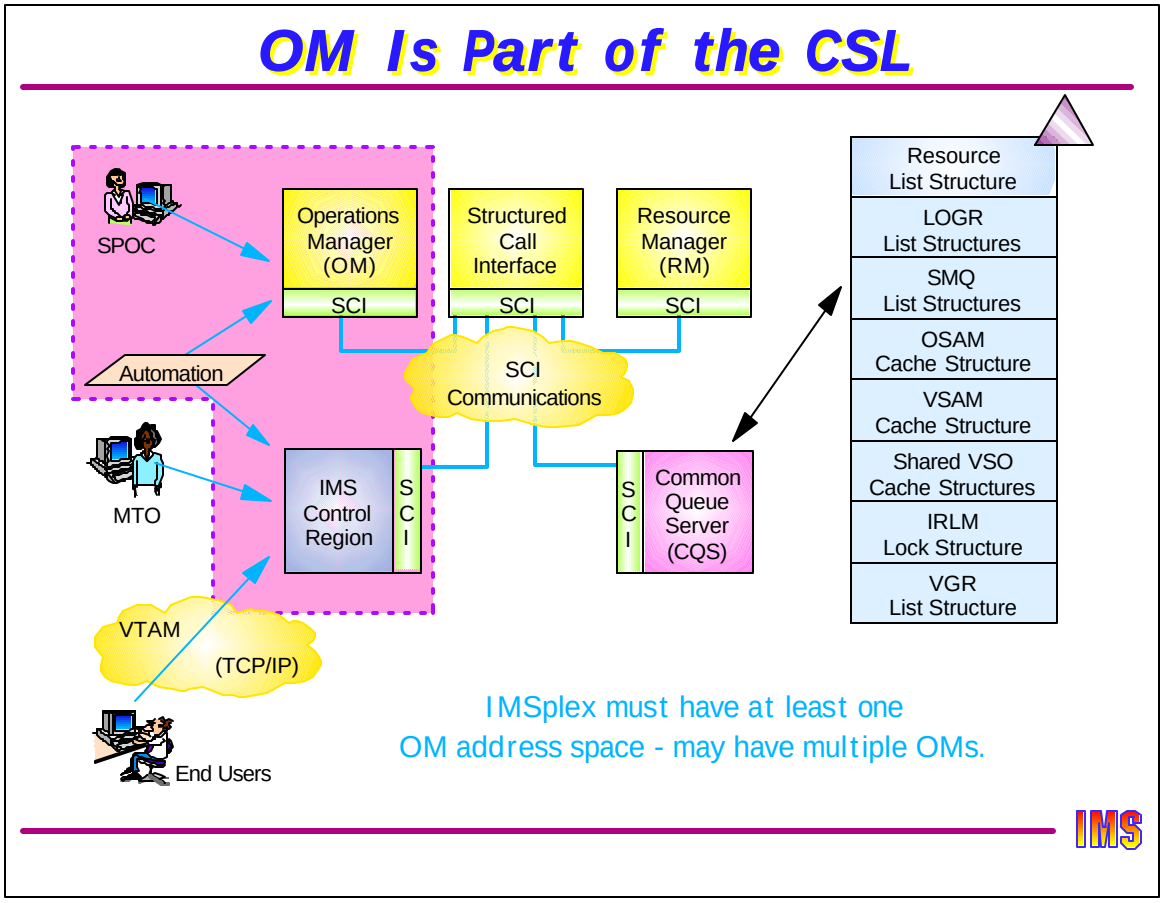
- Command originates from
  - Operator (TSO SPOC)
  - Automation (NetView Exec)
  - Network client (DB2 Control Center?)
- AO client
  - Registers with SCI
    - Accepts or creates command
    - Uses **CSLOMx** macro interface to
      - Send command to OM
      - Receive reply (in XML format)
    - Processes reply
      - Format for display
      - Forward to network client
  - Deregisters from SCI



An AO client is one which enters the command to OM. Human operators are not in direct contact with OM. For example, a TSO SPOC is delivered with IMS and has an ISPF application which registers with SCI and uses SCI services to communicate with OM. The human operator may be working at a workstation and not in direct session with an MVS AO client. In this case, the AO client would merely receive the command message from the workstation and pass it along to OM. A new Data Management Tool (DB2 Control Center) will be (or is) available which supports a GUI interface to DB2 and to through OM to IMS, giving that operator the ability to control both IMS and DB2 from the same terminal.

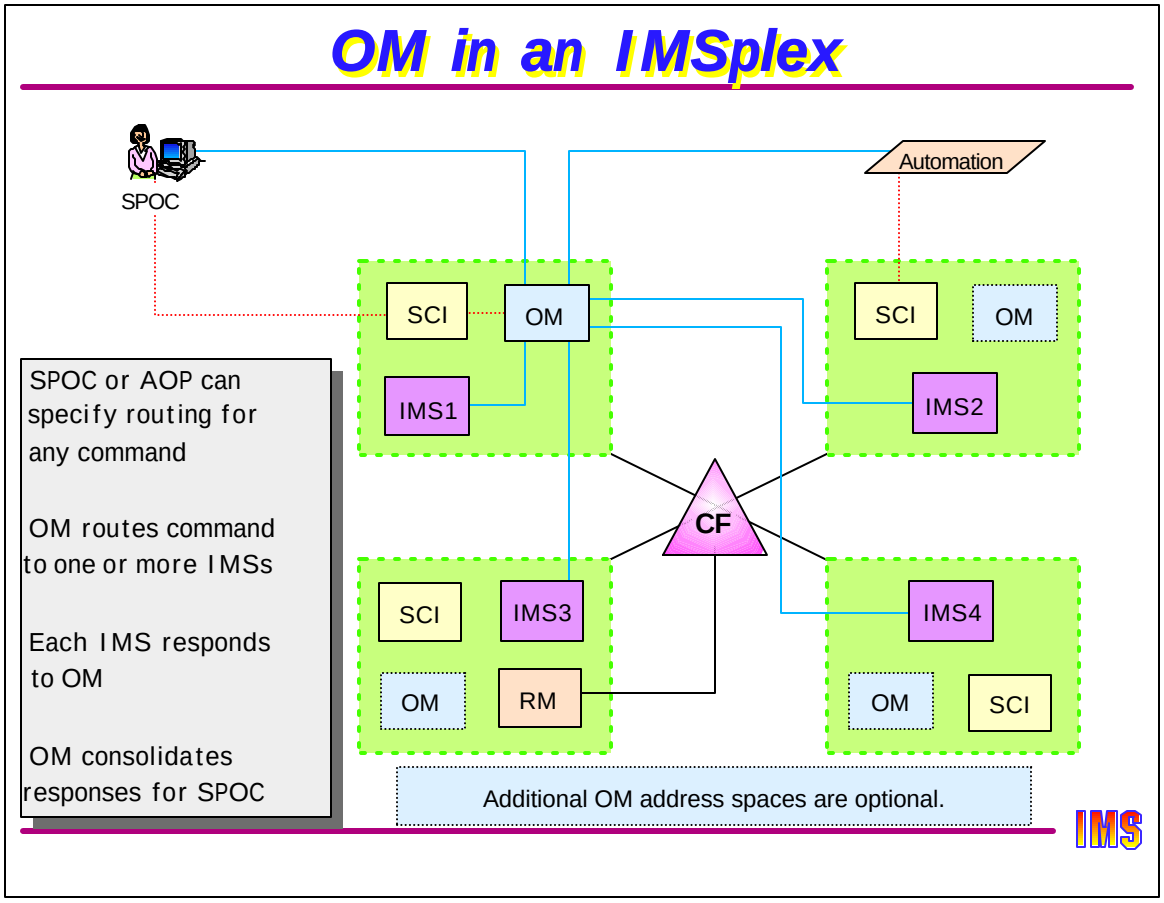
For human operators, it is the responsibility of the AO client to format the IMS response for display. IMS will return all responses to OM encapsulated in XML tags. This this makes it difficult for a human to read, the AO client should interpret the response and display it in a more readable format. The IMS V8 TSO SPOC does this.

Another type of AO client is an automation client. In this case, and automation program, such as a NetView EXEC, can join the IMSplex and enter commands to IMS. For example, if a message indicating that a transaction has been stopped is captured, a /START TRAN command could be issued to that IMS.



This chart shows the IMSplex components with interfaces to OM. Not shown is that the Resource Manager also registers with OM for one command (QUERY STRUCTURE). The SPOC (Single Point of Control) is used to describe any OM client where a human operator enters commands. The automation symbol indicates any program which enters commands without human intervention, such as a NetView EXEC.

Only one OM is required in the IMSplex. Additional OMs may be desired for availability.



This chart shows (at a high level) the flow of processing when an OM client enters a command to IMS and gets a response. The client must first register with SCI. SCI must be on the same MVS image as the client.

It then invokes OM services to enter commands and retrieve the responses. OM can be on any MVS image in the IMSplex. The request and response will be routed to an available OM using SCI communications services.

Each IMS will execute and respond to the command. OM will wait a specified amount of time (wait time is set by the client for each command) and the return all responses to the client. If an IMS does not respond, then an error message is returned for that IMS along with the responses for the other IMSs.

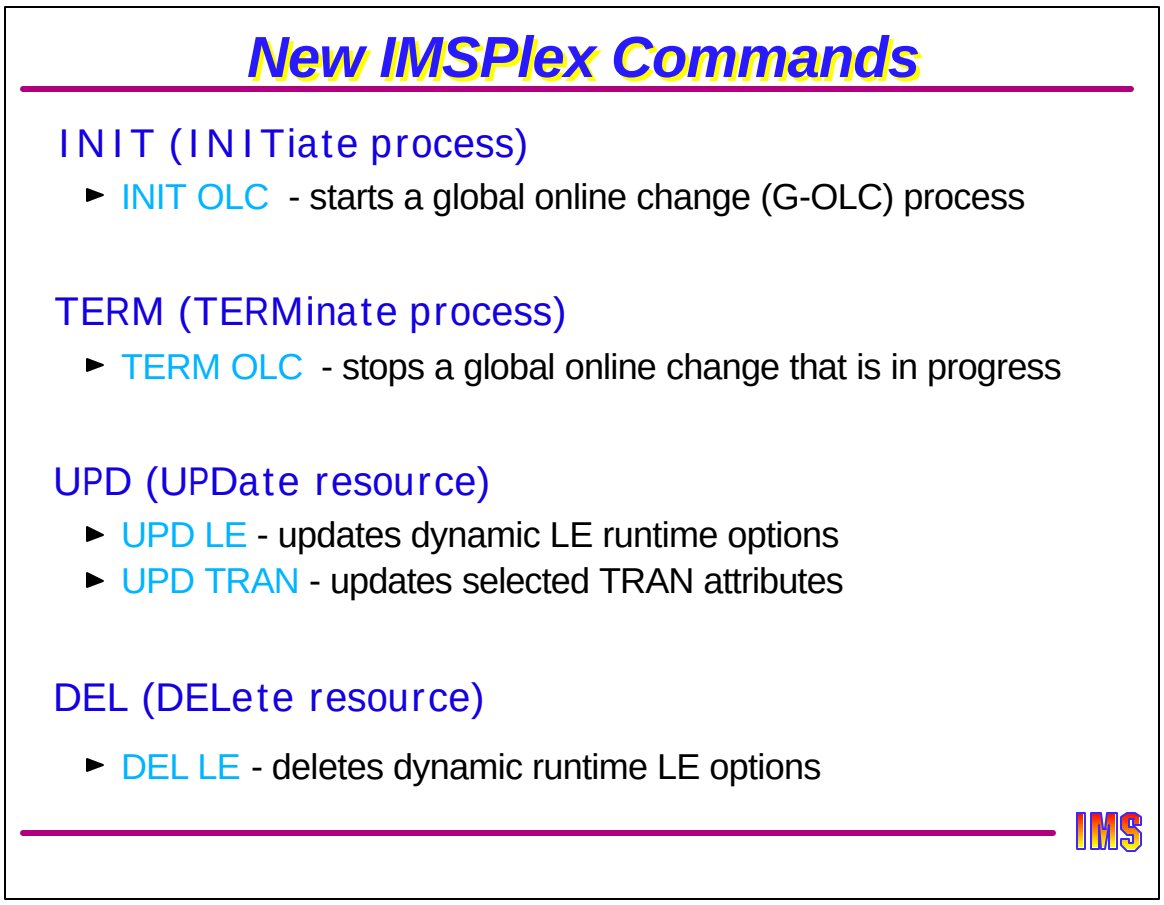
OM - Command Support

Commands

- New IMSPlex commands
- Classic IMS commands
- Command entry and response
- Command Security

IMS

The OM interface supports both the new IMSplex commands, and the classic IMS commands. Actually, OM supports any kind of command as long as someone registers it. OM's services include command entry and response, and command authorization.



This page shows four of the five new IMSplex commands. These commands can only be entered through the OM interface.

- INITiate - this command is used to initiate an IMSplex global process. In V8, the only global process in Global Online Change.
- TERMinate - this command is used to terminate a global process when something goes wrong. For global online change, this would be the equivalent of the /MODIFY ABORT command.
- UPDate - this command is used to update a resource. In V8, two resources may be updated - the dynamic LE runtime parameters, and some TRANSACTion attributes.
- DELete - this command is used to delete a resource. In V8, the only resource that can be deleted is the dynamic LE runtime parameters added earlier with the UPD LE command.

When IMS processes the UPDATE TRAN command it writes a new X'22' log record. IMS does not write a X'02' Log record because the condensed command buffer is not used

DFSLOG22 DSECT mapsthe X'22' log record.

New IMSPlex Commands ...

QRY (QueRY resource)

- QRY IMSPLEX - returns information about one or more members of the IMSplex
- QRY MEMBER - returns status and attributes of the IMS members in the IMSplex
- QRY LE - returns runtime LE options
- QRY OLC - returns OLC library and resource information
- QRY TRAN - returns TRAN info similar to /DIS TRAN
- QRY STRUCTURE - returns structure information of the RM resource structure

IMS

The QueRY command has several parameters. All of the QRY commands include a SHOW parameter to specify what fields you want returned. Or you can say SHOW(ALL).

- IMSPLEX displays information about the IMSplex itself. It includes for each member of the IMSplex: its MVS name, jobname, status, type (e.g., IMS, DBRC, OM, RM, AOP, ...) and subtype (e.g., DBDC, DCCTL, DBCTL, FDBR).
- MEMBER displays information about a specific member by member type. For V8. TYPE(IMS) is the only type supported. It shows the status some of the attributes of each member of that type, such as the global online change status for each IMS, or whether this IMS is using Shared Queues.
- LE displays the dynamic LE runtime parameters set by an earlier UPD LE command, including any filters and what the LERUNOPTS are.
- OLC displays information from the new OLCSTAT data set used for global online change. This information includes the OLCSTAT data set name, the suffixes for the active OLC libraries, the OLC modid, the type of the last OLC, and all IMS members who are currently in the global online change group and current with the active libraries.
- TRAN displays transaction attributes. The information displayed is equivalent to the /DIS TRAN command output, except that it can show the output from multiple IMSs. It also shows the global queue count if those IMSs are in a shared queues group.
- STRUCTURE displays information about the Resource Structure, including the number of entries and elements allocated and in use, and the Entry-to-Element ratio.

UPD / QRY TRAN Example

```
UPD TRAN NAME(PART) SCOPE(ALL) STOP(Q,SCHD)
     START(TRACE) SET(CLASS(4))

TRANCODE  MBRNAME   CC
PART      IMS1      0
PART      IMS2      0
PART      IMS3      0
```

Actual response is in XML format. Formatting for display is the responsibility of the command originator.

```
QRY TRAN NAME(PART) SHOW(CLASS,STATUS)

TRANCODE  MBRNAME   CC   CLS  STATUS
PART      IMS1      0    4    STOQ,STOSCHD,TRA
PART      IMS2      ...
```

IMS

This shows an example of an UPD TRAN command. SCOPE(ALL) tells OM to send this command to all active IMSs in the IMSplex. The other parameters are obvious. The response to the UPD command shows that the each of the IMSs executed that command.

Classic IMS Commands

Most classic IMS commands (/cmd ...) can be entered through OM API

- IMS commands specific to an input LTERM or NODE are not supported from OM
  - For example

```
/SIGN ON|OFF, /EXIT, /REL, /RCL, ...
```

If Resource Structure exists, some commands have global impact, for example

```
/STOP NODE ABC
```

- Node ABC is flagged as stopped in resource structure
- Node ABC cannot log on to any IMS in IMSplex

IMS

Classic IMS commands can also be entered through the OM interface. Each IMS will register with OM those commands which it will accept. In general, all IMS classic commands can be entered through the OM interface except those that apply to the inputting Lterm. For example, it would not make sense to enter the /SIGN ON command from the SPOC since this is not an IMS teminal. Other examples are shown.

Some IMS commands have global impact if Sysplex Terminal Management (discussed later) is active. For example, the /STOP NODE ABC command can stop the node resource in the resource structure. When this is done, that node is stopped on all IMSs in the IMSplex.

Classic IMS Commands ...

Some commands execute in every IMS where command sent

- Not aware of IMSplex

  ```
  /DIS TRAN TRX1 QCNT
  ```

  - Will execute in each IMS where command is routed
  - All will return same value (global queue count)

Most commands depend on several factors

- Command source, RM active with structure, affects significant status, resource exists on structure, resource owned by this IMS, resource owned by another IMS, display or update, ...

Command changes documented in *Command Reference* manual

- This is worth studying!!

IMS

Some clasic commands execute on every IMS which receives it. These are commands for resource status recovery is not supported by STM. For example, /DIS TRAN will execute on every IMS. Since part of the command output is the global queue count (in a shared queues environment), each IMS will query CQS, which queries the SQ structure, for this count.

Those commands displaying or changing the status of an STM-support resource (e.g., node, lterm, user), execution of the command depends on several factors. For example, if a NODE is logged on to IMS1, the /STOP NODE command can only be executed by IMS1. For display commands, only one IMS will return global information. For example, for /DIS LTERM XYZ QCNT, only one IMS will display the global queue count. Each IMS will display other local LTERM attributes. For these command, OM selects one of the IMSs as the MASTER for that command. Only the MASTER will perform global operations.

There are a significant number of changes to the way IMS commands work in this environment. They are all documented in the V8 Command Reference manual, and should be studied when migrating to IMS V8 with a CSL (even without a CSL, there are some changes).

Command Entry and Response

For commands entered through OM API

- AO client specifies
  - Command text
  - Routing information
    - Any or all IMSs
  - Wait time
    - How long should OM wait for IMS to respond?
- Target IMSs (one is selected as *master* by OM)
  - Execute command locally
  - Master IMS processes commands with global scope
  - Respond to OM in *XML format*
- OM will consolidate responses from all target IMSs
  - Sends consolidated response to AO client
  - Negative reply if any IMS does not respond within WAIT interval
- AO client
  - Formats XML response for viewing -or-
  - Sends XML response to network client

IMS

Summarizing,

- The AO client specifies the command text, routing information used by OM to route the command, and how long OM should wait for all IMSs to respond before returning to the AO client.
- OM selects one of the target IMSs (CP clients) as the MASTER for those commands supported by STM (nodes, lterms, users, msnames). Only the master processes any global portions of that command, such as displaying the global queue count for an Lterm. IMS returns all responses in XML format.
- OM will wait the specified amount of time for all IMSs to respond before responding to the AO client. If any IMS does not respond, an error message is returned to the AO client for that IMS.
- The AO client is then responsible for further processing of the response, such as formatting it for display, or sending it on to a network client.

OM Command Security

Depends on CMDSEC value in OM initialization Proclib member (DFSOIxxx)

- If CMDSEC=E(xit) or A(ll)
  - Define OM security exit in BPE User Exit List Proclib member

```
EXITDEF(TYPE=SECURITY,EXITS=(OMSCTYX0),COMP=OM)
```

  - Write security exit OMSCTYX (for example)
    - Linkedit into authorized library in OM steplib
- If CMDSEC = R(acf) or A(ll)
  - Define OM commands and security to RACF
    - QRY requires READ access
    - UPD, INIT, TRM, and DEL require UPDATE access

In (new) IMS Proclib member DFSCGxxx

- Should OM entered commands be authorized by IMS?

```
CMDSEC=R|E|A|N
```

IMS

Command authorization for command entered through the OM interface can be checked by OM, by IMS, or by both. In the OM initialization proclib member (DFSOIxxx), the CMDSEC= parameter tells OM whether to check command authorization, and if so, how. The choices are the same as for IMS - user RACF, use a user-written security exit, use both (RACF followed by the exit) or neither.

IF the exit is to be used, then the exit must be defined in the BPE User Exit List proclib member pointed to in the BPECFG proclib member. The user exit list entry should identify the type of exit as TYPE=SECURITY, and any name the user wants (e.g., OMSCTYX0).

If RACF is to be used, then the commands should be defined to RACF (example on next page). The QRY command requires READ access, the others require UPDATE access.

Then in the new IMS proclib member DFSCGxxx, used when IMS is running with CSL, the CMDSEC parameter identifies how IMS should handle security for commands entered through the OM interface. Presumable, if OM is performing security authorization, it can be skipped in IMS (CMDSEC=N).

OM Command Security ...

RACF definitions

- Define classic and IMSplex commands
- Permit users READ or UPDATE access to commands

```
RDEFINE  OPERCMDS IMS.CSLPLX0.UPD.TRAN UACC(NONE)
RDEFINE  OPERCMDS IMS.CSLPLX0.STO.DB UACC(NONE)
RDEFINE  OPERCMDS IMS.CSLPLX1.UPD.TRAN UACC(NONE)
RDEFINE  OPERCMDS IMS.*.QRY.* UACC(NONE)
RDEFINE  ....
PERMIT   IMS.CSLPLX0.UPD.TRAN CLASS(OPERCMDS)
         ID(MAKENA) ACCESS(UPDATE)
PERMIT   IMS.*.QRY.* CLASS(OPERCMDS)
         ID(LUKE) ACCESS(READ)
PERMIT   IMS.*.UPD.* CLASS(OPERCMDS)
         ID(NICK) ACCESS(UPDATE)
PERMIT   IMS.CSLPLX0.STO.DB CLASS(OPERCMDS)
         ID(ANDREA) ACCESS(UPDATE)
PERMIT   ....
```

IMS

Command security in RACF is defined in the OPERCMDS class. The format of the resource is:

IMS.imsplexname.command-verb.resource-type

Note that this is one level deeper than command security in IMS, which only authorizes commands to the command verb level.

Exploiting the OM API

TSO SPOC

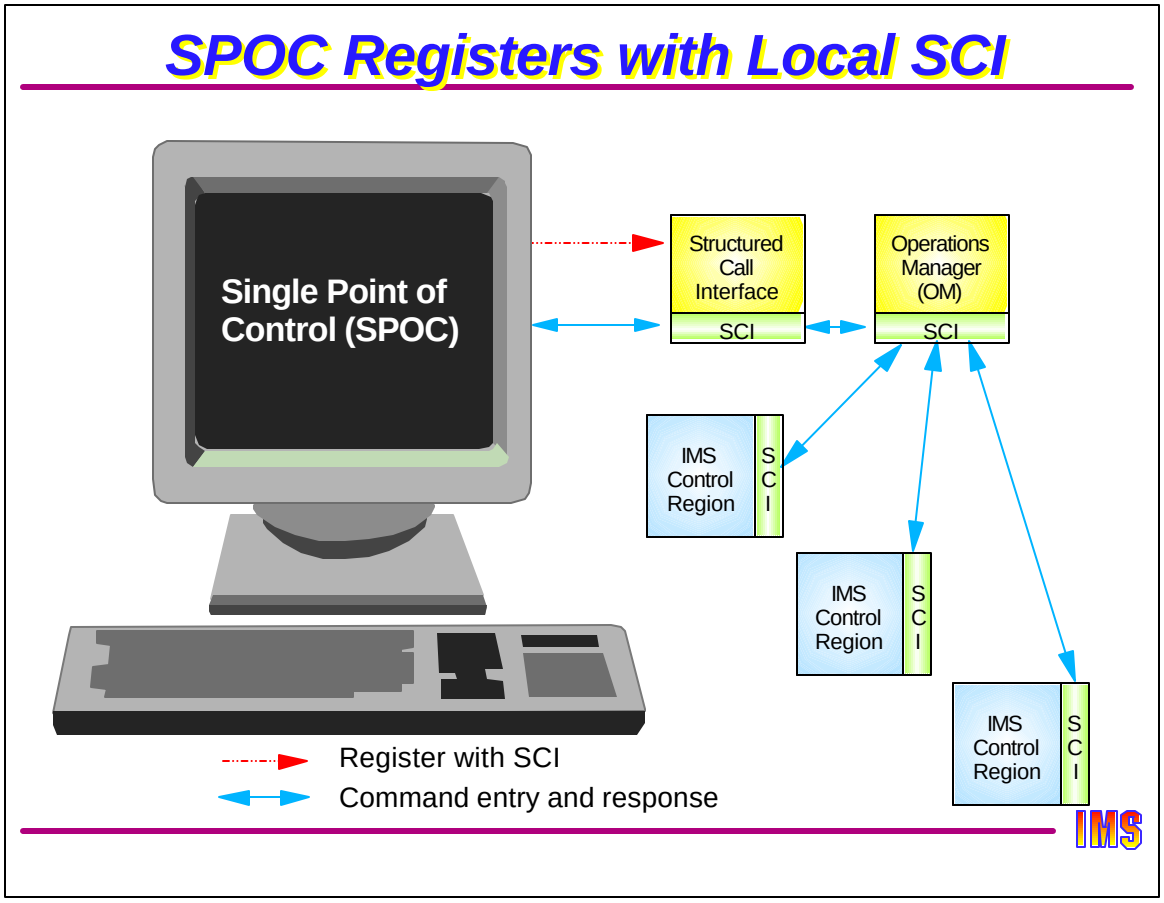
- ★ Provided with IMS V8

REXX EXEC

- ★ Sample exec using OM API

IMS

The following shows examples of the TSO SPOC delivered with IMS V8, and a sample REXX SPOC program (exec).



This section address the TSO SPOC delivered with IMS V8. Although it is called the Single Point of Control, it does not imply that there can be only a "single" SPOC. Nor does it imply that a single SPOC can completely control an IMSplex. There are many cases where commands must be entered into IMS as a result of an operator or automation program seeing a DFS message, or as the result of a scheduled time or time interval. The TSO SPOC does NOT see any IMS messages other than those returned in response to command entry by that SPOC.

Every SPOC must register with SCI. The SPOC does not have to be on the same MVS image as OM or IMS, but it does have to be on the same MVS image as SCI. After registering, the SPOC will use SCI to communicate with OM, send commands to IMS, and receive responses.

IMS Single Point Of Control (SPOC)

TSO SPOC

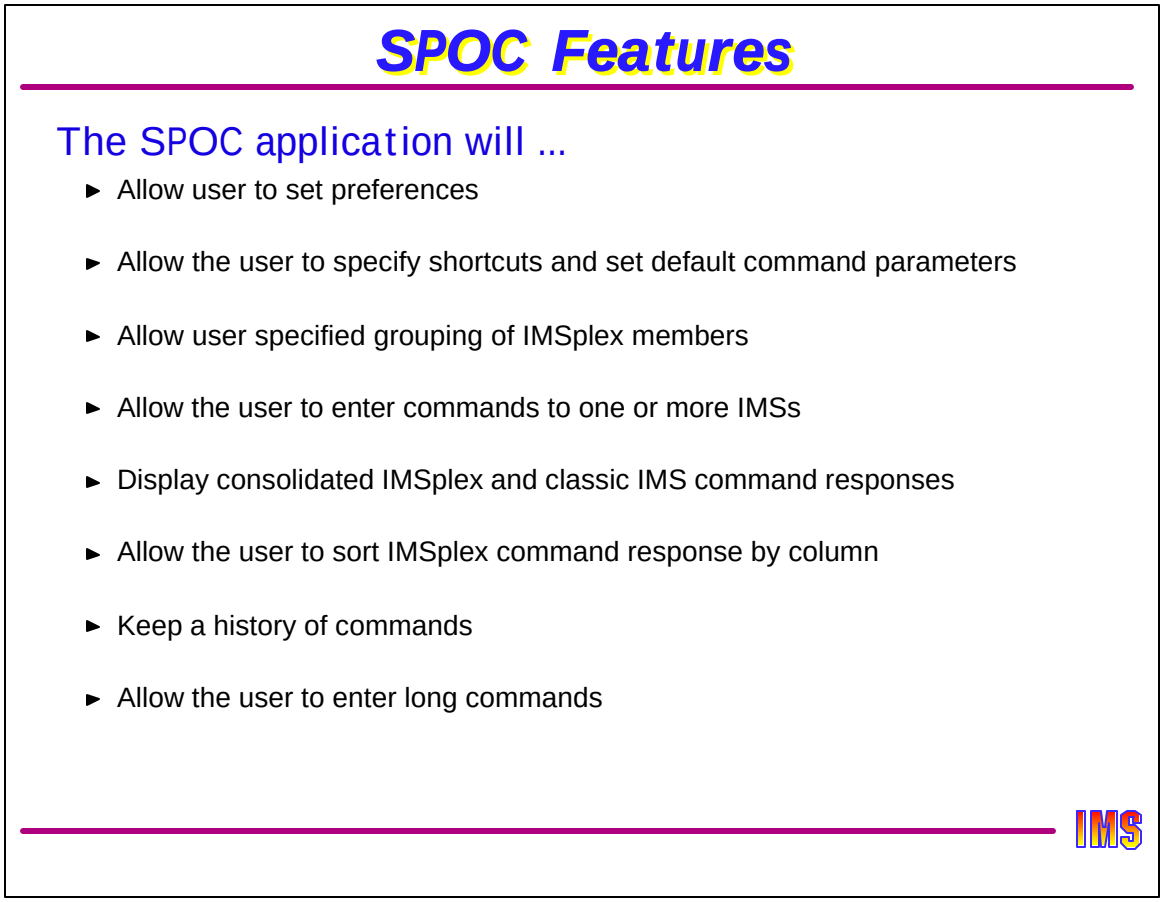
- Runs under MVS and TSO
  - ISPF Application (DFSSPOC)
- May or may not be on the same MVS as OM
  - Must be on same MVS as SCI
  - Uses SCI to communicate with OM
- Provides a terminal from which IMS commands may be entered by a person to one or more members of an IMSplex
- Formats command responses to be read by a person
  - OM response is encapsulated in XML
- OM provides security checking
  - TSO userid is used to determine RACF authorization

IMS

The IMS V8 TSO SPOC is an ISPF application invoked using the command DFSSPOC. Like other ISPF applications, a split screen can be used to start multiple DFSSPOC applications. As mentioned several times before, the TSO SPOC must be on the same MVS as SCI, but not OM.

The DFSSPOC application enters commands to IMS through OM using the CSLOMxxx interface and receives responses in XML format. It then formats these responses and displays them on the screen similarly to the way IMS would display them.

The TSO SPOC does not security checking on its own. This is done by OM and/or IMS using the TSO USERID for authorization.



The DFSSPOC application has many nice features to make it easy to use.

- The user can set "preferences" for command entry, such as the IMS or IMSs to route the command to, whether or not to allow "short cuts" and how to process them, a default OM wait time, etc. An example of the preference screen will be shown in a few pages.
- Shortcuts are abbreviated ways of entering commands. A command which is entered frequently can be abbreviated to just a few characters.
- IMSs can be defined as part of a group. For example, in an 8-way IMSplex, 4 could belong to one group and 4 to another. Command can then be routed to groups instead of individual IMSs.
- The user can override the default routing on any command input.
- DFSSPOC collects the consolidated responses from OM and displays them, identifying which IMS each response is from.
- Command responses can be sorted on column value.
- DFSSPOC will keep a short history of recently entered commands and the responses, allowing the user to go back and browse early command responses and even to edit and reenter the same command.
- The ISPF editor can be used to generate long command - commands perhaps with many database names.

Preferences

```
                IMS Single Point of Control Preferences
Command ===>
Select your options and press the Enter key.

Default IMSplex. . . . PLX0
Default routing. . . . IMS1 IMS2 IMS3 IMS4_____________________________

Wait interval. . . . . 2:45    (MM:SS)
Waiting preference . . 1  1.  Wait for command to complete.
                          2.  Do not wait for command response.

Command shortcuts. . . 1  1.  Use command shortcuts.
                          2.  Do not use command shortcuts.

Shortcut processing. . 2  1.  Merge explicit and default parameters.
                          2.  Explicit parameters override defaults.

Initial view . . . . . 1  1.  SPOC command panel.
                          2.  SPOC status list.
```

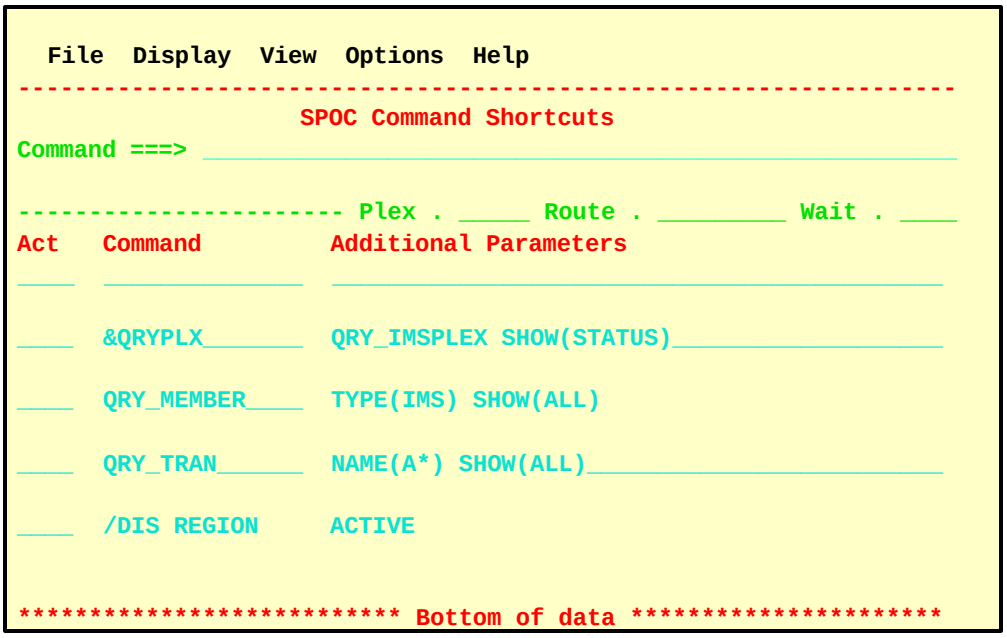
IMS

The preferences panel is used to set default values and to select some processing options.

- IMSPLEX - the name of the IMSplex
- Routing - names of command processors. Leave blank for all systems in the IMSplex.
- Wait Interval - the OM timeout value. OM will stop waiting and return a 'timed out' status.
- Wait Preference - do you want to wait for command response or look for it later in the command status panel
- Command Shortcuts - whether to use the short cuts or not.
- Shortcut Processing - merge or override parameters.
- Initial View - do you want to see the command entry panel first or the command status panel.

These values are saved in the ISPF profile dataset and are used the next time you use DFSSPOC.

Command Shortcuts

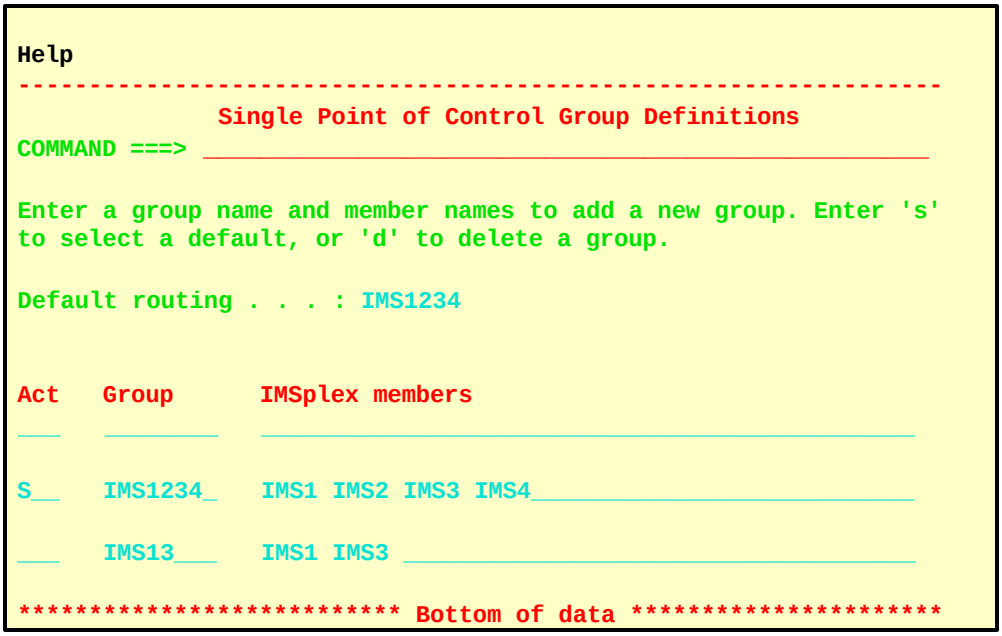


The shortcut table is a set of user defined commands. When a command is issued in SPOC and it matches an entry in the table, the additional parameters are either appended to, or replace, explicit command parameters (depending on user preferences).

If an ampersand (&) character is used as the first character, the entire command is replaced instead of being appended to.

Use the blank line to add new commands.
In the ACT(ion) column, use 'd' to delete an entry, use 'i' to issue the command.

Group Definitions



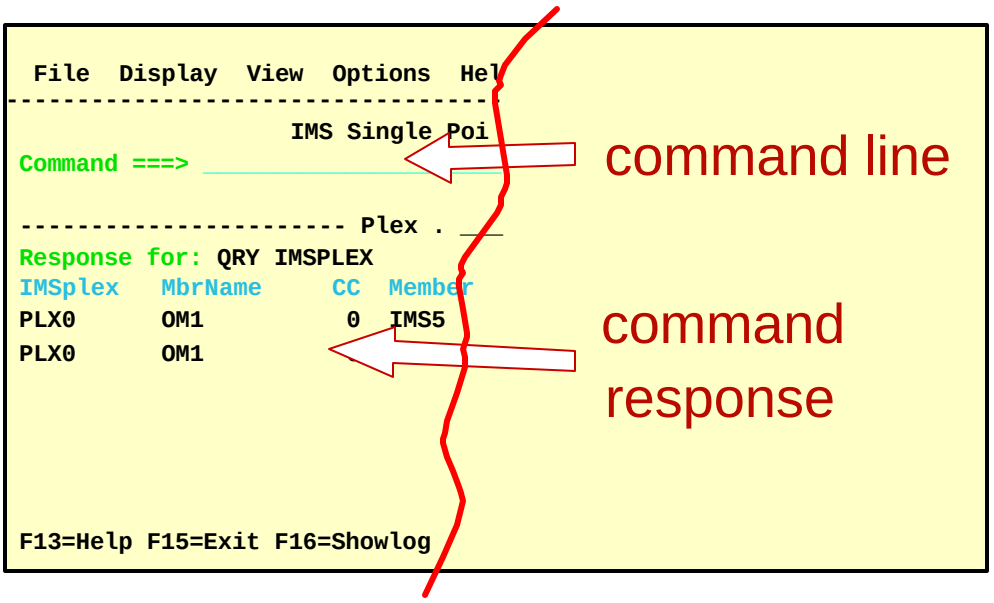
IMS

Use the group definition dialog to create user defined groupings of command processors. When a command is routed to this group, only the command processors listed will execute the command.

Use the blank line to create a new group. Use the 's' to select a default group, or 'd' to delete a group definition.

If you delete a group name that is in the default routing list, you get prompted to confirm the delete.

Command Entry Panel



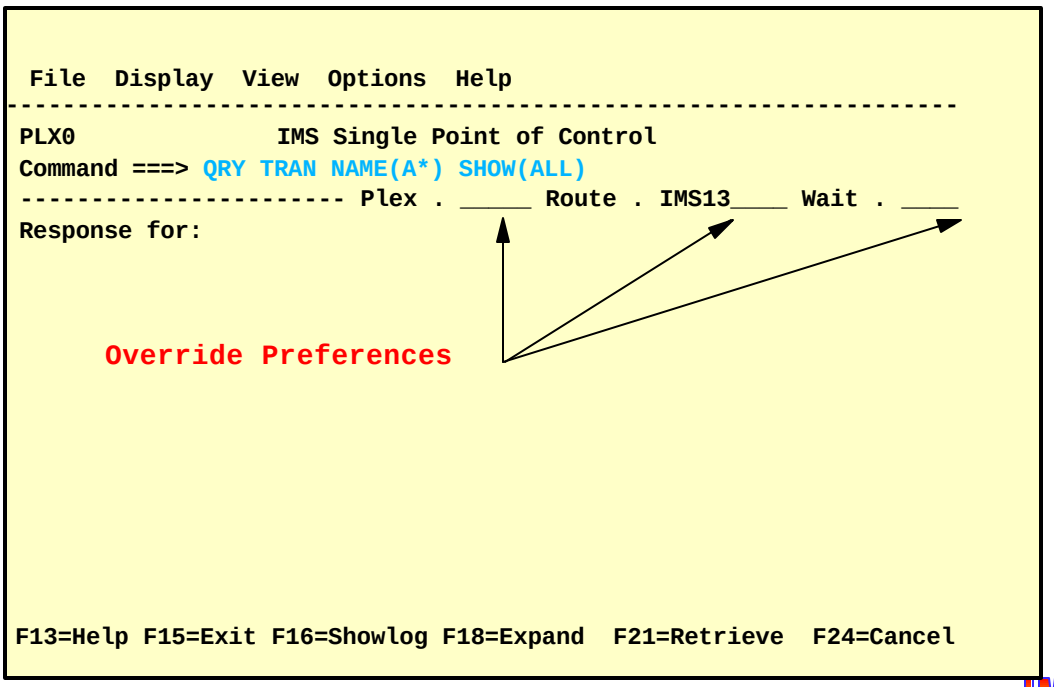
IMS

The basic operation is to enter a command in the command line and for the command response to be provided in the data area below the command line.

The command line is cleared in preparation for the next command: the command issued is shown just above the command response.

There are three short fields: plex, route, and wait. These are temporary overrides of the fields in the preferences panel. These values are discarded after you exit SPOC.

Command Entry



IMSPlex command and response pages with a lot of data will contain a scrolling indicator: "More: <-+> ".

- < more data to the left
- - more data below
- + more data above
- > more data to the right

Some IMSplex command responses will have key columns that do not scroll horizontally, so the user can have a point of reference. The columns are defined as keys by the XML. In the example, the TranCode and MbrName values will not scroll horizontally.

Command Response

```
  File  Display  View  Options  Help
-------------------------------------------------------------------
                  IMS Single Point of Control
Command ===> ____________________________________________________
---------------------- Plex . _____ Route . IMS13____ Wait . ____
Response for: QRY TRAN NAME(A*) SHOW(ALL)                More:   +>
Trancode MbrName    CC PSBname      QCnt LCls    LQCnt  LLCT LPLCT
ADDINV   IMS1        0                 0
ADDINV   IMS1        0 DFSSAM04            4        0     2 65535
ADDPART  IMS3        0                 0
ADDPART  IMS3        0 DFSSAM04            4        0     2 65535
AOBMP    IMS1        0                 0
AOBMP    IMS1        0 TS2IAOB0           23        0 65535 65535
etc.
```

Display formatted by SPOC from XML response.

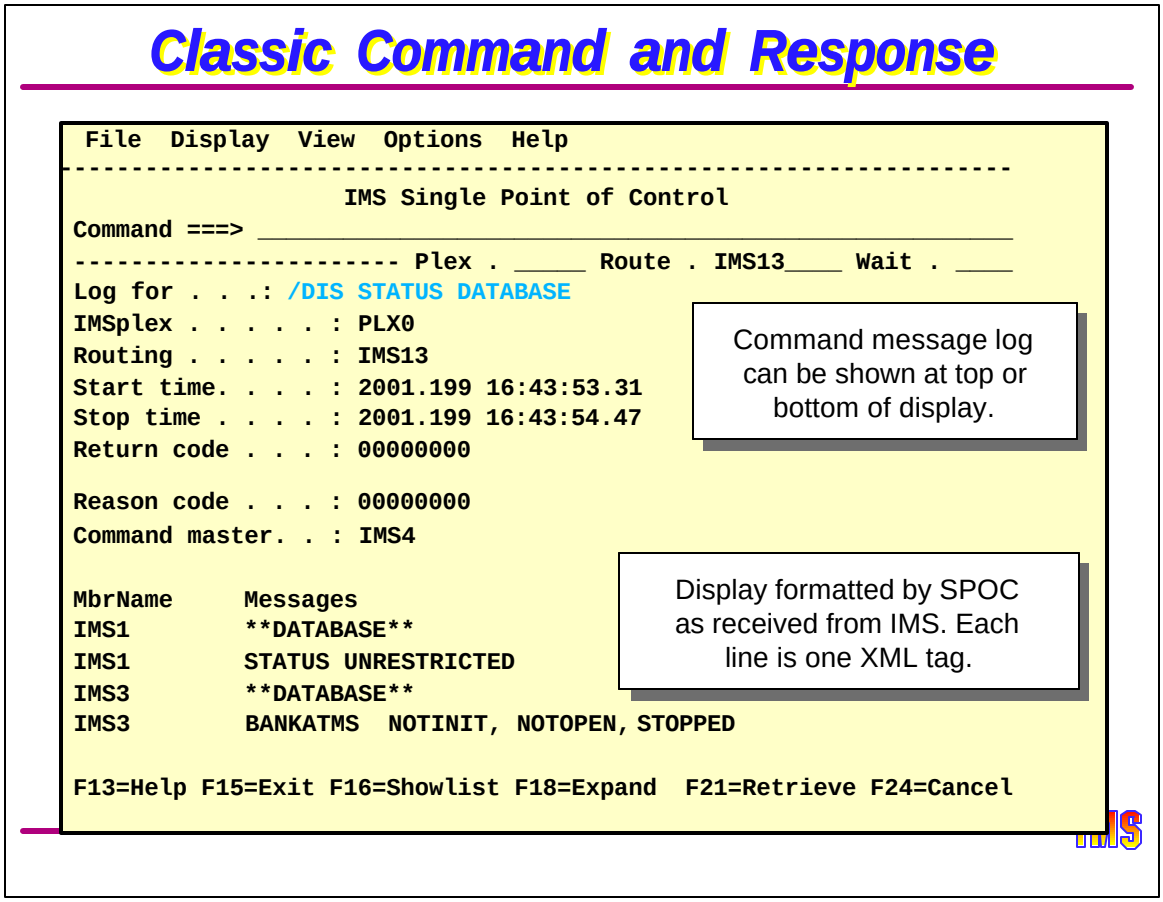
```
F13=Help F15=Exit F16=Showlog F18=Expand  F21=Retrieve  F24=Cancel
```

IMS

This screen is an example of a command response which does not fit a single screen. The + and > signs indicate more data below (+) and to the right (>).

Note that the command entry line has been cleared, but the command text is shown in the response section.

Classic Command and Response



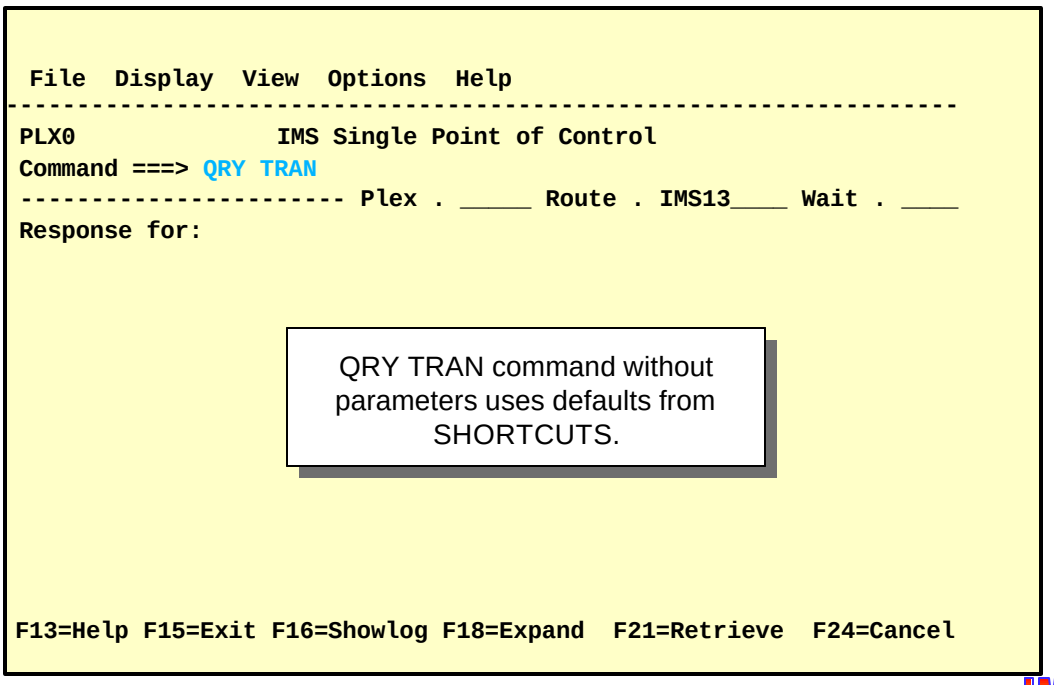
The command response for IMS commands is in a sequential format.

At top are some execution statistics and information.

Below are the messages produced by the IMS command. The text is in the same format as that of prior releases. The text is prefixed by the member name. Information from each member is grouped together.

IMSplex commands may have log information too if there were some messages to display. For example, one system may say 'no resources found' while other systems provide valid resource information. The 'no resources' indication would appear in the log.

Command Entry Using Shortcut



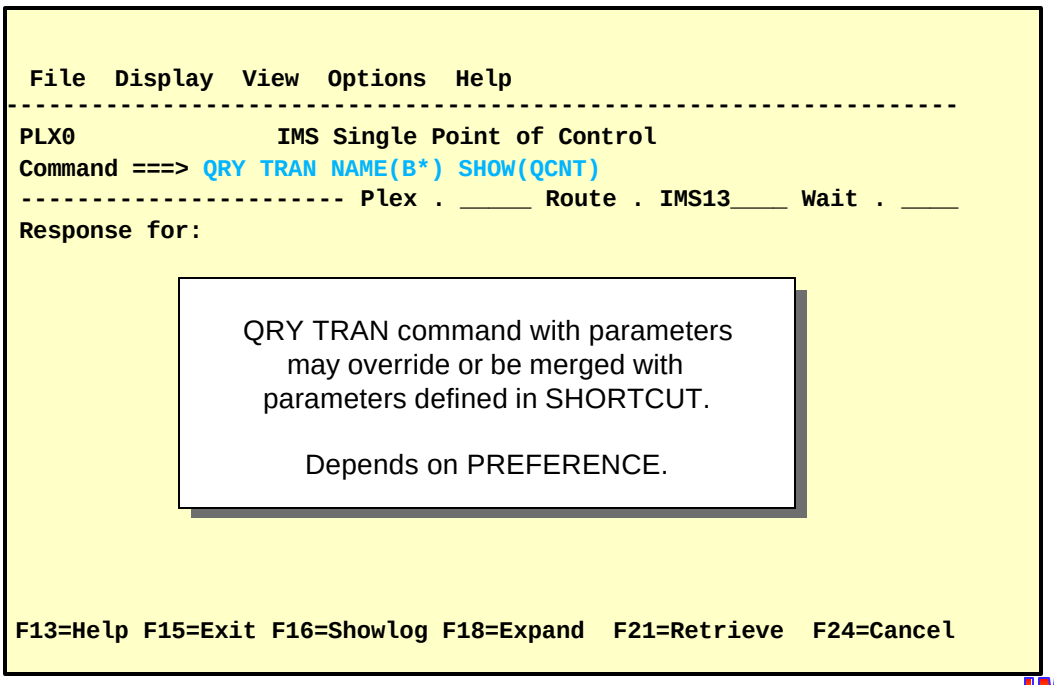
IMS

This is an example of a command entered using a shortcut. "QRY TRAN" was defined as a shortcut to be:

QRY TRAN NAME(A*) SHOW(ALL)

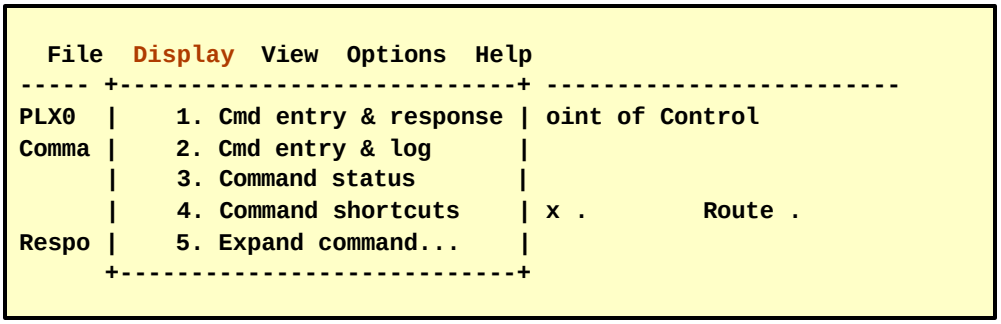
This is the command that would be sent to OM and IMS.

Command Entry Using Shortcut ...



When a short cut is used with additional parameters, the PREFERENCEs determine whether these parameters are to be added to, or replace, the short-cut parameters. Since in our earlier example, we opted to override defaults when explicit parameters are entered, the command shown on this screen would be the one entered to OM and IMS.

Action Bar



Cmd Entry and Log shows error or non-response messages
Command Status shows recent commands and responses
Expand Command is used for long commands

FILE includes SAVE AS and PRINT
VIEW includes FIND and SORT
OPTIONS includes setting PREFERENCES and GROUPS

IMS

You can display different command oriented panels using the action bar. Position the cursor on the 'ACTION' word in the action bar and press Enter.

Choose an option by typing the number or by positioning the cursor and pressing Enter. For example, 'Cmd status' pulls up recent commands and their responses.

REXX SPOC Application

Sample REXX interface to OM

- Run under TSO, Netview, ...
- May or may not be on the same MVS as OM
- Uses SCI to communicate with OM
- Saves command responses to a REXX "stem variable"

IMS

The REXX interface allows REXX programs to submit commands to OM and to retrieve the responses. The REXX programming language is frequently used to implement automation software. Programs written in REXX can run in NetView, foreground TSO, or batch TSO.

REXX SPOC Environment

CSLULXSB (TSO command)

- Sets up the REXX environment for the REXX SPOC API
- Establishes REXX function for retrieving response
  - CSLULGTS()
- Provides REXX variables for return code and reason code

Example:

- To send command to TSO environment

```
Address TSO "CSLULXSB"
```

IMS

The REXX host command environment is setup by the TSO command - CSLULXSB. This command sets up the REXX environment, establishes a REXX function for response retrieval, and defines REXX variables for return and reason codes.

Since this is a TSO command, the "Address TSO CSLULXSB" instruction sends the command to TSO to set up the environment.

An example of a REXX SPOC using these commands and functions will be shown shortly.

REXX SPOC Environment ...

IMSSPOC (TSO command)

- Establishes IMSSPOC environment
- Subcommands
  - IMS
    - Sets the name of the IMSplex
  - ROUTE
    - Sets the names of the IMSplex members
  - CART
    - Sets the name of the Command and Response Token
  - WAIT
    - Sets the OM timeout value in mm:ss

IMS

Environment variables provide "host command environment" in which IMS commands may be entered to one or more members of an IMSplex After it is issued as a TSO command, the IMSSPOC environment is available to the REXX program.

Host commands are typically quoted strings and passed directly to the host command processor.

Commands IMS, ROUTE, WAIT, CART, and END are supported and perform specific local functions. Anything else is passed to SCI as a command to be performed.

REXX SPOC Environment ...

CSLULGTS() (REXX function)

- Retrieves command response from OM and puts it into a REXX stem variable
- Parameters
  - Stem variable name
  - CART name
  - CSLULGTS() function timeout value seconds

IMS

CSLULGTS
This program is used by the REXX interface to get the response from OM. The XML returned by OM is parsed and each individual line is saved in a REXX stem variable. The name of the stem variable is specified by the user. There could be more than one active command response. Which XML is used depends on the CART specified by the caller.

Note that the timeout value specified here is how long the REXX program should wait for OM, not how long OM should wait for IMS. This should be a longer time than that specified on the WAIT subcommand.

REXX SPOC Example

```
 1  /* sample rexx exec */
 2  parse upper arg theIMScmd
 3  Address TSO 'CSLULXSB'
 4  if rc = 0 then do
 5     Address IMSSPOC
 6     "IMS   plx0"
 7     "ROUTE ims1"
 8     "CART  test12"
 9     "WAIT  3:00"
10     theIMScmd
11     results = cslulgts('resp.','test12',"3:15")
12     say 'imsrc='imsrc   'imsreason='imsreason
13     if resp.0 /= '' then do
14         say resp.0' lines of output'
15         do indx = 1 to resp.0
16           say resp.indx
17         end
18       end
19     "END"
20   End
```

IMS

This is an example of a simple REXX program which accepts an IMS command as an argument, submits that command to IMS through OM, retrieves the IMS responses, and displays them using the REXX 'say' command.

- IMS - sets the name of the IMSplex
- ROUTE - sets the name(s) of the command processors that will process the command.
- WAIT - sets the maximum timeout valued for OM to wait for a command response. If the time is reached, OM will return with a 'timed out' return code rather than command response information.
- CART - The Command and Response Token is a way to associate a name with the command that will be issued. CART is an 8-byte field containing a command and response token to be associated with the message. The command and response token is used to associate user information with a command and its command response.
- CSLULGTS - the REXX function to retrieve data associated with a specific token and assign responses to a REXX stem variable. In this example, it will wait 3:15 for the response before displaying the return and reason codes and, if there is a response, the response itself.
- END - cleans up control blocks.

OM Summary - Operations Manager

Operations Manager is part of Common Service Layer

- Joins IMSplex
  - Registers with SCI
  - Uses SCI to communicate with other IMSplex members
- One OM address space required per IMSplex
  - May have multiple OMs for availability and performance
  - Built on Base Primitive Environment (BPE)
- Provides services to IMSplex Command Processing (CP) and Automated Operations (AO) clients
  - API for submitting commands
  - Command registration for CP clients
  - Routes commands from AO clients to CP clients
  - Consolidates responses from CP clients and passes to AO client
  - Provides command security for classic and IMSplex commands

IMS

To summarize for operations management, the CSL Operations Manager address space becomes a member of the IMSplex by registering with SCI. At least one OM is required in each IMSplex. OM then provides services to AO clients for command entry to CP clients. OM can also perform command authorization for commands using either RACF, a user written security exit, or both.

OM Summary - AO Clients

TSO SPOC

- Allows the user to enter commands using ISPF panels
  - Set preferences
  - Create groups and shortcuts
- Displays IMSplex and IMS classic command responses
  - Process OM XML command response
  - Search or sort response
- Saves and retrieves previous commands and responses
  - See previous responses
  - Edit and reenter command

REXX SPOC

- REXX application to enter commands through OM API
  - Register with SCI
  - Submit command and process XML command response from OM
- Standalone batch program, TSO application, or NetView exec
  - IMS provides TSO commands and REXX function

IMS

An ISPF application called the TSO SPOC is delivered with IMS V8 which joins the IMSplex as an AO client and then sends commands to and retrieves and displays responses from one or more IMSs in the IMSplex. Since responses are in XML format, the TSO SPOC must format the message for display.

The user, or vendor, may write any program to join the IMSplex and act as an AO client. The interface is documented in the *CSL Guide and Reference*.

IMS V8 Highlights

CSL Components

- Structured Call Interface
- Operations Manager
- Resource Manager
- Resource Structure

IMS

This section addresses the Resource Manager and the Resource Structure.

CSL Components (RM)

Resource Manager (RM)

- *Provides infrastructure* for managing global resources and IMSplex-wide processes
- *Maintains global resource information* for clients using a Resource Structure in the Coupling Facility
  - IMSplex global and local member information
  - Resource names and types
  - Terminal and user status
  - Global process status
- Resource structure is optional
  - If resource structure not defined
    - Only one RM per IMSplex
    - Sysplex terminal management not enabled

IMS

The Resource Manager address space provides the infrastructure and services to allow its clients (IMS) to manage global resources (e.g., terminal and user resources) and global processes (e.g., global online change).

To provide the "global resource management" services, a Resource Structure is required. In this structure, RM maintains information about the IMSplex and its members, as well as information about the Sysplex Terminal resources. It may also contain information about global processes, however, the structure is not required for global online change.

While the Resource Structure is optional in this environment, if it has not been define, Sysplex Terminal Management (IMS's expoitation of the global resource management services of RM) is not enabled. We'll discuss STM in more detail a bit later. Also, if there is no Resource Structure, only a single RM address space is allowed in the IMSplex.

CSL Components (RM) ...

RM clients

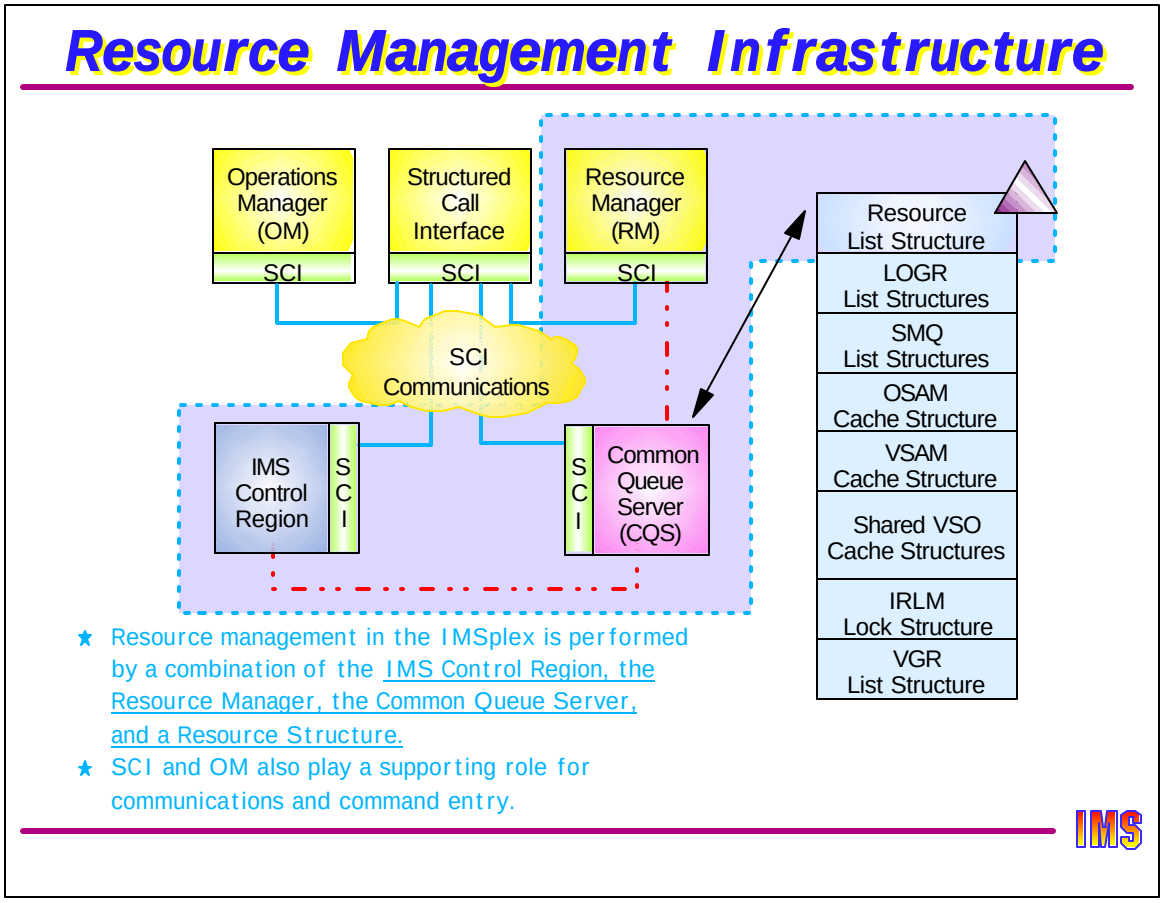
- IMS control region
  - To provide *sysplex terminal management* functions
    - Resource type consistency across IMSplex
    - Resource name uniqueness across IMSplex
    - Restore terminal and user status when switching IMSs (e.g. restore conversation on new IMS after an IMS failure)
  - To *coordinate global online change*
    - With OM and IMS, coordinates OLC across IMSplex
  - To expand functionality of IMS exits
    - *Global callable services* of IMSplex-wide status
- Vendors?

IMS

RM, as of IMS V8, has only one client type - an IMS control region. The control regions are responsible for exploiting the RM services to provide several systems management functions:

- Sysplex Terminal Management (STM) consists of three subfunctions
  1) Resource type consistency - guaranteeing that the same resource name is not used to define different resource types by different IMS systems
  2) Resource name uniqueness - guaranteeing that the same resource is not active on multiple IMSs at the same time
  3) Resource status recovery - restoring sysplex terminal and user status following a logoff/logon, signoff/signon, or IMS restart.
- Global Online Change (G-OLC) coordinates the online change process across all active IMSs in the IMSplex. This function requires the Operations Manager and an AOP 9e.g., the TSO SPOC) to submit the commands to initiate OLC.
- Global Callable Services allows an IMS exit using FIND or SCAN callable services to retrieve global status when a node, lterm, or user is not active locally but does have some global status on the Resource Structure.

Since the RM interface is externalized, vendors or users may write RM clients.



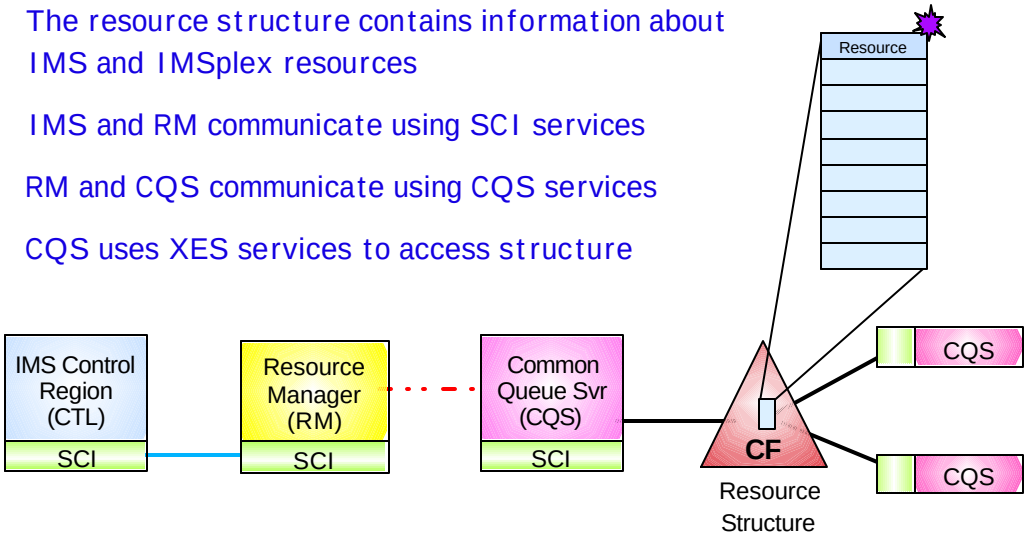
The complete architecture of Resource Management in the IMSplex is shown here. Full service requires an IMS control region, a Resource Manager, the Common Queue Server, and a Resource Structure. If no resource structure is defined, then CQS is not required (unless, of course, shared queues is active).

SCI provides communications services between IMS and RM. IMS does not communicate directly with CQS for Resource Structure access. Note (again) that communications between IMS and CQS for shared queues, and between RM and CQS is not through SCI, but through the CQS interface introduced in IMS V6.

OM is required to enter G-OLC commands.

RM Infrastucture ...

- IMS uses RM to manage resource information
- RM uses CQS to manage resource structure
- The resource structure contains information about IMS and IMSplex resources
- IMS and RM communicate using SCI services
- RM and CQS communicate using CQS services
- CQS uses XES services to access structure



IMS

This chart shows the all components used for resource management, and how they interact.

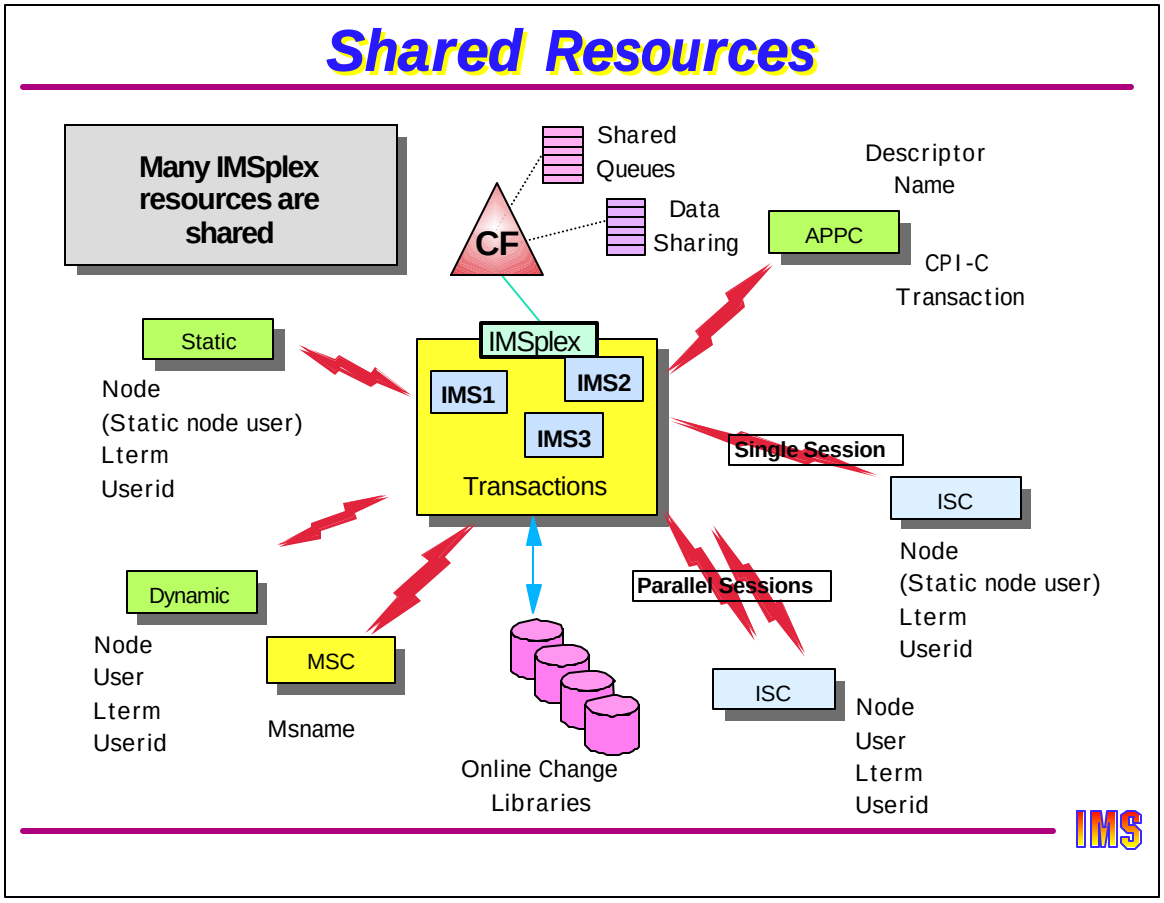
CSL Highlights

Sysplex Terminal Management

- ★ Resource type consistency
- ★ Resouce name uniqueness
- ★ Resource status recovery

IMS

This section addrss Sysplex Terminal Management and its three subfunctions. All of these require a resource structure and shared queues.



This diagram shows an IMSplex with all of the resources that are managed to some degree by STM. The online change libraries are shown too. Even though they are not part of STM, they are a sysplex resource. Management of the G-OLC process will be described later.

All of the other resources (other than OLC data sets) are related to the terminal network that has access to the IMSplex. Each of these resources has at least one name associated with it, and usually several names. For example, a VTAM Node has a node name and an lterm name. If it is an ETO node, it also has a user name and a userid. Static nodes may also have userids, although they are not required. There is no user name associated with a static node, but for consistency in saving information in the resource structure, a "static node user" has been invented. The "static node user name" is the same as the node name.

For APPC clients, resources are not defined to IMS - they are defined to APPC/MVS. There are only two "names" that STM keeps track of for APPC sessions: the LU Descriptor Name defined in DFS62DTx, and a CPI-C transaction code defined in the TP_Profile.

Although technically not a terminal resource, IMS also keeps track of transaction codes in the resource structure, but only for purposes of providing resource type consistency - that is, to make sure a transaction code and an lterm (for example) do not have the same name on different IMSs.

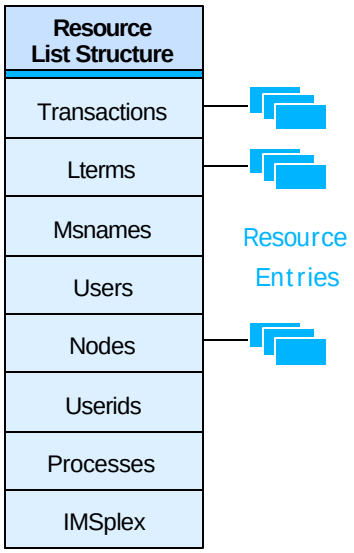
Resource Structure

Resource structure contains global resource information for uniquely named resources

- Transactions
- Nodes, lterms, users, userids, msnames, APPC descriptors
- Global processes
- IMSplex local and global information

Resource structure is optional

- Defined in CFRM Policy (if used)
- If no resource structure defined
  - Terminal/user resource status saved locally; cannot be shared
  - Sysplex terminal management disabled
- Resource structure not required for global online change



IMS

The Resource Structure must be defined in the CFRM Policy, and also defined in RM and CQS proclib members. When CQS connects to the structure, it defines it as a LIST structure. Information about the resources being managed by STM are list entries queued off list headers representing the resource type. For example, each active Lterm has a list entry queued off one of the 9 Lterm list headers; each active Node has an entry queued off one of the 9 Node list header; etc. The list header types (representing resource types) shown in the visual do not represent the complete set of list headers or resource types, but are sufficient to demonstrate how STM works.

If a resource structure is not defined, then STM is not enabled and all information about sysplex terminals is maintained only in IMS's local control blocks and log records. The information is not shared. This is as it has been in all prior releases. Global online change, however, is supported even without a resource structure. The structure merely helps G-OLC recover more gracefully from some types of errors.

Sysplex Terminal Management

Enables improved systems management in an IMSplex by sharing resource status information

- Applies to ***VTAM*** terminal and user resources
  - BTAM and OTMA resources not supported

Global resource sharing requires the resource manager, a resource structure, and shared queues

- Resource names and status saved in structure
- Shared by all IMSs in IMSplex

Without a resource structure, user can opt for ...

- Local status recovery
  - Same as pre-V8
- No status recovery
  - New function in V8

IMS

Sysplex terminal management (STM) applies only to VTAM terminals and users. BTAM and OTMA resources are not supported. APPC is supported only minimally (resource type consistency - described later).

Global resource sharing requires the resource manager address space, a resource structure, and the IMSs must be running with shared queues enabled.

Without a structure, the user can opt for local status recovery (same as previous versions) or no status recovery (new in V8).

Sysplex Terminal Management ...

Sysplex terminal management objectives

- Enforce global *resource type consistency*
  - Prevent naming inconsistencies between IMSs
- Enforce global *resource name uniqueness*
  - Prevent multiple logon / signon within the IMSplex
- Enable global *terminal and user resource status recovery*
  - Resume significant status on another IMS after failure
    - Conversation, fast path response, STSN sequence numbers
    - Command status (e.g., stopped, assigned, ...)
  - Reduce need for IMS-managed VGR affinity
- Enable *global callable services*
  - User exits can access terminal and user information across IMSplex

IMS

STM has four objectives, as shown.

- Resource type consistency - prevents naming inconsistencies across the IMSplex for resources which are IMS shared queue destinations. These include transaction codes, lterm names, msnames, and APPC descriptor names (which are treated similarly to Lterm names when used in a CHNG call).
- Resource name uniqueness - prevents the same resource "name" from being active on more than one IMS at a time. For example, a single session node will be preventing from logging on to more than one IMS at a time. An Lterm name can be active only only one IMS at a time. This function applies to nodes, lterms, users, userids (unless SGN=M).
- Resource Status Recovery - allows a user with some significant status to terminate a session (normally or abnormally) on one IMS and continue that status on another. For example, if a user is in conversation on IMS1 when IMS1 fails, that user can log on to IMS2 and continue the conversation. There are several types of significant status that a terminal/user can have. This will be discussed a bit later. Also, by keeping status information in the resource structure, the need for VTMA Generic Resource affinities is reduced. For example, if the structure contains information about a conversation, then it is not necessary to retain VGR affinity and force that user to log back on to the same IMS.
- Global Callable Services - allow TM-related user exits using the SCAN and FIND functions of callable services for nodes, lterms, users, and userids to determine if a resource is active on any other IMS in the IMSplex if it is not active on the local IMS.

Resource Type Consistency

Prevents the same *resource name* from being used for different message destination *resource types*

- For example, don't allow IMS1 to define transaction PRSNL and IMS2 to define Lterm PRSNL

Applies to *message destinations*

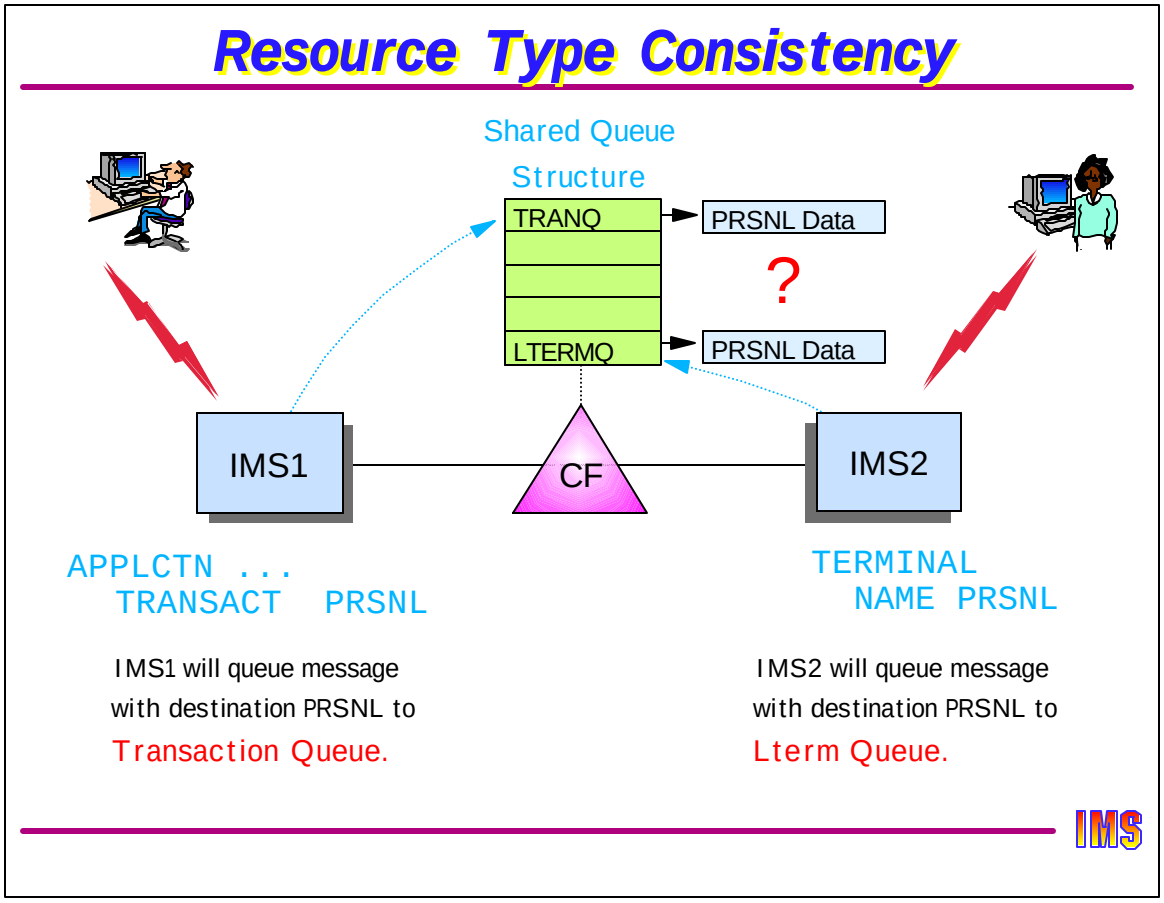
- *Transaction names - static, dynamic, and CPI-C*
- *Lterm names*
- *Msnames*
- *APPC descriptor (lterm) names*

These are all Shared Queue destination names. !

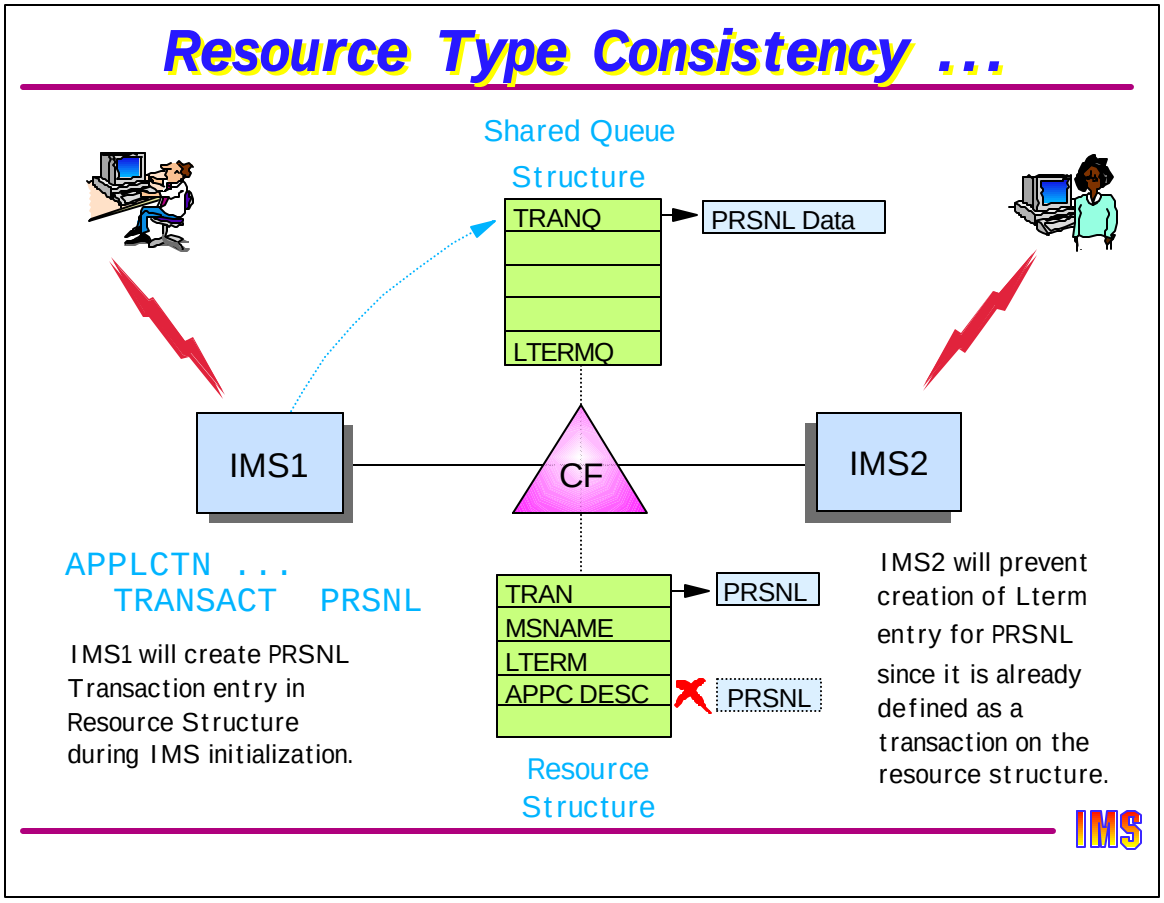
Does not apply to

- Nodes, users, userids
- These are not message queue "destinations"
  - For example, OK to have node name and lterm name the same

IMS



This diagram shows an example of Resource Type Consistency in action. Assume that IMS1 has defined the name PRSNL as a TRANSACTion in its system definition. Assume that IMS2 has defined the same name (PRSNL) defined as an Lterm. When an input message arrived to IMS1 with the first 8 characters being PRSNL, IMS1 would queue it on the Transaction Ready Queue. If the user on the node defined with the same Lterm name logs on to IMS2, IMS2 would queue messages for PRSNL to the Lterm Ready Queue, but IMS1 would queue messages for PRSNL to the Transaction Ready Queue. Very confusing.



With STM, when IMS1 is initialized, it will create a Transaction entry in the resource structure, including PRSNL. This entry will never be deleted unless the structure itself is deleted.

If a user tries to log on to IMS2 from an Lterm named PRSNL, IMS2 would attempt to create an Lterm entry in the resource structure but would fail because it already exists as a Transaction. The logon would be rejected.

Resource Name Uniqueness

STM prevents some resource types from being active in more than one IMS

- These resources are *owned* by one IMS while active
  - Ownership maintained in structure

Applies to

- *Single session VTAM Nodes, (ETO) Users, Lterms*
- *Userids*
  - Only if single signon requested by first IMS to join IMSplex

Does not apply to

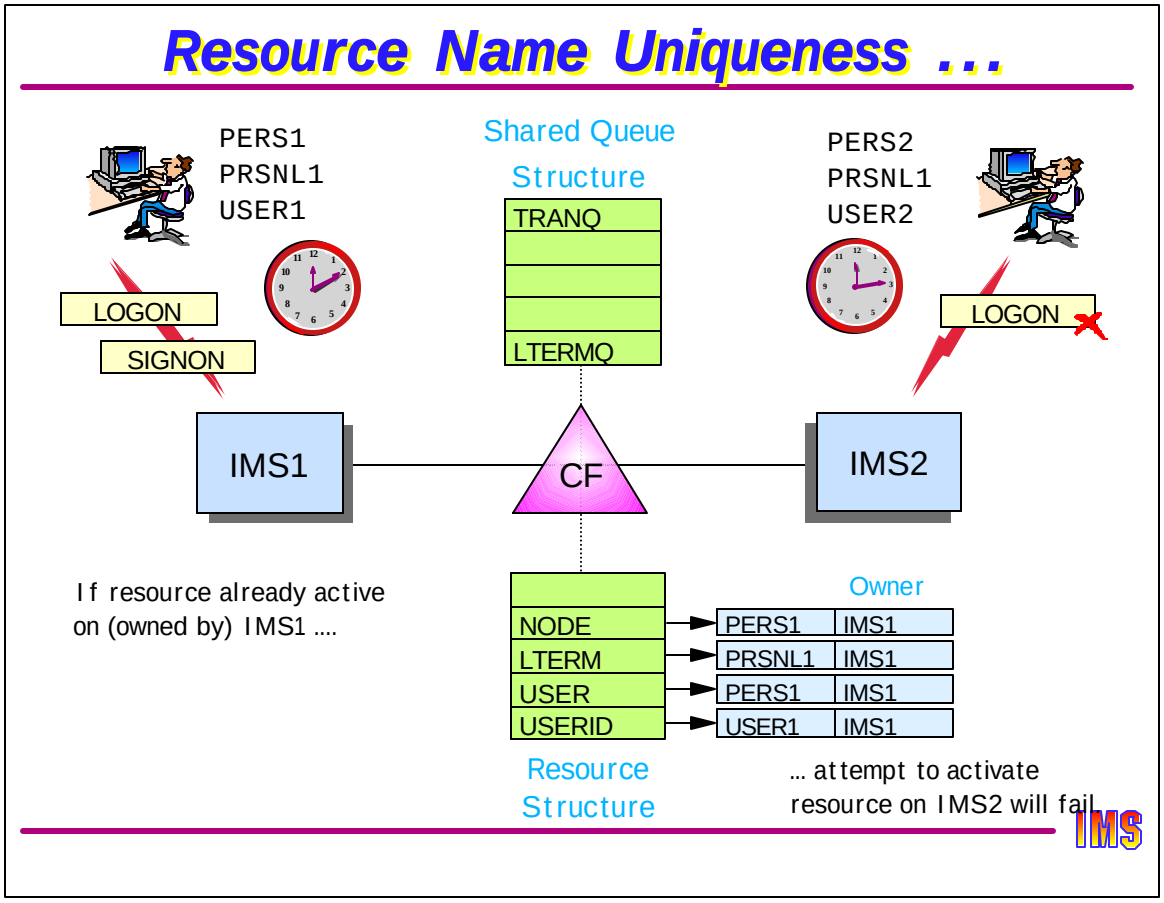
- Transactions
- Parallel session VTAM nodes
- Msnames
- APPC descriptor names
- Userids if SGN=M

IMS

Resource name uniqueness prevents multiple resources with the same name from becoming active on more than one IMS at a time. For example, if Lterm PRSNL1 were defined to IMS1 as the Lterm for PERS1, and the same Lterm name defined to IMS2 as the Lterm for PERS2 (or perhaps someone assigned Lterm PRSNL1 to PERS2), whichever Lterm is successful logging on first gets the Lterm entry in the resource structure. The second logon attempt would be rejected since PRSNL1 is already active.

Resource name uniqueness applies to nodes, users, and lterms. It may also apply to userids if the user wants to enforce single user signon. Single signon is the default, but can be overridden at startup by coding SGN=M in DFSPBxxx.

When a resource is active on any IMS, that IMS is the "owner" of the resource. Ownership, discussed more later, is kept in the resource entry.



This example shows what would happen is the Lterm PRSNL1 were assigned to node PERS1 in IMS1 and PERS2 in IMS2. This assignment may be by system definition or by command. Assuming PERS1 logs on first, IMS1 would create an Lterm entry for PRSNL1 with itself as owner. When PERS2 logs on, IMS2 would try to create an Lterm entry for PRSNL1 but would fail since it already exists and is owned by IMS1. The logon would be rejected.

Resource Status Recovery

Recoverable status

- If status is known to IMS when resource reconnects
  - Recover (restore) status
- Saved in Resource Structure
  - Node, lterm, or user entry

Significant status

- When session terminates, IMS will not delete entry if it has ...
  - End-user significant status
    - Conversation, fast path response mode, STSN
  - Command significant status
    - STOP, EXC, TEST MFS, TRACE
    - ASSIGN or CHANGE USER with SAVE keyword
- Not all recoverable status is significant, for example ...
  - LTERM assignments made without SAVE keyword

IMS

Resource status recovery is the most complex of the STM functions. There are several terms which must be understood when discussing this topic.

- Recoverable status - this is status that IMS will restore when an inactive resource becomes active again IF IMS knows what that status is (i.e., if IMS did not delete that status when the resource became inactive). Recoverable status will be saved in the resource entry for that resource if global recovery is requested.
- Significant status - this is recoverable status which will cause IMS to NOT DELETE a resource entry when the resource becomes inactive. There are two types of significant status:
  1) End-user significant status - this includes conversational status, STSN status (sequence numbers), and fast path response mode status. If a resource structure exists, this status may be saved in the resource entry.
  2) Command significant status - this is status which is set by command. There are six commands which can produce command significant status as shown on the visual. If a resource structure exists, this status will be saved in the resource entry.

Two points - not all recoverable status is significant. For example, if an Lterm is assigned to another node without the SAVE keyword, this is not considered significant and that assignment will not prevent the resource entry from being deleted when the resource becomes inactive. Secondly, the user may elect, for each resource, whether end-user status will be saved in the resource entry or not. See next visual.

Resource Status Recovery ...

Status recovery mode (SRM)

- Determines *where* end-user significant status is maintained
  - GLOBAL - in the *resource structure*
  - LOCAL - in *local control blocks*
  - NONE - *deleted* when resource becomes inactive

Status recoverability (RCVYxxxx)

- Determines *if* end-user significant status should be recovered
  - RCVYCONV=YES|NO
  - RCVYSTSN=YES|NO
  - RCVYFP=YES|NO

Set at logon or signon time

- System default set in DFSCGxxx
- Logon or Signon Exit

IMS

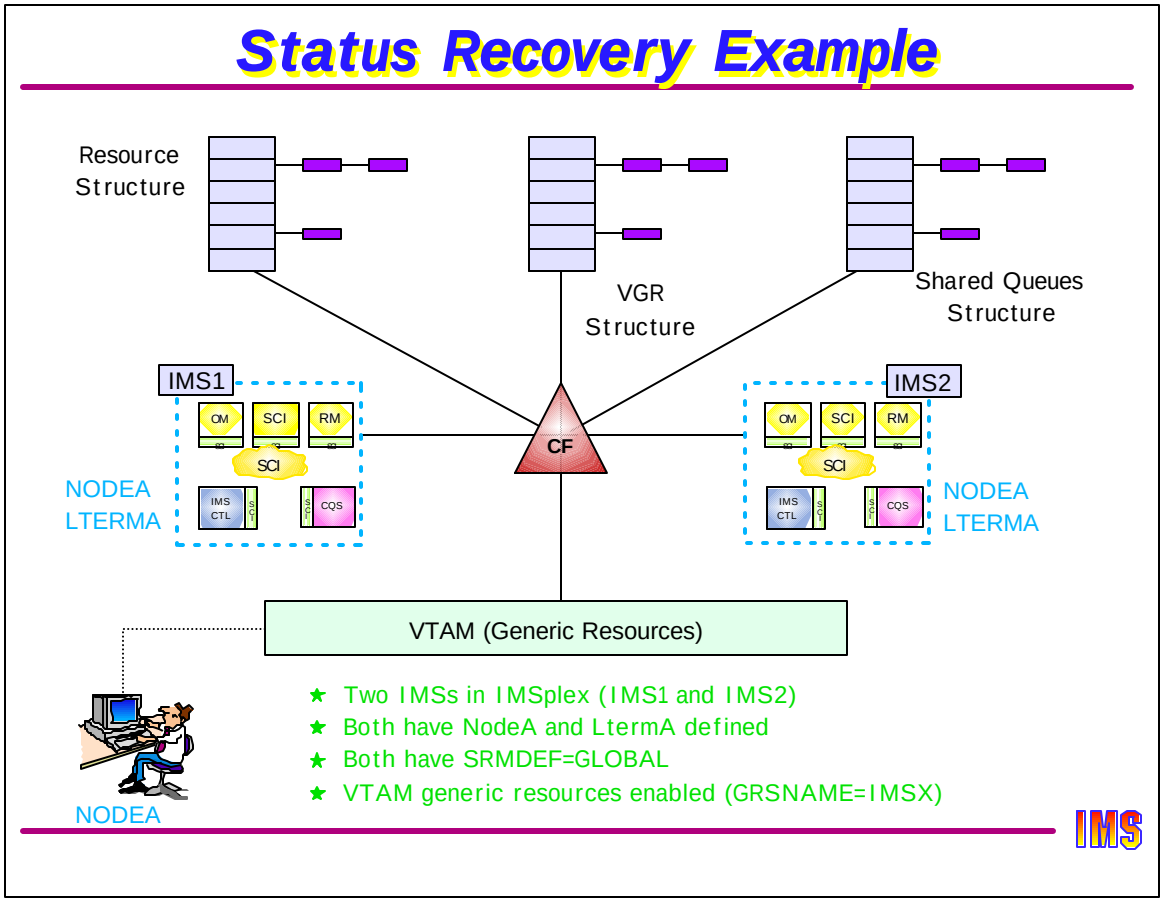
The Status Recovery Mode (SRM) determines where end-user status is maintained for each resource. There are three choices:

1. SRM=GLOBAL - end-user status will be kept in the resource entry.
2. SRM=LOCAL - end-user status will be kept only in the local IMS control blocks and log records; this is the way IMS works without STM.
3. SRM=NONE - end-user status will be kept only in the local IMS control blocks and only as long as the resource is active. As soon as the resource becomes inactive, the status is deleted. This is new in IMS V8 and can be specified even when not running with CSL and STM.

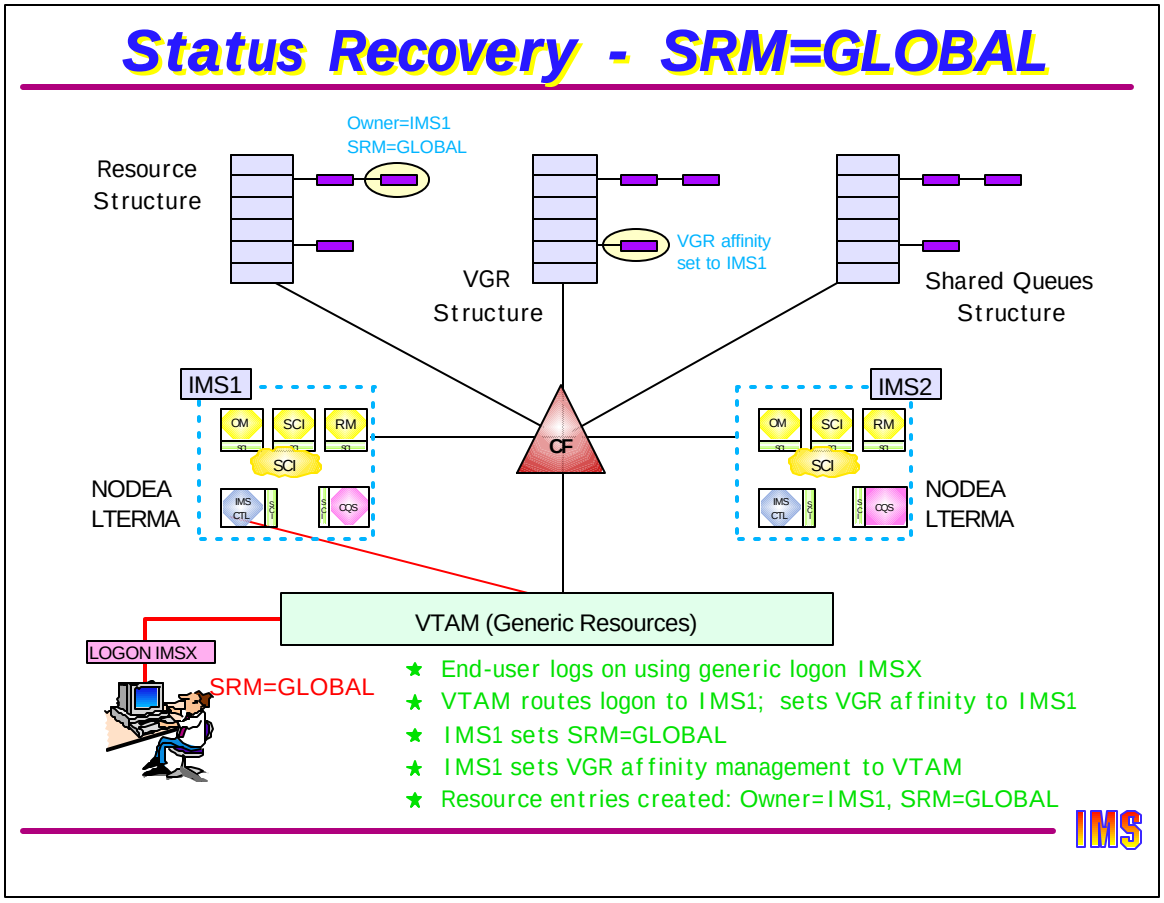
There is no SRM option for command significant status. It is always maintained globally if there is a resource structure, and always maintained locally if there is no structure.

In addition to SRM, there is another parameter that is more granular - recoverability, set by the RCVYxxxx parameter. This parameter allows the user to determine whether specific end-user statuses should be recoverable. For example, a user might decide to recover conversations but not STSN sequence numbers.

System defaults for these are set in proclib member DFSCGxxx and can be overridden for each resource at logon or signon time by the Logon Exit or the Signon Exit.

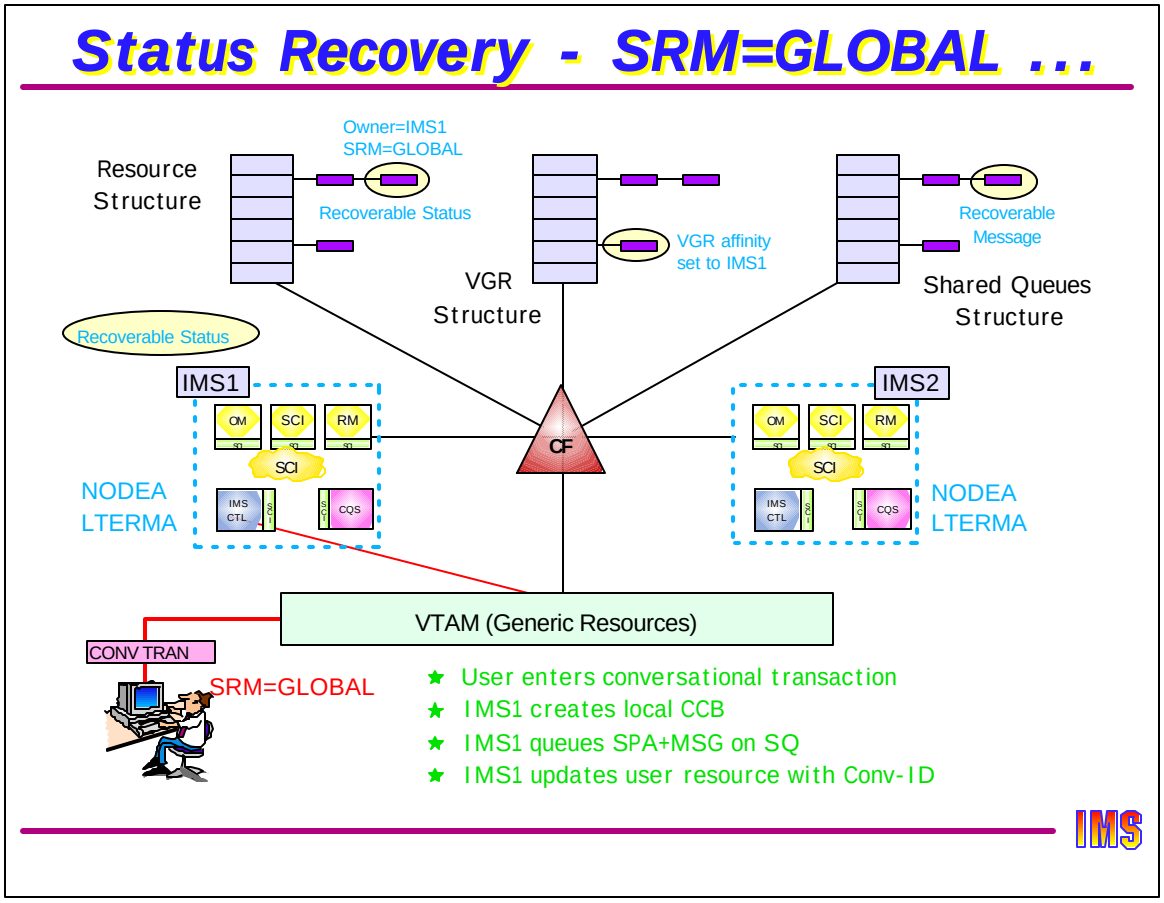


We are going to walk through an example of STM Global status recovery. This diagram shows two IMSs (IMS1 and IMS2) which each have NODEA and LTERMA defined. These IMSs are part of a data sharing group, a shared queues group, and a VTAM generic resources group (GRSNAME=IMSX).



A user logs on from NodeA using the generic resource name IMSX. VTAM will check the VGR structure and determine that this node does not currently have an affinity for any application in the IMSX generic resource group. It selects one of the active IMSs to route the logon request to. In this example, it is IMS1.

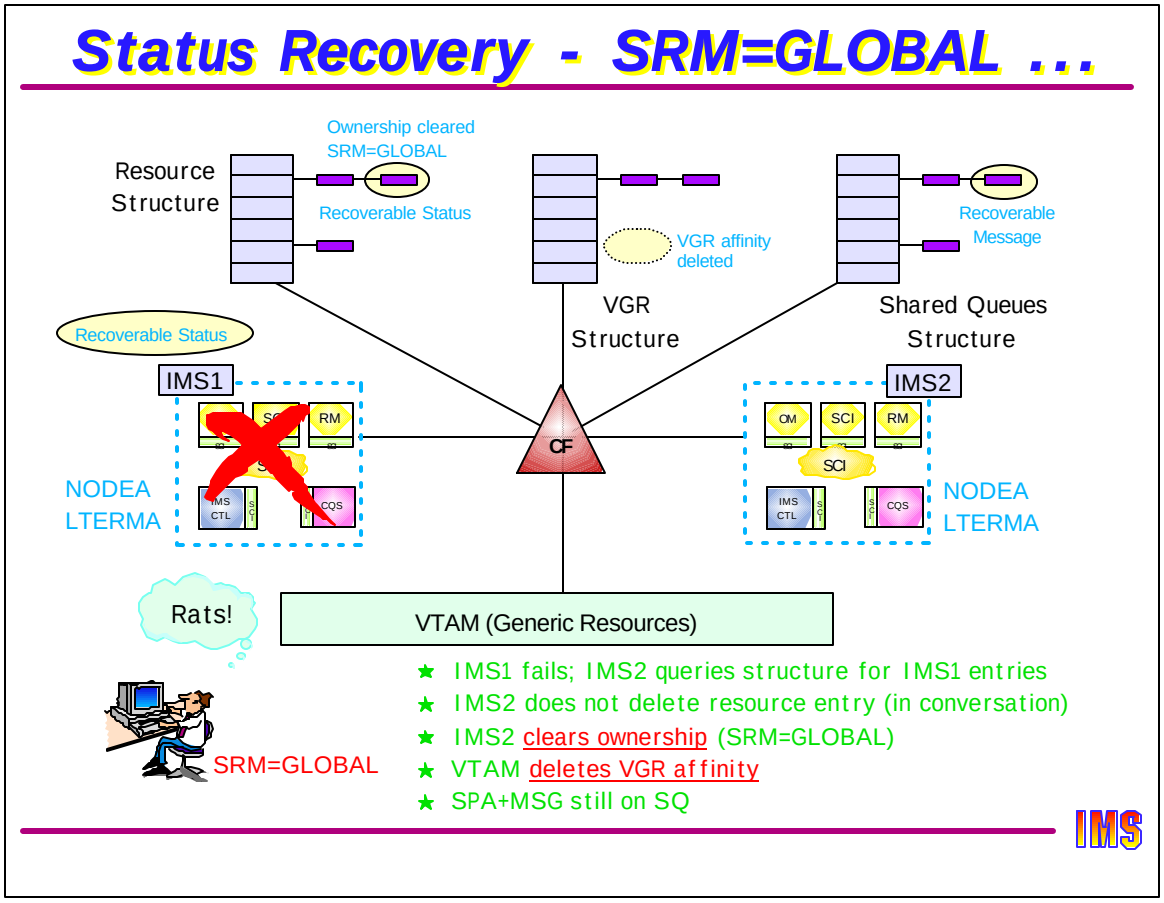
IMS1 creates entries in the resource structure for NodeA, Static-Node-UserA (also named NodeA), and LtermA. If the user signed on, and SGN was not equal to M, then an entry for the userid would also be created. All have OWNER=IMS1 and SRM=GLOBAL. If any of these resources already exist, then the create would fail ("resource name uniqueness") and IMS1 would determine the reason why. If it is because that user is currently active on (owned by) another IMS, then logon would be rejected. In this case the logon is successful. VGR affinity is set to IMS1 and, because SRM=GLOBAL, IMS1 will tell VTAM to manage the VGR affinity (session level affinity is enabled only with z/OS 1.2 and later). This means that VTAM should delete the VGR affinity when the session terminates.



The user then enters a conversational transaction. IMS1 will do several things:

- Create a new conversation by creating a CCB for that user. This user is now in conversation status on IMS1.
- Put the SPA+Input-Transaction on the Transaction Ready Queue
- Update the Static-Node-UserA resource in the resource structure to indicate the conversation id and set a flag indicating "input conversational transaction in progress" - the conversational status is also maintained in IMS1s control blocks

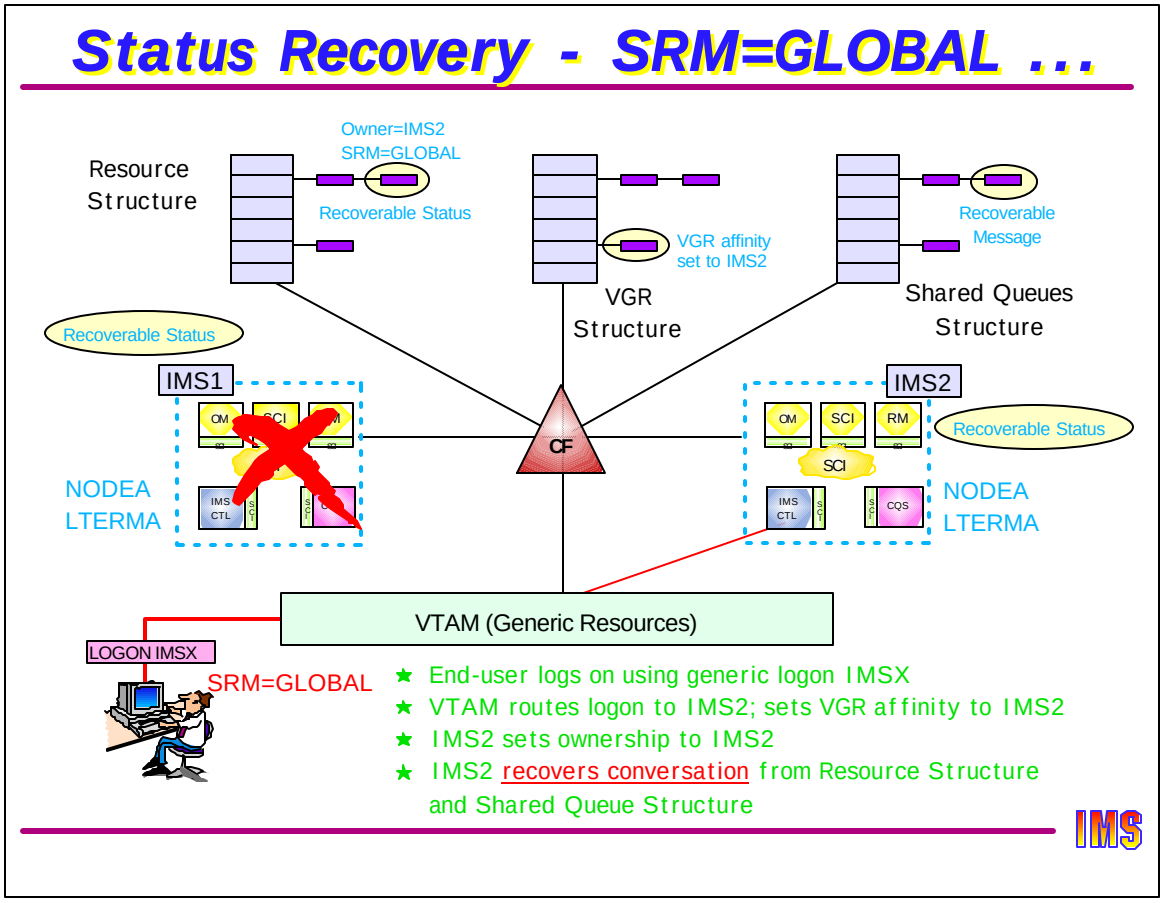
This continues for several iterations of the conversation, with the CONV-IP flag being set and reset for each transaction.



Now IMS fails while the user is preparing to enter the next transaction. The previous SPA+OUT-MSG are still on the shared queue (LOCKQ). VTAM will delete the VGR affinity.

IMS2 is informed that IMS1 has failed and initiates some "clean-up" activities.

- Queries RM for entries owned by IMS1
- Finds entries for NodeA, Static Node UserA, and LtermA. Entries indicate that SRM=GLOBAL and than the user is in a conversation (end-user significant status)
- Since end-user status is in the structure and available to any other IMS, IMS2 resets ownership to "not owned" meaning user is allowed to log on to any IMS.
- IMS2 will also requeue the SPA+OUT-MSG "locked" by IMS1 to the Lterm Ready Queue to make it available to any IMS that the user logs on to. (Locked messages are only available to the IMS that locked them. LRQ messages are available to any IMS).

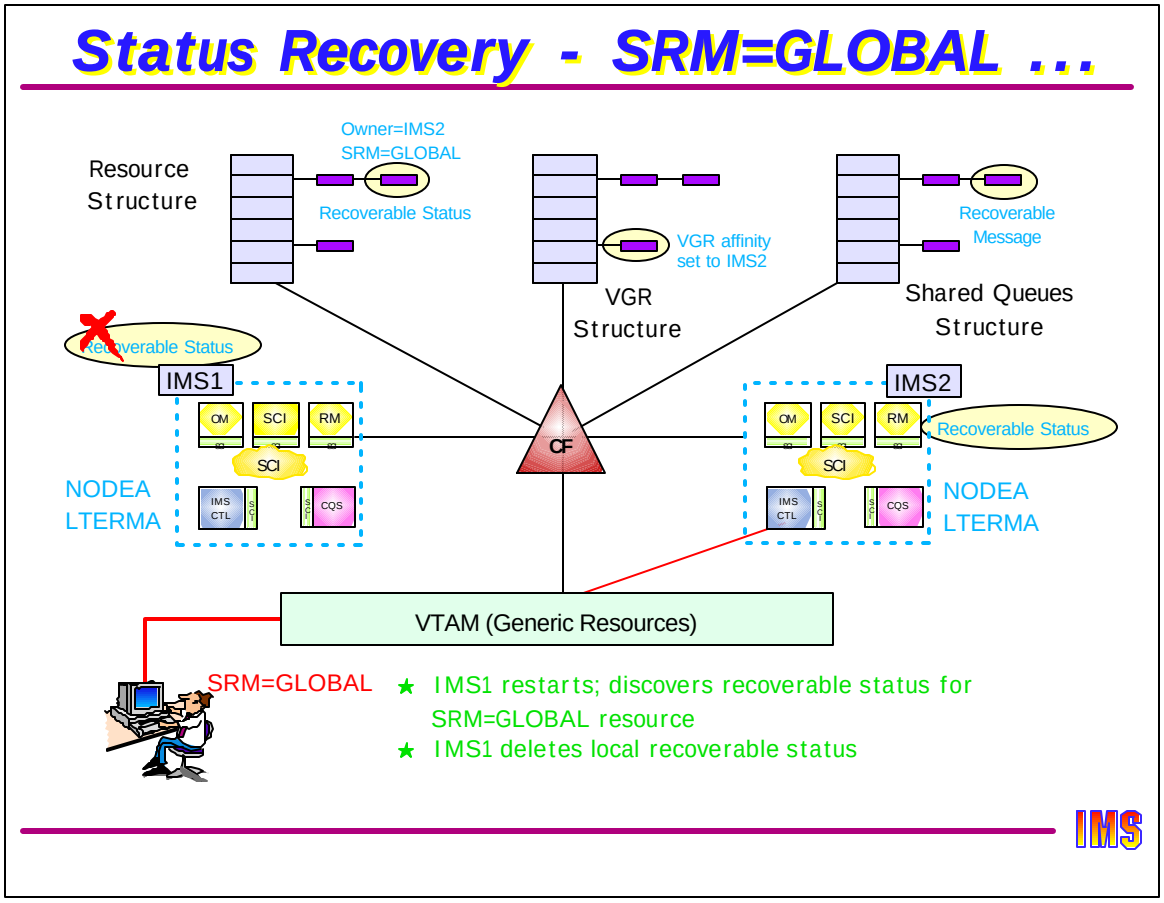


The user decides not to wait for IMS1 to be restarted and logs back on using IMSX. VTAM checks the affinity table, sees not affinity for NodeA, and routes the logon request to the only active IMS in the group - IMS2.

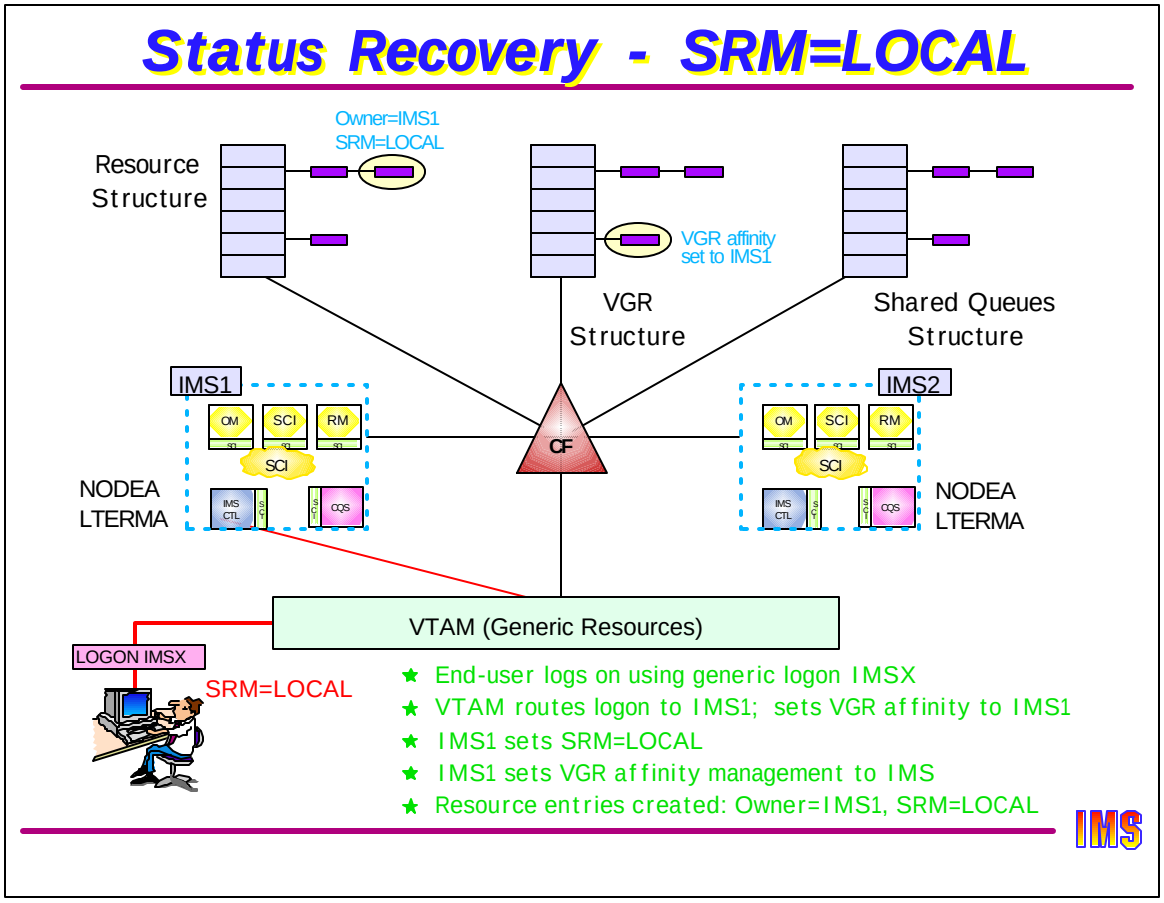
IMS2 tries to create resource entries in the structure and fails because they are already (still) there. IMS2 checks and discovers that the resource has significant status but is not owned. IMS2 accepts the logon and sets ownership to IMS2. IMS2 also "relocks" the SPA+OUT-MSG that it found on the Lterm Ready Queue (put there during cleanup processing after IMS1 failed).

The user can continue the conversation by holding and then releasing the conversation. This causes IMS2 to go back to the shared queue, retrieve the last SPA+OUT-MSG from the LOCKQ, and resend the last output to the user terminal.

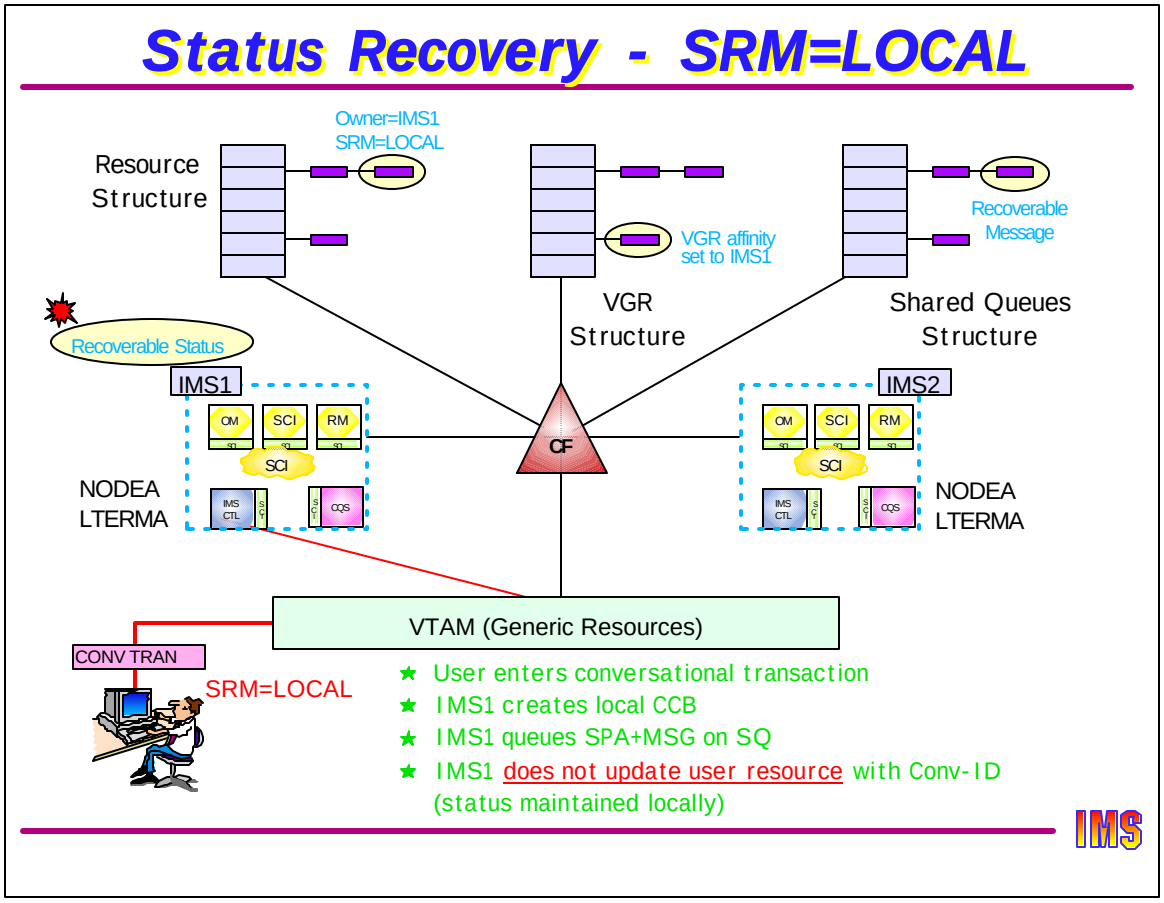
The user then continues the conversation as normal.



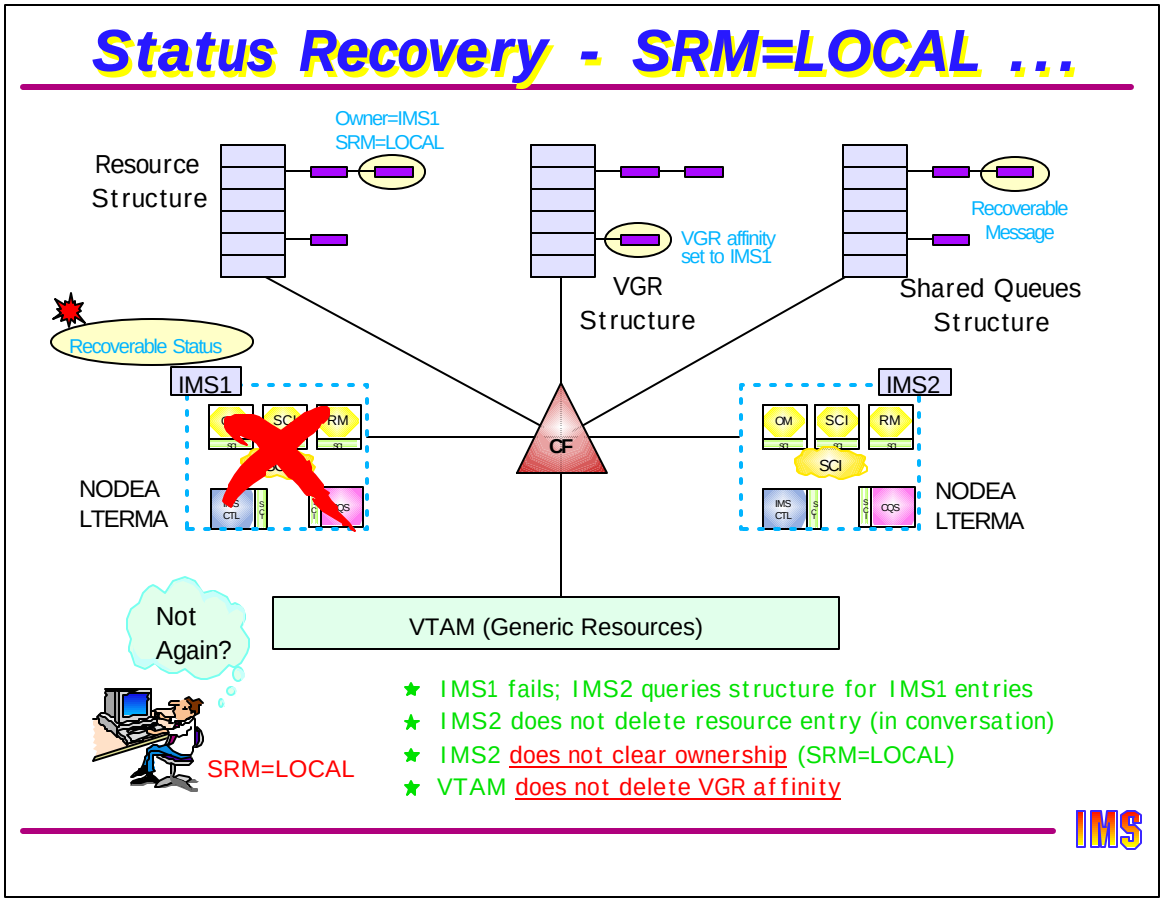
When IMS1 is restarted, its control blocks will indicate that the user was in a conversation with SRM=GLOBAL. Since the status was on the structure at the time of failure, IMS1 knows that the user may have logged on to another IMS and continued the conversation. IMS1 deletes his local status. Even if the user had not logged on to another IMS, the status is still on the structure and IMS1 can safely delete the local status.



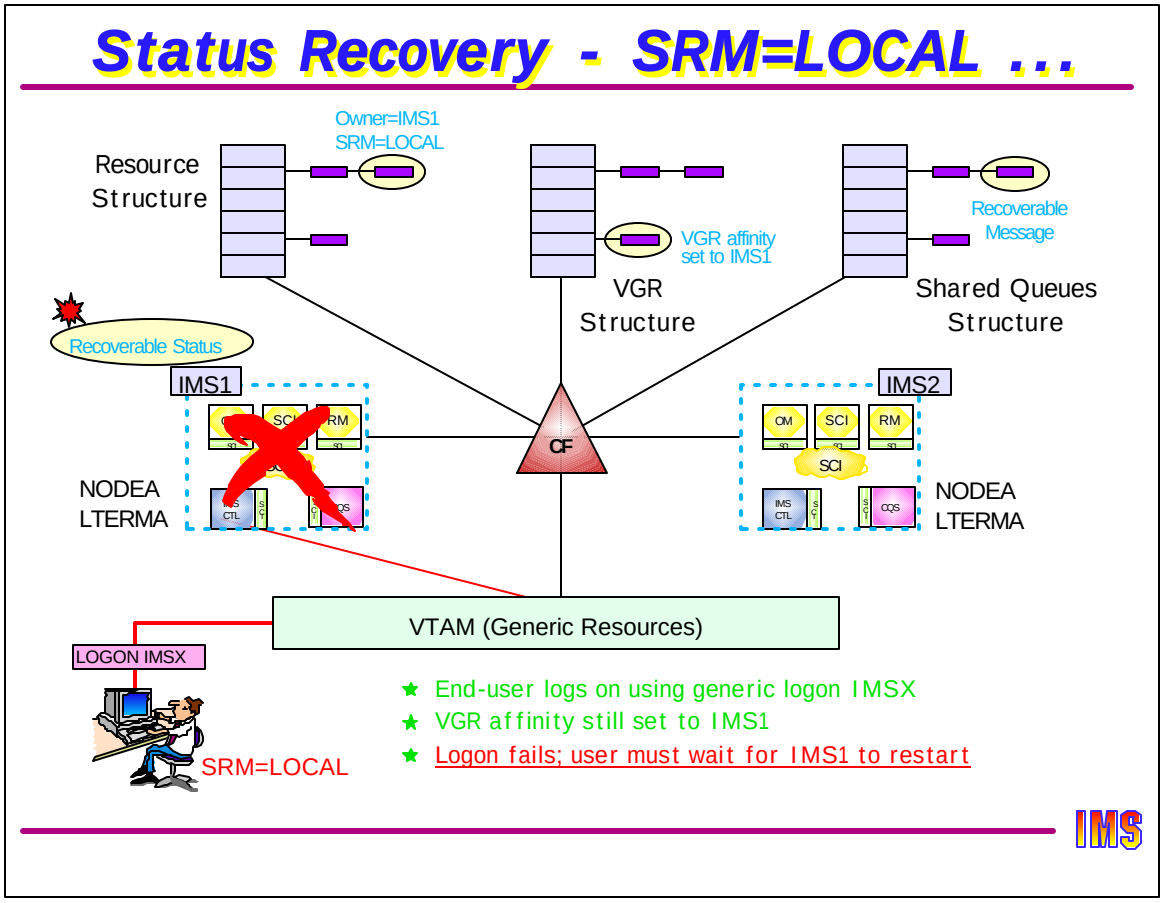
In this example of SRM=LOCAL, there are a few differences.
- The VTAM GR affinity is set to IMS-managed instead of VTAM-managed
- The SRM value in the resource entry is set to LOCAL



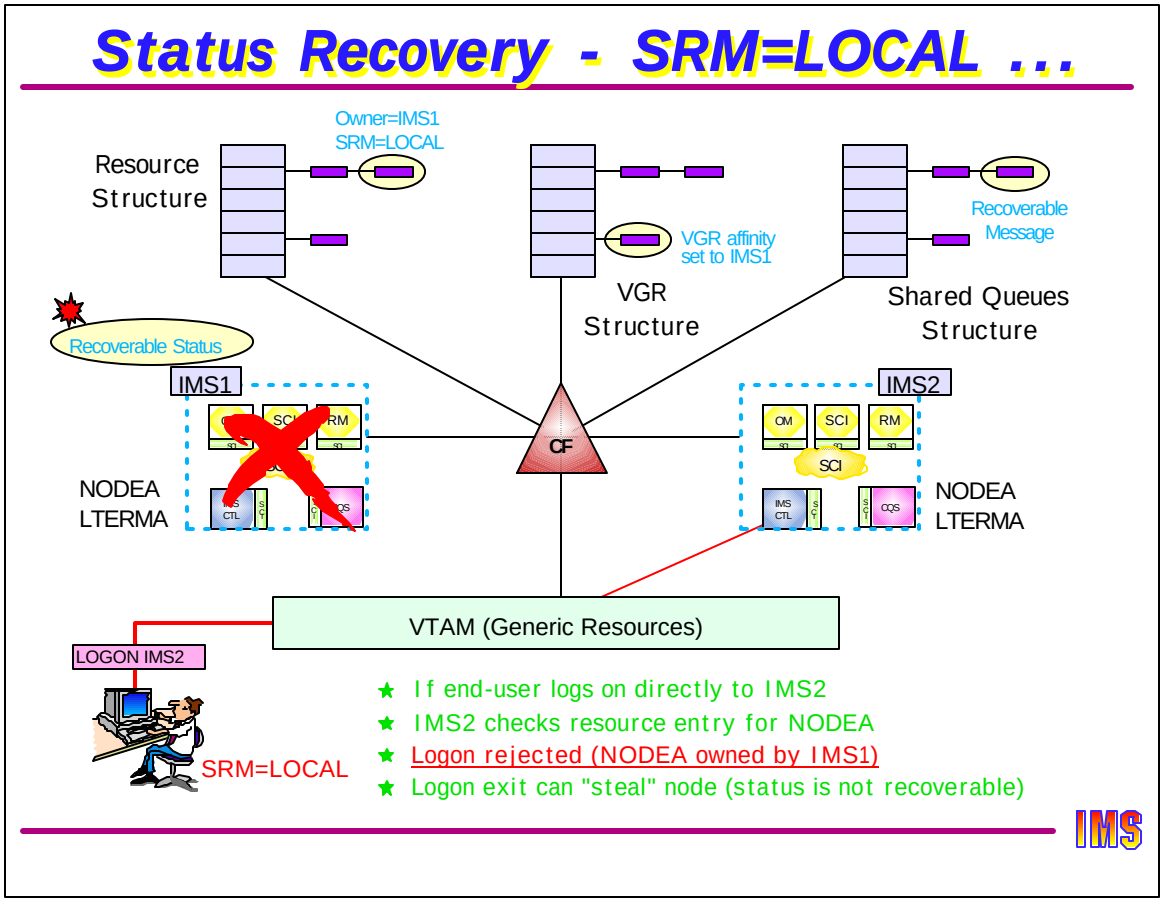
When the user enters a conversational transaction, the conversational status is NOT maintained in the resource structure. It is know only to IMS1.



When IMS1 fails and IMS does cleanup, IMS2 sees that the SRM=LOCAL for the resource and does not know whether there is any end-user status. So, IMS2 does not delete the resource entry, and IMS2 does NOT clear ownership. Also, the VGR affinity, which is IMS managed, is NOT cleared by IMS1 since the terminal is in conversational mode with SRM=LOCAL. IMS1 is the only IMS that knows anything about the conversation.

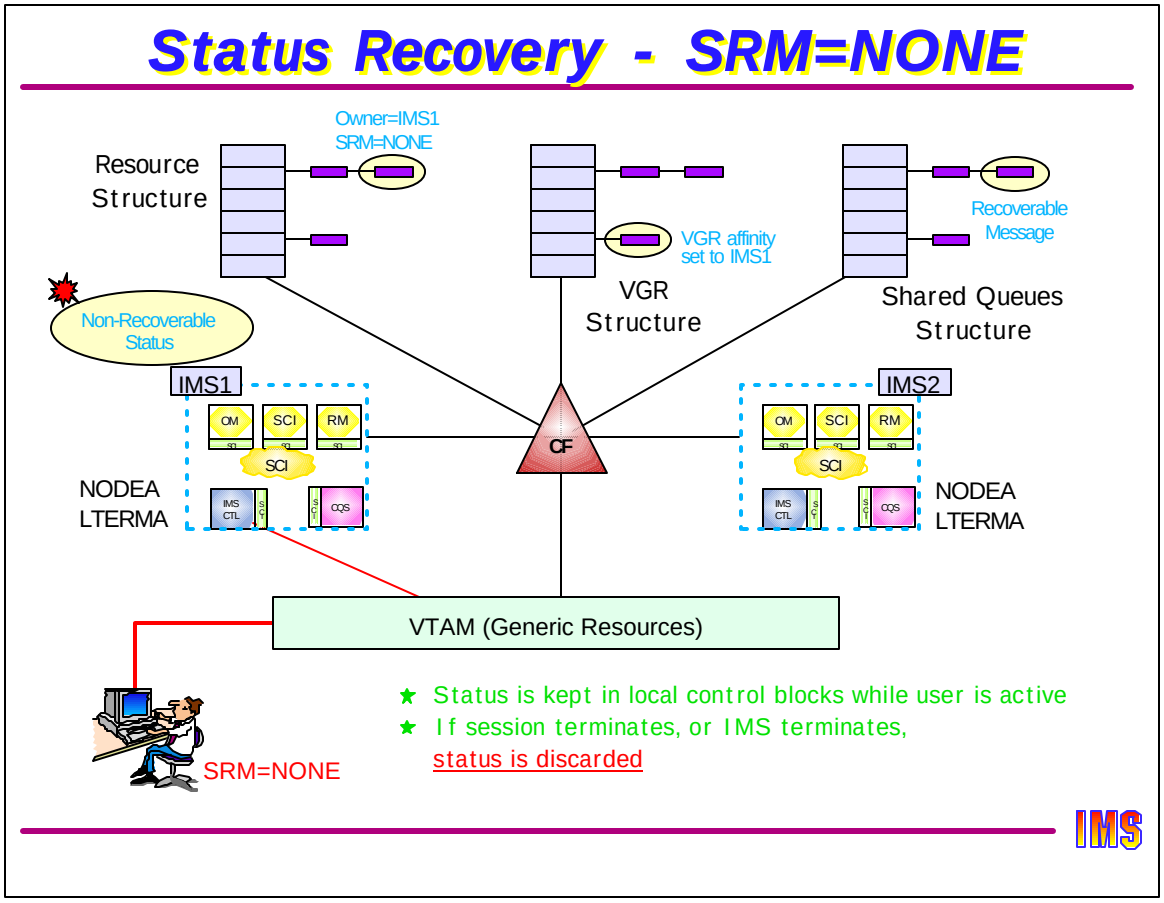


If the user tries to log on again using IMSX, VTAM will discover that NodeA has an affinity for IMS1. Since IMS1 is not active, the logon will fail. The user must wait for IMS1 to restart.



If the user gets tired of waiting (maybe he knows that IMS1 will be down for an extended period of time) and logs on directly to IMS2, IMS2 will check the NodeA entry and find that NodeA is owned by IMS1 and reject the logon.

Too bad, HOWEVER, IMS2 can "steal" the node if the logon exit (DFSLGNX0) says its OK. A clever user shop will write a logon exit to recognize some user data entered at logon as a request to steal the node. For example, the user data might say "I really mean it!" If this happens, then the conversation is not recoverable. When IMS1 is restarted and finds the resource with recoverable status and SRM=LOCAL, IMS1 will check the resource structure and discover that NodeA/UserA/etc. are no longer owned by IMS1 and delete the status. That conversation is toast.



SRM=NONE (or RCVYxxxx=NO) means that end-user status is NOT to be recovered, even locally. When a session terminates with SRM=NONE, all end-user status is deleted. This might be useful with ETO, which will not delete control blocks if there is any status. For example, and ETO STSN device ALWAYS has status (always has a sequence number). SRM=NONE for these devices would delete the status, and then the resources themselves could be deleted.

CSL Highlights

Global Online Change

- ★ Enabling G-OLC
- ★ Executing G-OLC
- ★ G-OLC commands

IMS

Next topic is Global Online Change, sometimes called Coordinated Online Change, or even Coordinated Global Online Change. We will call it Global Online Change (G-OLC).

Enabling Global Online Change

Global OLC enabled by DFSCGxxx Proclib member

- Requires CSL environment
  - Resource structure not required, but useful
- DFSCGxxx
  - OLC=GLOBAL | LOCAL
    - Not all IMSs in IMSplex have to participate in Global OLC
  - OLCSTAT=OLCSTAT data set name
    - OLCSTAT data set replaces MODSTAT
    - All IMSs with OLC=GLOBAL must use same OLCSTAT data set
    - IMSs with OLC=LOCAL continue to use MODSTAT
  - NORSCCC=(MODBLKS,ACBLIB,FORMAT)
    - Turns OFF online change data set name consistency checking for these data sets
    - Unless turned off, all IMSs must use same OLC data sets

IMS

G-OLC requires a CSL environment with all the address spaces, but does not require a structure. If one exists, it will be used but is only beneficial for restarting G-OLC after certain types of failures.

New proclib member DFSCGxxx contains three parameters related to G-OLC.

- OLC=GLOBAL or LOCAL indicates whether or not that IMS wishes to be part of the global online change process. It is OK to have some IMSs global and other local.
- If global is specified, then OLCSTAT= gives the data set name of a new OLC status data set that must be used by all global IMSs. The data set is dynamically allocated. MODSTAT is not required for G-OLC but is still used for L-OLC.
- The default is that all IMSs will use the same OLC data sets (MODBLKS, ACBLIB, FORMAT). If you don't want to use the same data sets (same DSNs) then specify NORSCCC= for whichever ones are different. It is the users responsibility to be sure that even though the data sets are different, the contents are the same.

Enabling Global Online Change ...

OLCSTAT data set

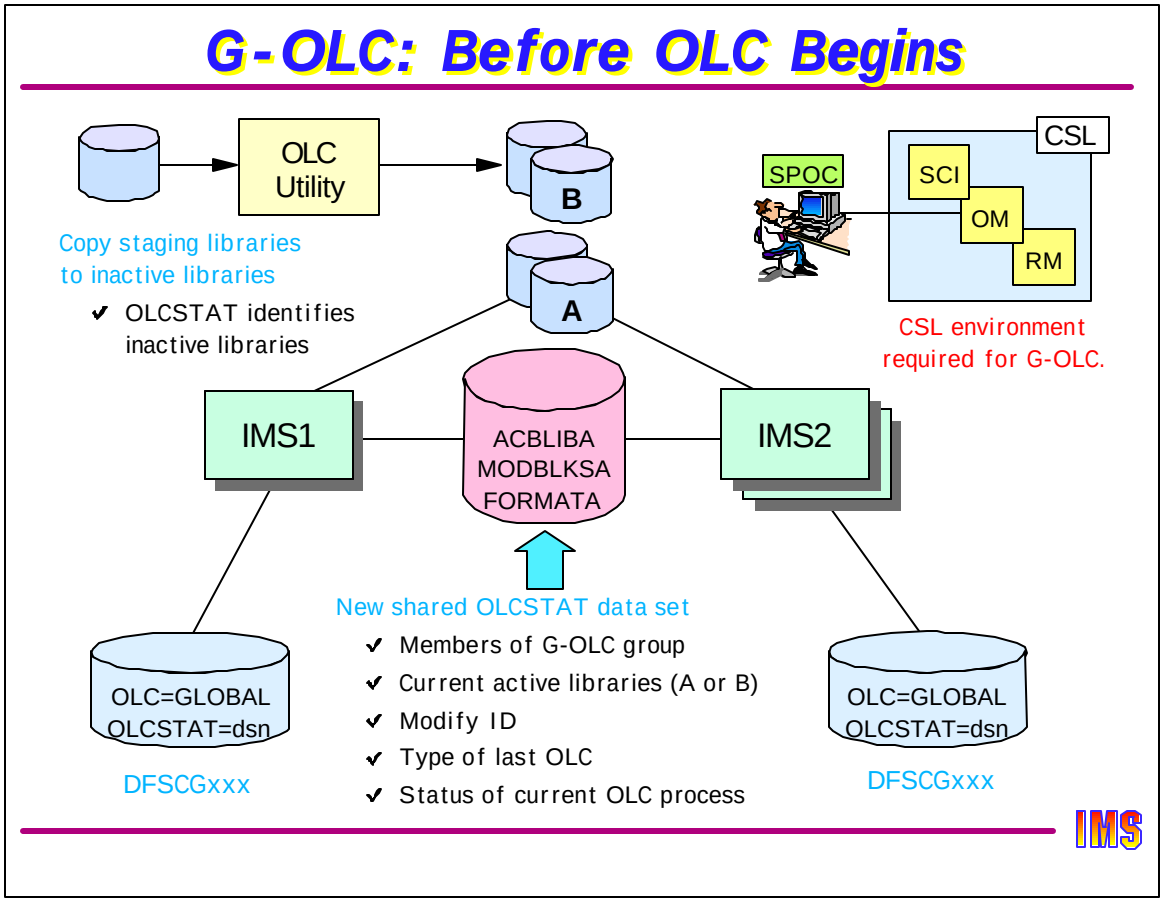
- Must be initialized with Global OLC Utility (DFSUOLC0)
  - Sets initial OLC library suffixes (A or B)
- Header record
  - Current active library suffixes (A or B)
  - Modify ID of last successful G-OLC
  - Type of last successful G-OLC
  - G-OLC in progress flag
- IMS record
  - One for each IMS with OLC=GLOBAL
  - Created as each IMS cold starts
  - Deleted if IMS shutdown with /CHE FREEZE LEAVEPLEX
  - Deleted if IMS "misses" a global online change
    - May require cold start

IMS

The OLCSTAT data set contains information about the G-OLC environment. It must be initialized one time by the Global OLC Utility (DFSUOLC0) which will set the initial OLC library suffixes to A or B in a Header Record.

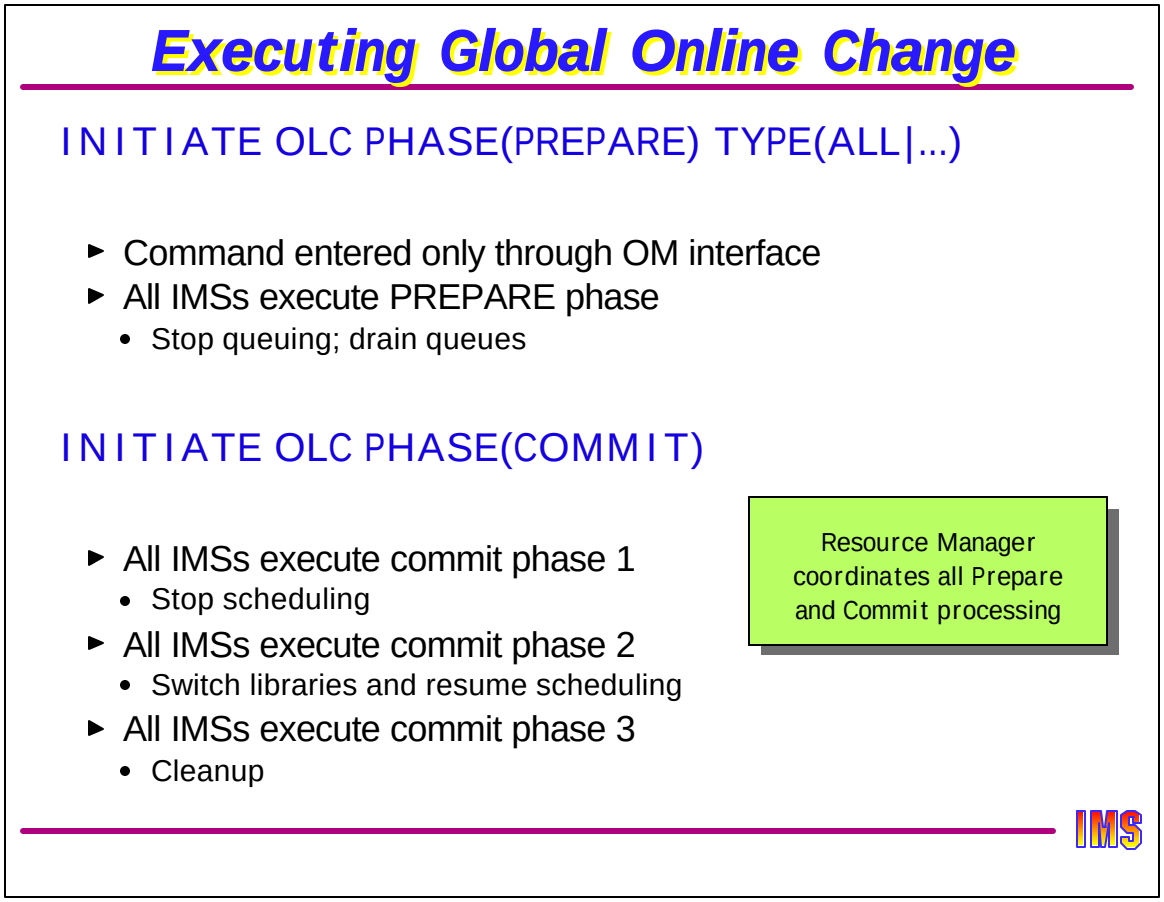
The Header Record identifies the G-OLC suffixes for each library, the modify-id of the last (current) OLC, the type of OLC last performed, and a flag indicating that G-OLC is in-progress. The flag is to keep someone from trying to start another G-OLC while one is in-progress.

Then each IMS with OLC=GLOBAL and is current (participated in the last G-OLC) has an entry. During G-OLC, this is used to keep track of the status of each IMS - what phase it is in and whether is has completed that phase. If IMS "misses" a G-OLC because it was not active when G-OLC was executed, then that IMS might have to cold start, depending on how many G-OLC processes it missed, and what the last type was. For example, if the last type was FORMAT only, then a warm start is OK. Otherwise, a cold start is required.



The next few charts walk through a G-OLC process. They are somewhat simplified, but the general concept and flow is correct.

Before G-OLC begins, the usual preparation and staging of libraries must be done. Libraries to be changed must be copied from the staging libraries to the inactive libraries by the OLC Copy Utility. The utility will query OLCSTAT to determine what the inactive libraries are. If different IMSs are using different data sets, then this must be done for each data set.



When the copy is done, an operator initiates the G-OLC process from a SPOC. The command is:

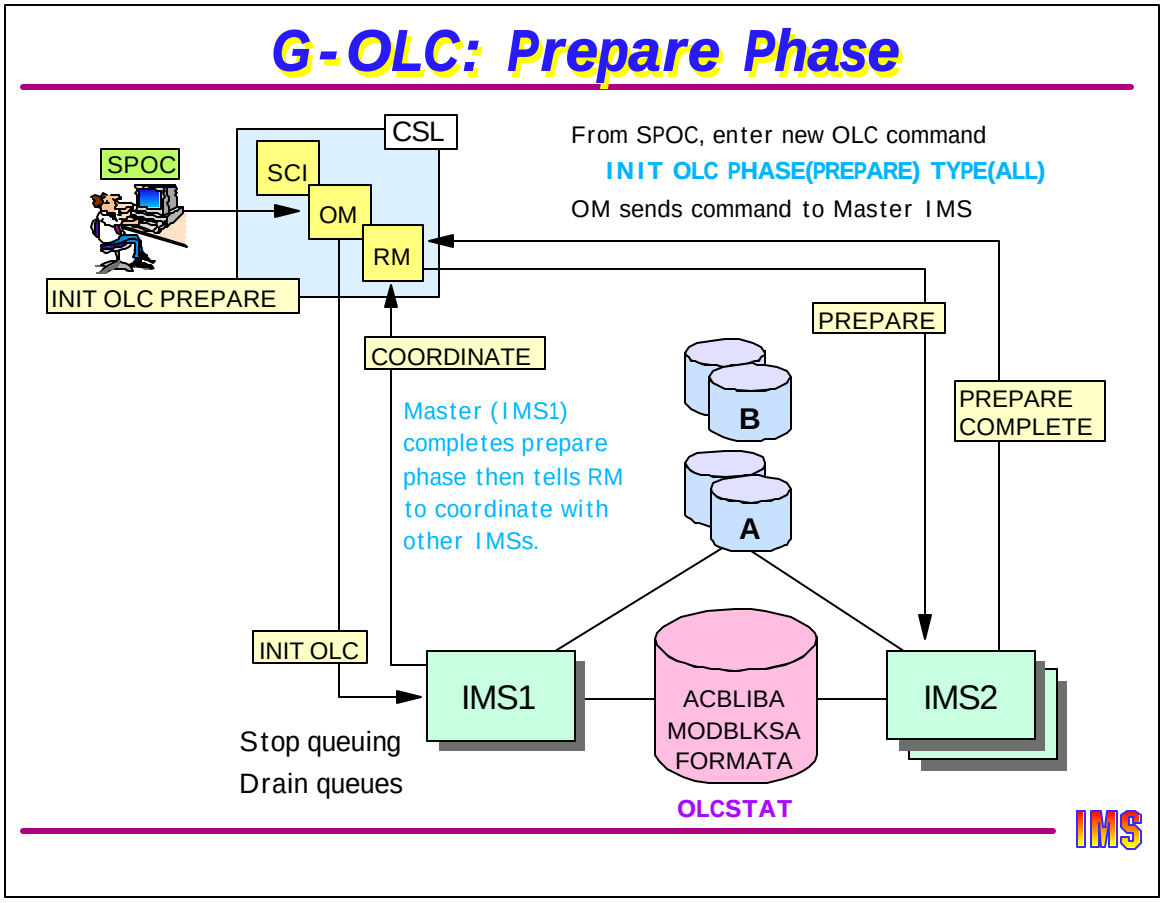
INIT OLC PHASE(PREPARE) TYPE(ALL|ACBLIB|CTLBLKS|FORMAT))

During the prepare phase, all IMSs stop queuing new work and attempt to drain the queues for those resources being changed. When all IMSs complete the PREPARE phase, the operator enters a command to commit:

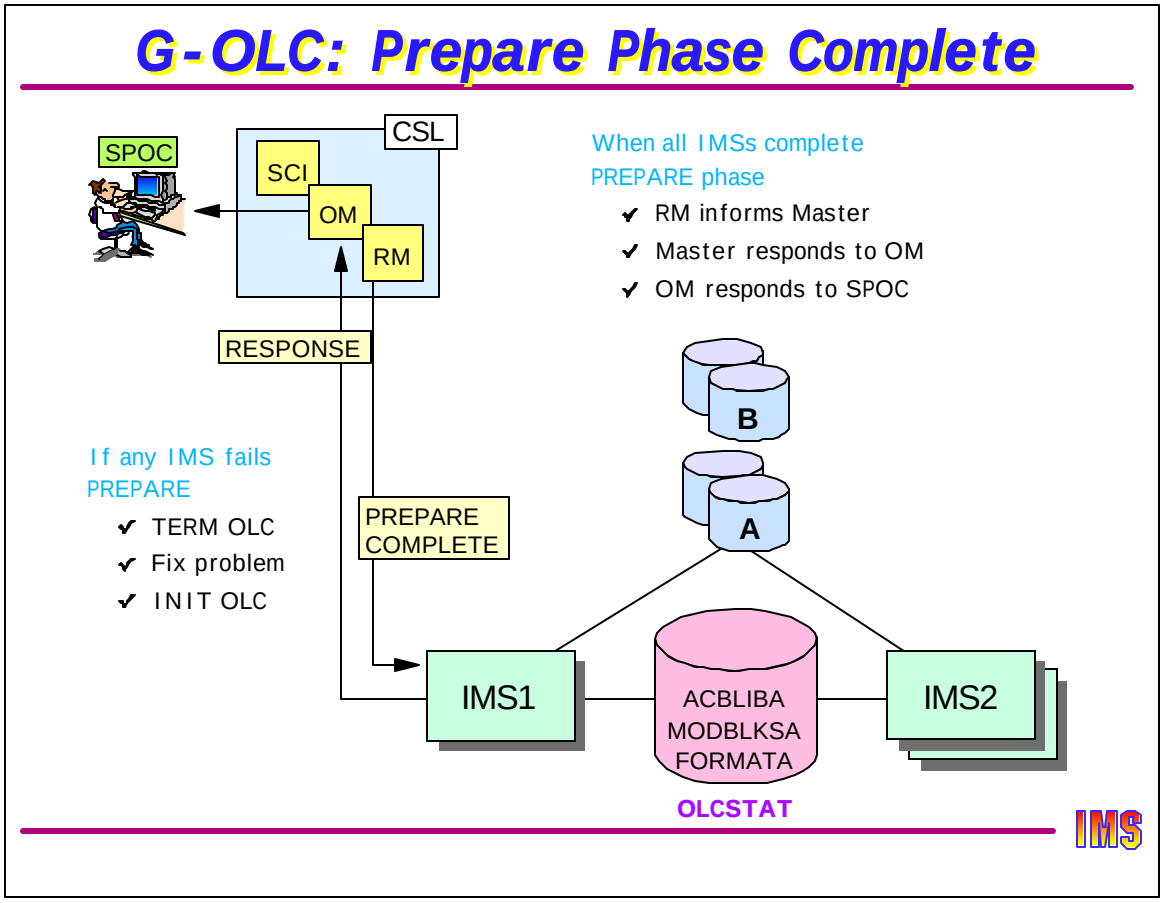
INIT OLC PHASE(COMMIT)

Commit is executed in three phases. Each IMS must complete each phase before any IMS goes on to the next phase.

1. Stop scheduling
2. Switch libraries and resume processing
3. Clean up

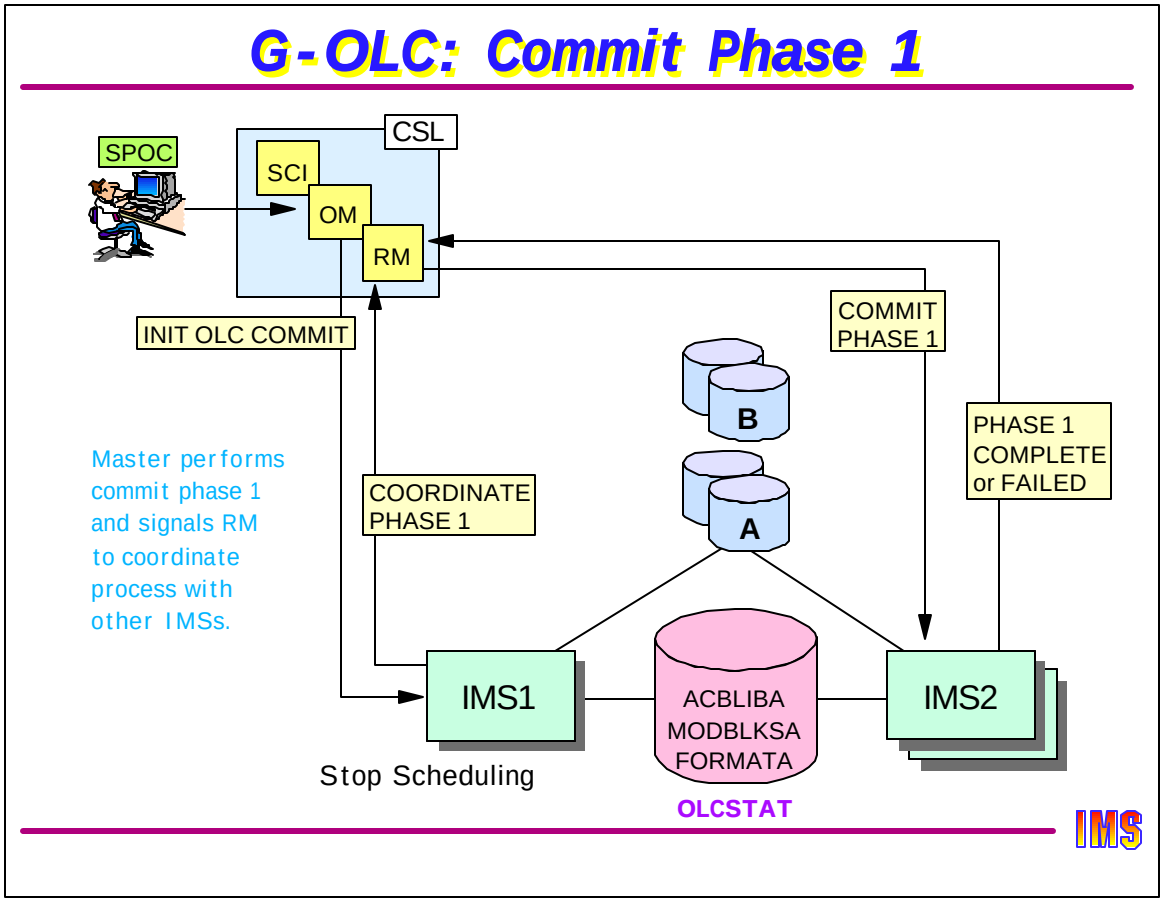


In a bit more detail now, when the PREPARE command is entered, OM sends the command to one of the IMSs in the G-OLC to be the PREPARE MASTER. That IMS will perform the prepare phase locally and, if successful locally, tells RM to coordinate the prepare phase across the other IMSs. As each IMS completes its prepare processing, it notifies RM.



When RM gets a response from all IMSs, it notifies the Prepare Master who then notifies the operator that prepare is complete. If any IMS fails the prepare phase (for example, an I/O error on ACBLIB), then the operator must terminate OLC with the TERMINATE OLC command, fix the problem, and then repeat the process.

The message to the operator indicates the OLC PREPARE completed successfully, or it provides a return code and reason code for why it failed.

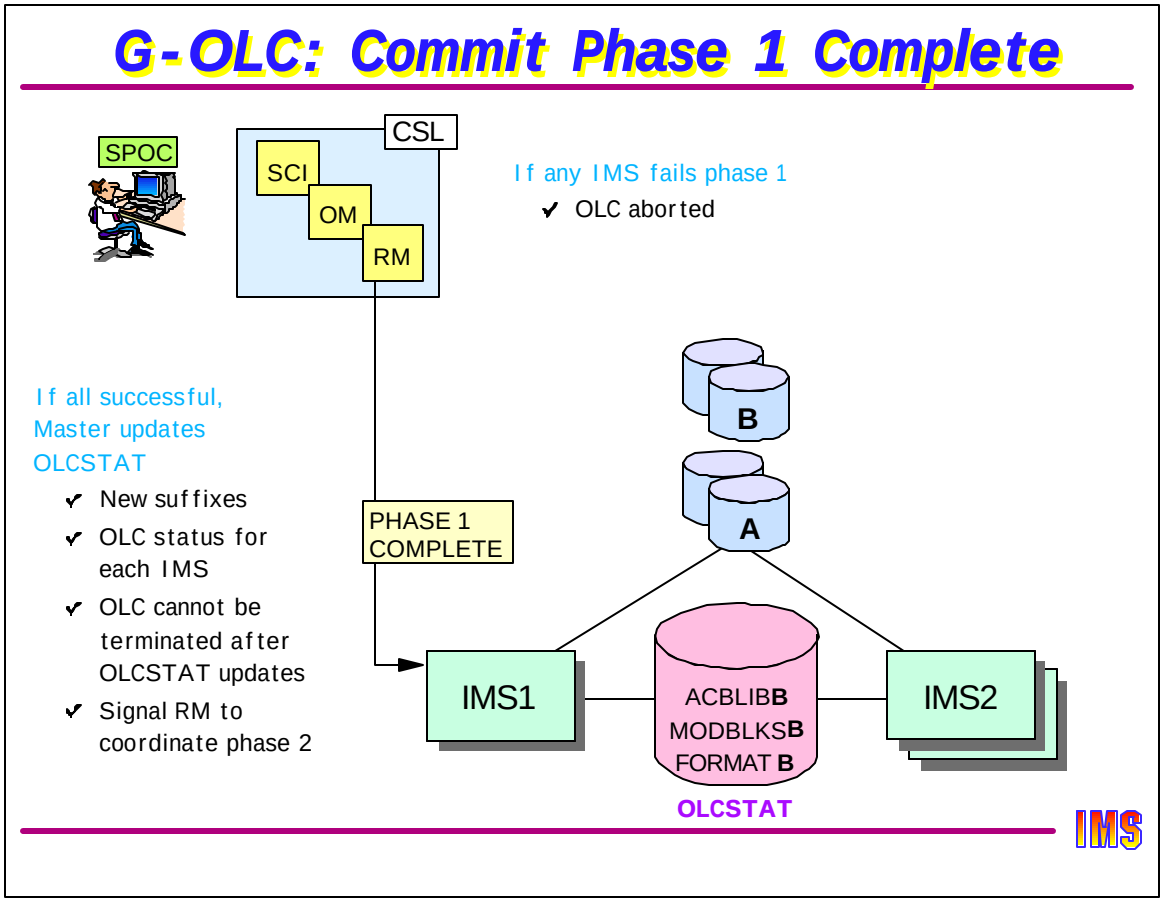


Once the operator receives an indication that all IMSs have successfully completed the PREPARE phase, he must enter the command to commit the change.

INIT OLC PHASE(COMMIT)

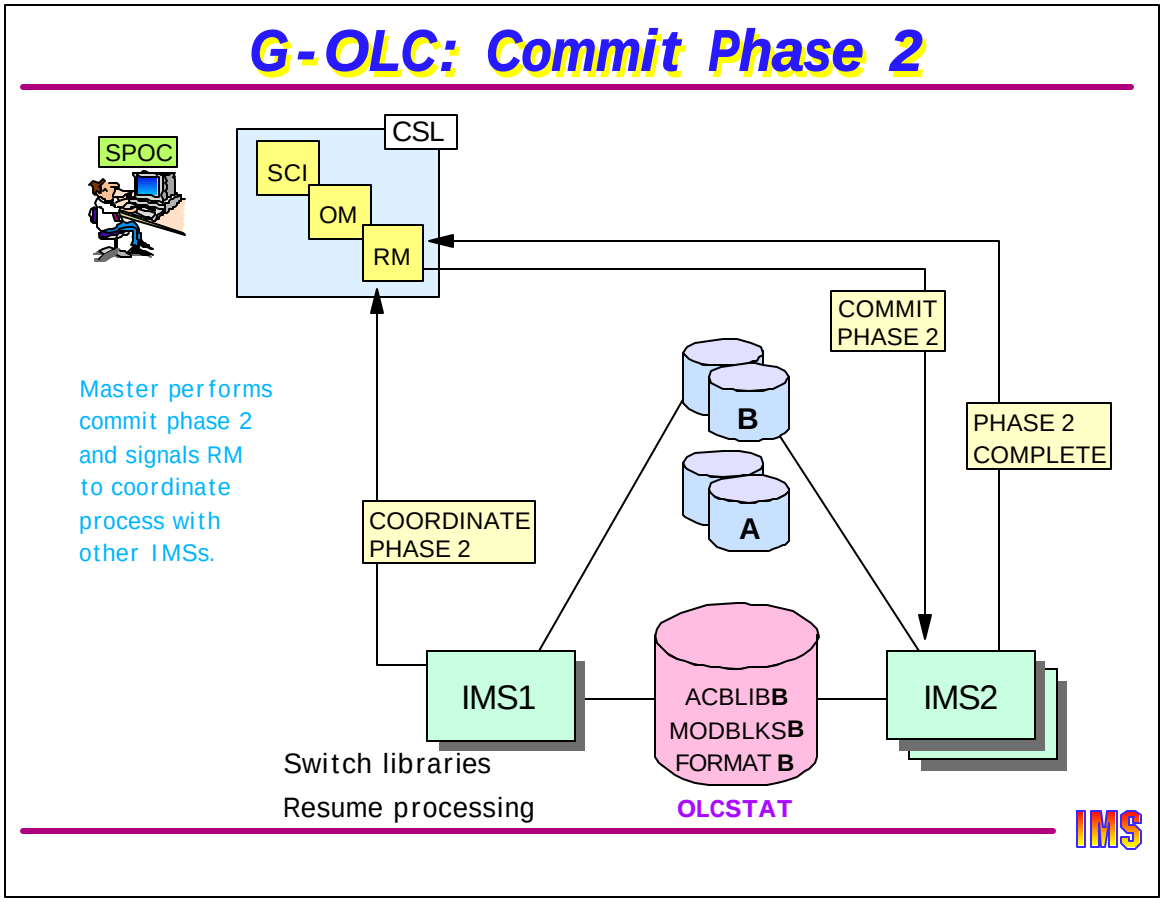
Again, OM selects one IMS to be the master and sends the command. The master tells RM to coordinate Commit Phase 1 and performs phase 1 processing itself. These are done in parallel (the master doesn't complete this phase first before telling RM to coordinate as it does with the prepare phase).

Each IMS performs Commit Phase 1 processing (stops scheduling) and informs RM of success or failure. Commit Phase 1 could fail because, after prepare completed successfully, something was scheduled (e.g., a BMP, a CICS transaction, an ODBA connection, ...) which uses resources to be changed.



Assuming all IMSs signal RM that they have completed Commit Phase 1 successfully, the master updates the OLCSTAT data set. Once the master updates the OLCSTAT data set, after which it is no longer possible to terminate OLC - it is committed. If any IMS fails after OLCSTAT is updated, the problem must be corrected and INIT OLC PHASE(COMMIT) reentered.

If any IMS fails Commit Phase 1, OLC must be terminated.



Once the OLCSTAT data set updates are complete, the master signals RM to coordinate Commit Phase 2 with the other IMSs. During this phase, the IMSs switch libraries and resume processing. When this is done, they signal Phase 2 complete. If any IMS fails Commit Phase 2, a return code and reason code explain why and the problem must be corrected and the commit command reentered. Phase 3 cannot begin until all IMSs have completed Phase 2.

Global Online Change Status

QUERY MEMBER TYPE(IMS) SHOW(ALL)

- Displays current OLC status of each IMS

```
Response for: QUERY MEMBER TYPE(IMS) SHOW(ALL)

MbrName    CC TYPE  STATUS           LclAttr LclStat
IMS1        0 IMS   OLCPREPC,OLCMSTR
IMS1        0 IMS                    GBLOLC  OLCCMT1C
IMS2        0 IMS                    GBLOLC  OLCCMT1C
IMS3        0 IMS                    GBLOLC  OLCPREPC
IMS4        0 IMS                    LCLOLC
```

/DIS MODIFY shows local status

- OLC libraries
- Work in progress
- Resources to be added, changed, and deleted

IMS

If any process seems to be taking a long time, or to be sure that the prepare phase has completed, a new command can display the current status of each member:

QUERY MEMBER TYPE(IMS) SHOW(ALL)

In this example, there are 3 IMSs participating in G-OLC. IMS4 is using L-OLC. IMS1 and IMS2 have successfully completed Commit Phase 1 (OLCCMT1C). IMS3 is still in Commit Phase 1 (OLCCMT1I). There are other "statuses" which indicate each IMS's progress in the other phases. For example, OLCPREPC for each IMS tells the operator that each IMS has successfully completed PREPARE and that it is OK to enter the COMMIT command.

If a problem occurs on one IMS (for example, it is "stuck" in the prepare phase), the classic /DISPLAY MODIFY command can be used on that IMS to see what the problem might be.

Note that the status is reported for both global and local online change. If IMS4 were in the process of local OLC, its status would be reported.

CSL Highlights

Dynamic LE runtime options

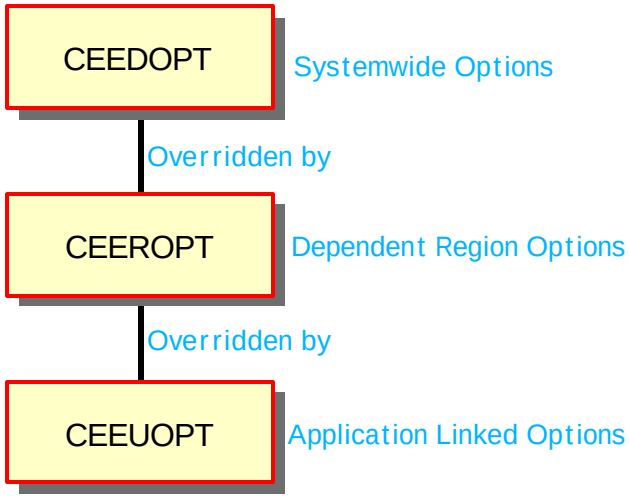
Automatic RECON loss notification

IMS

This section addresses two more CSL capabilities: the ability to dynamically change LE runtime options, and automatic RECON loss notification.

Language Environment RunTime Options

LE runtime options for IMS programs are set at the system, dependent region, or application program level



IMS

LE runtime options can be set at several levels, with each lower level overriding the higher levels. The levels are:

- System-wide options set by CEEDOPT. These options are used unless overridden.
- Dependent region options set by CEEROPT. These options apply to everything that runs in a dependent region and will override any set by CEEDOPT.
- Application options set by CEEUOPT. These options are compiled and link-edited with the application program and will override any options set by CEEDOPT or CEEROPT.

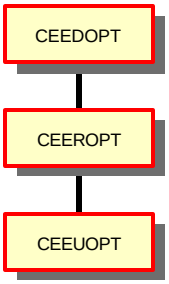
LE RunTime Options ...

LE run-time options sometimes need to be changed

- To collect problem identification information
  - Produce a dump
  - Collect trace data
  - Invoke a debug tool
- To change the storage options for a transaction

Adding or changing LE run-time options prior to IMS V8 requires one or more of the following

- Recompile and relink the LE modules used to supply run-time options (CEEDOPT)
- Stop and restart the dependent region with new/changed run-time options (CEEROPT)
- Recompile and relink the application containing run-time options (CEEUPOT)



IMS

There many different runtime options, and sometimes they may need to be changed temporarily. For example, if you are having problems with a particular program, you might want to change the dump options or turn on tracing, or invoke the debugging tool.

Changing this options may require recompiling and relinking the LE modules used to supply the system-wide optons, or stopping and starting the dependent region, or compiling and relinking the application program..

LE Dynamic Run-Time Options

IMS V8 provides the ability to dynamically update LE run-time options

- Eliminates the need to
  - Stop/start dependent regions
  - Recompile and relink application programs or LE modules

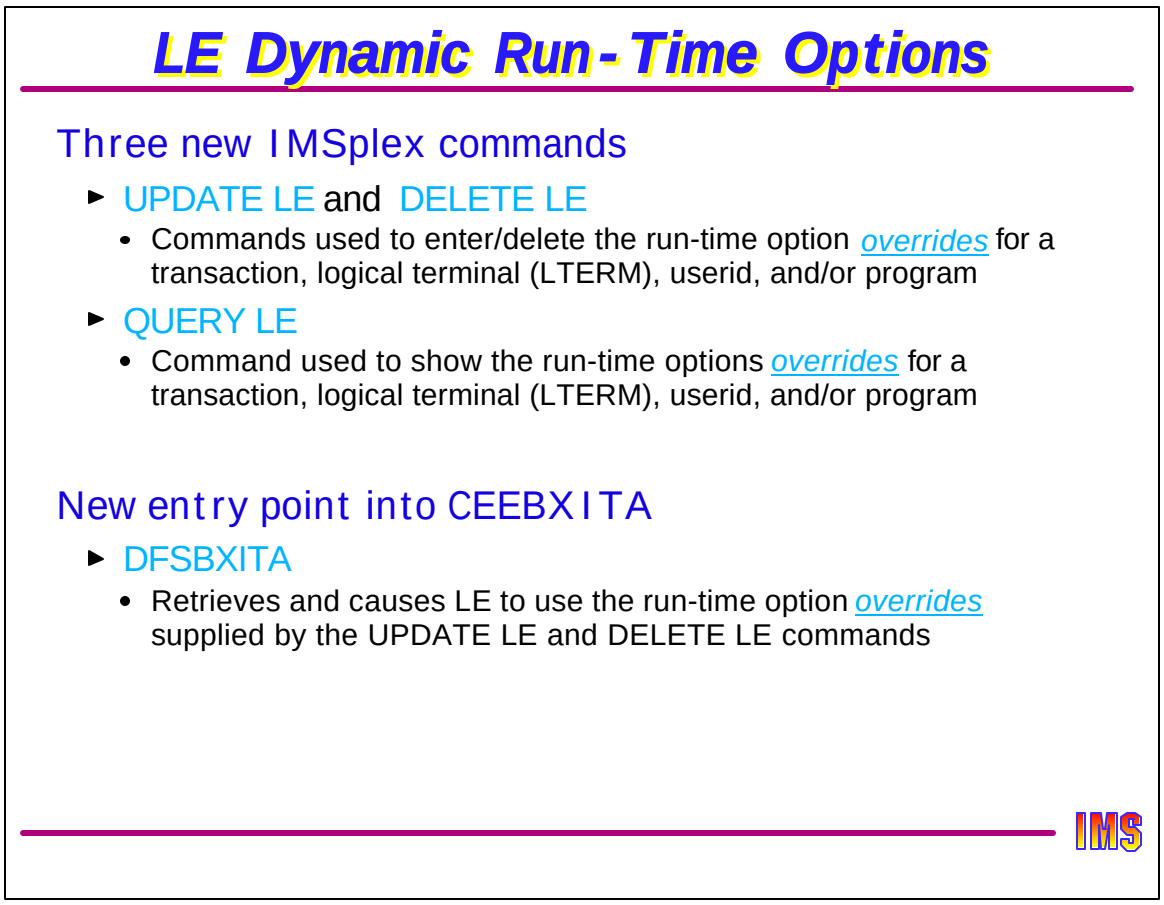
IMS V8 dynamic run-time option support

- Is enabled or disabled by either of the following
  - DFSCGxxx parameter: LEOPT=Y|N
  - IMSplex command: UPD LE SET(LEOPT(YES|NO)
- Requires IMSplex configuration with CSL
  - TSO SPOC to enter command
  - OM to route UPD LE command to IMS

IMS

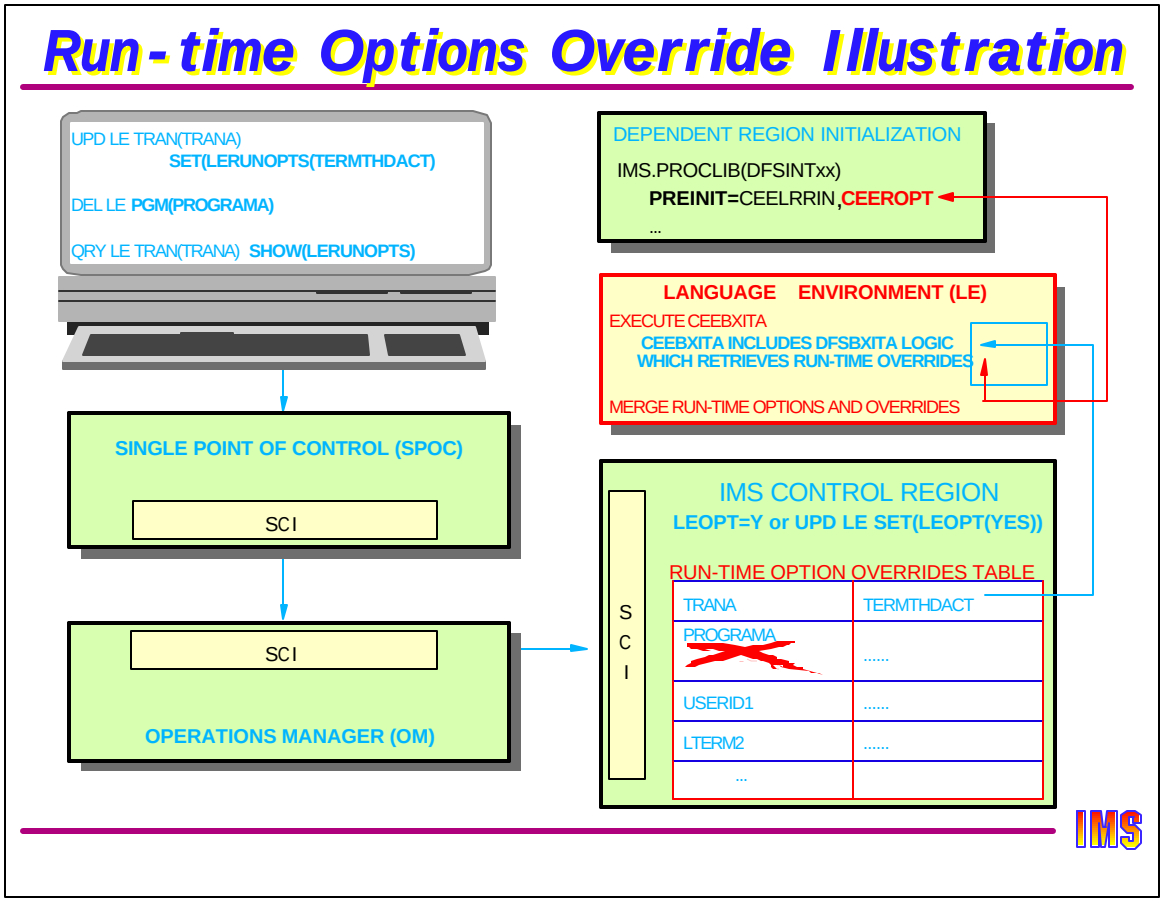
Since none of these are particularly attractive, especially if the change is for a short time only, IMS V8 provides a set of commands to dynamically override all of the options.

It can be enabled in one of two ways:
• Code LEOPT=Y in DFSCGxxx
• Enter the UPD LE SET(LEOPT(YES)) command from a SPOC



There are three commands relate to LE:
- UPDATE LE - used to override the existing LE runtime options. These options can be specified at the lterm, program, transaction, or userid level (called a filter). For example, the override could apply only to transactions entered from LTERM1, or apply only to TRANX.
- DELETE LE - used to delete dynamic LE options set by the UPDATE command. It cannot delete higher level options, such as CEEROPT.
- QUERY LE - displays the LE runtime option overrides currently in effect. Note that this does not display all options - only those previously set by the UPDATE command.

Implementing these requires that a new CSECT be added to the CEEBXITA exit. This CSECT is called DFSBXITA and, in simple terms, builds a table to be used with the "regular" options table to contain the overrides. This exit will be invoked when a program is scheduled into a dependent region. Once program scheduling completes, the options remain in effect for all subsequent transactions processed during that PSB schedule - regardless of which userid or lterm the subsequent transactions came from.



This diagram follows the logic (at a high level) when the TERMTHDACT option is set for TRANA. IMS must be enabled for this capability by having LEOPT=Y or setting it on with the UPD LE command.

When the command gets to IMS, a "Runtime Option Overrides Table" is updated with the new/changed options. In this case, TRANA is updated for TERMTHDACT. When TRANA is scheduled into any dependent region, DFSBXITA will be driven and the LE option set for that transaction.

Automatic RECON Loss Notification

RECON reconfiguration with previous IMS Releases

- When IMS subsystem detects bad RECON, it begins reconfiguration process
  - Copies good RECON to spare
  - IMS V7 writes message identifying subsystems with RECONs open
- To create new spaer bad RECON must be deleted and redefined
  - Deletion requires deallocation by each subsystem
    - Subsystem's reconfiguration process deallocates bad RECON
    - Reconfiguration done when discovered at next RECON access

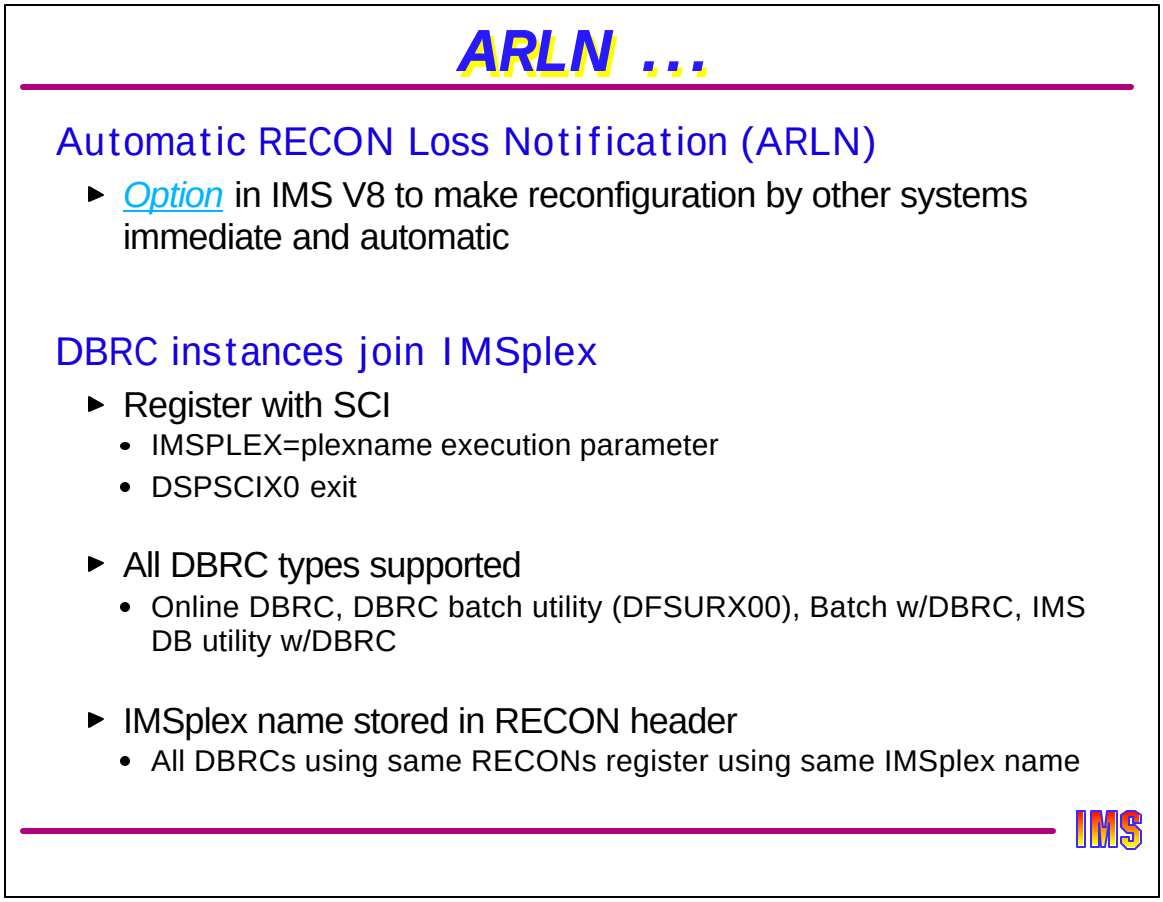
IMS

When an I/O error on a RECON or a CHANGE.RECON REPLACE command is issued, DBRC begins a reconfiguration process. This is true for all releases of IMS. The loss of a RECON causes the remaining good RECON to be copied to the spare. This makes the spare a member of the good RECON pair. It also leaves the RECONs without a spare.

IMS V7 added the DSP0388I message. It is sent by the system which begins the RECON reconfiguration process. The message identifies the subsystems using the RECONs. It is intended to be used by operators or automation processes. The message is of the form:

DSP0388I nnnn SSYS RECORD(S) IN THE RECON AT RECONFIGURATION
DSP0388I SSID=ssidname FOUND

The installation needs to create a new spare so that it may handle any failure of a member of the new pair. The bad RECON data set name is used for the new spare. The bad RECON must be deallocated from all subsystems which were using it before it may be deleted and redefined. In previous releases each subsystem deallocates the bad RECON the next time it attempts to access the RECONs. At that time, it will discover the change in the RECONs and reconfigure. This may take a long time for batch jobs or utilities which use DBRC.

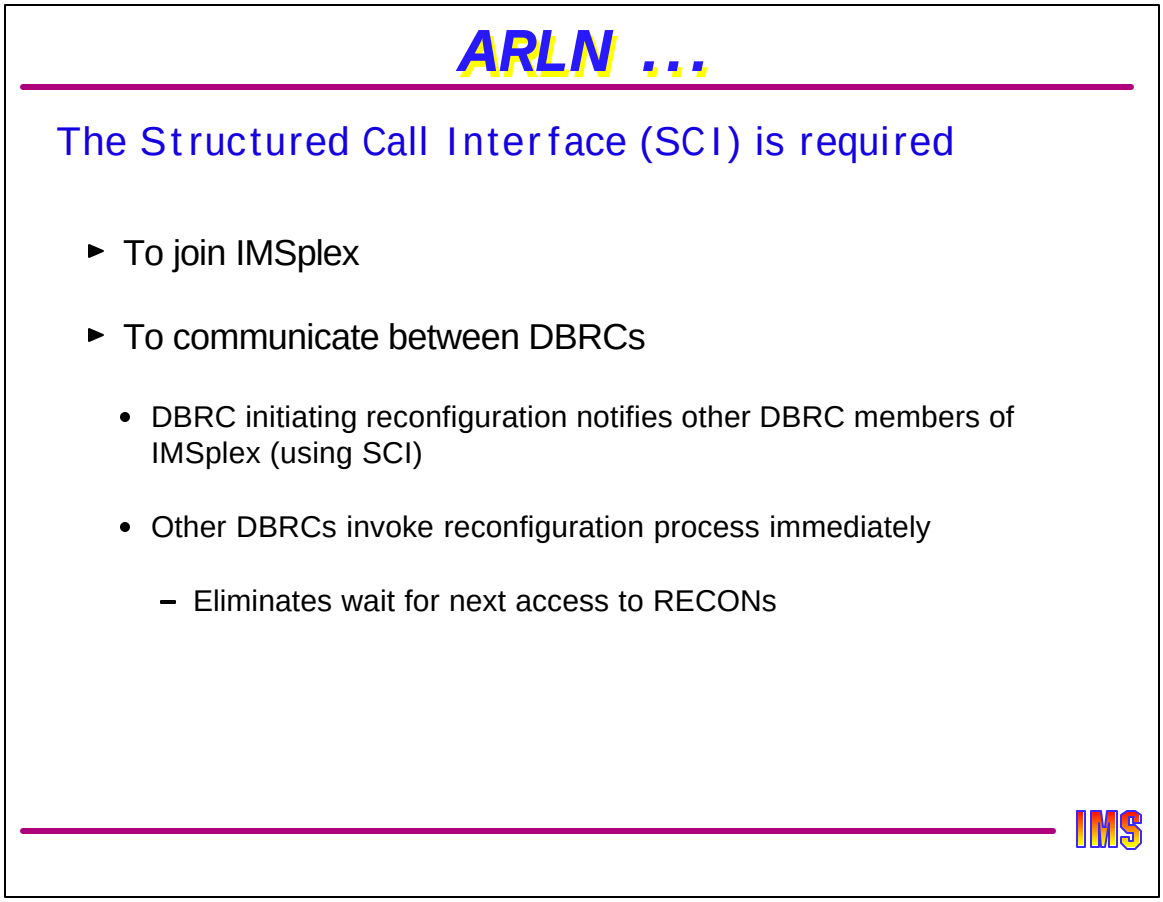


ARLN is optional. It requires DBRC registration with SCI. SCI is used to communicate between the DBRCs. The communication is used to inform other DBRCs that one has done a reconfiguration of the RECONs.

The benefit of ARLN is the elimination of the wait for the other DBRCs to do their reconfiguration. This means that a new spare may be created much sooner.

When DBRC registers with SCI, ARLN is available. Registration may be enabled by coding the IMSPLEX name as an execution parameter for DBRC, or by writing an exit (DSPSCIX0) which can be used by all programs and which can invoke registration with SCI, eliminating the need to change all the JCL.

The IMSplex name is stored in the RECON header, so if a DBRC tries to register with the wrong set of RECONs, it will fail.



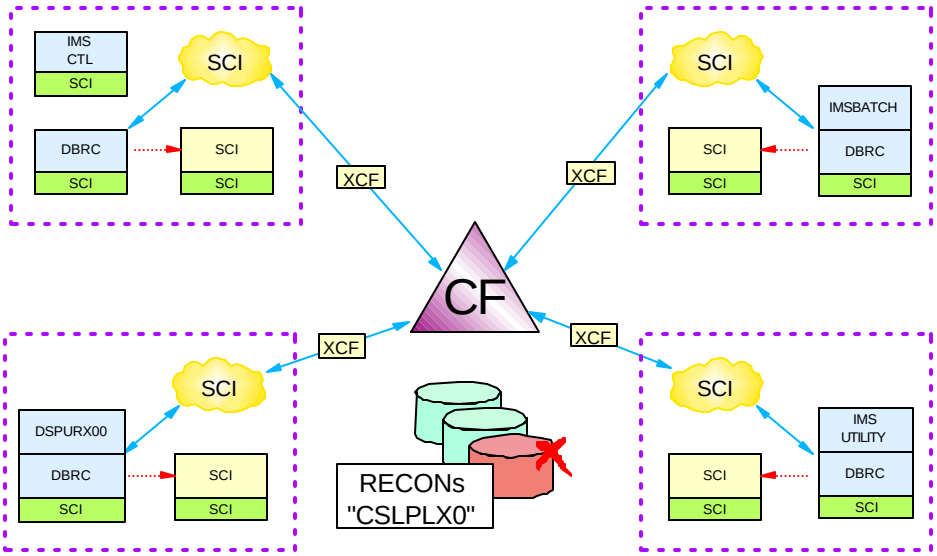
SCI registration is required by DBRC for ARLN. SCI knows the names of all the DBRC address spaces which have registered and can be used by the first DBRC to notify all the others through SCI. A message can be sent to all type DBRC address spaces.

However, the rest of the CSL address spaces are NOT required if this is the only function desired. That is, ARLN can be used without all the other capabilities offered by RM and OM.

ARLN ...

DBRC with SCI

- Only DBRC needs to register with SCI



IMS

This is an illustration of the use of SCI by DBRC. Each DBRC instance in an IMSplex shares the same set of RECONs. The IMSplex name is stored in the RECONs. Each DBRC instance registers with SCI. It uses the IMSplex name for the registration. SCI is used for communication between the DBRCs in an IMSplex. SCI uses XCF for communication between different LPARs.

This illustration shows the IMS Control Regions using SCI. This is not required for ARLN.

IMS V8 Part III Review

CSL

- ★ Summary
- ★ Migration

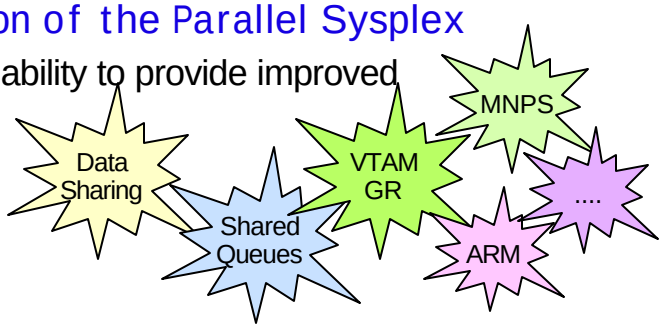
IMS

That's it folks. Time to summarize and talk a little about migrating to this great new function.

IMS in a Parallel Sysplex

IMS's exploitation of the Parallel Sysplex

- Increased IMS's ability to provide improved
  - Capacity
  - Performance
  - Availability
  - but ...



- Added to the complexity of managing IMS systems in the parallel sysplex
  - Resource Management
    - Managing resource activity across multiple IMSs
    - Restoring end-used status when switching IMSs
  - Operations Management
    - Controlling the operations of multiple IMSs from a single entry point

NODE
USER
USERID
LTERM
/START
/STOP
/MODIFY
/DISPLAY

IMS

IMS's exploitation of parallel sysplex technology has provided great benefits in performance, capacity, and availability, but t the same time, made management of the resources involved much more complex.

IMS V8 begins the process of IMSplex systems management by offering both resource management and operations management enhancements.

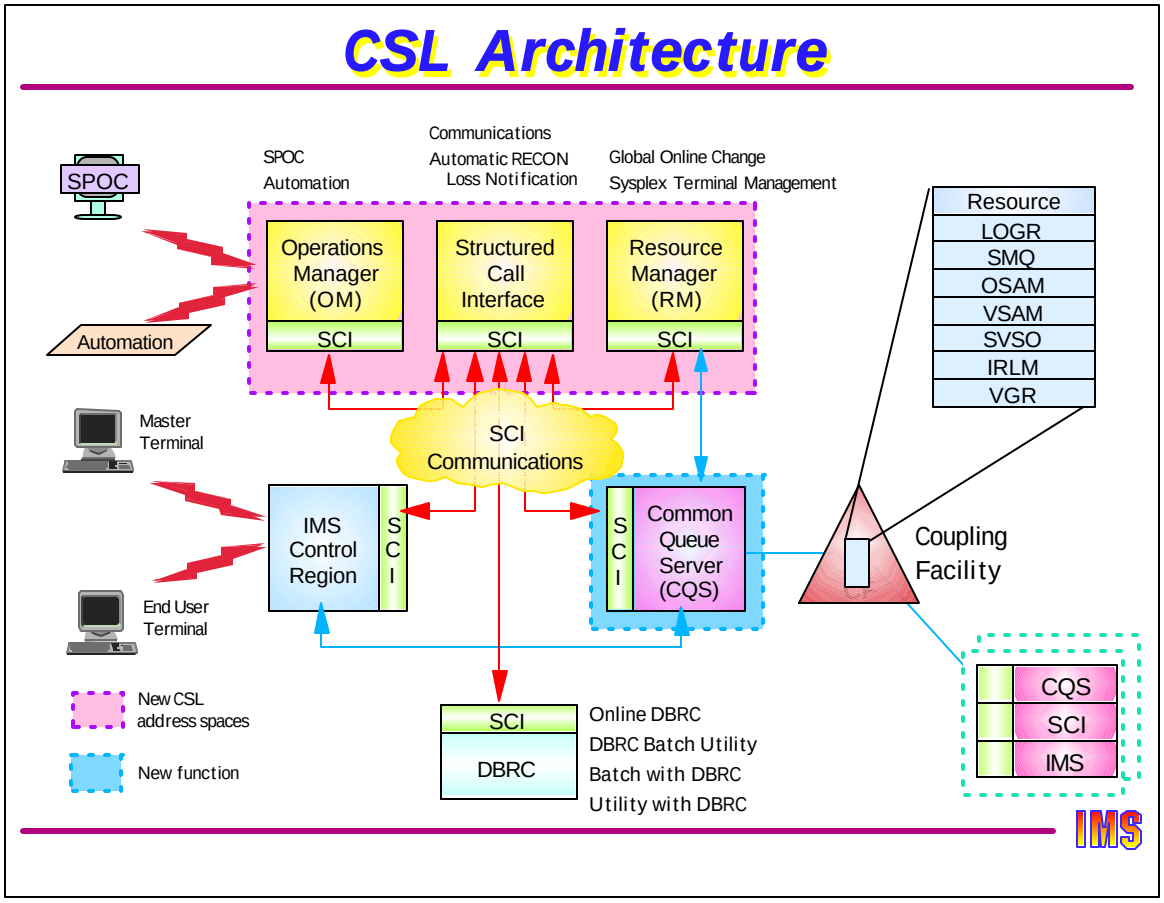
Systems Management ...

IMS V8 targets systems management functions of IMS in a parallel sysplex (IMSplex)

- Common Service Layer is part of an evolving IMSplex architecture
  - Required to take advantage of new systems management functions
- Base Primitive Environment
  - Enhanced to support new architecture
  - Basis for all new CSL address spaces
- New address spaces provides services to IMS clients
  - Structured Call Interface
  - Operations Manager
  - Resource Manager
- New function for Common Queue Server address space
  - Manage new CF Resource Structure to hold resource status information

IMS

The architecture of IMS in the paralle sysplex has evolved to include a Common Service Layer - required to take advantage of these new systems management functions. CSL consists of three new address spaces, built on the Base Primitive Environment (BPE) introduced in IMS V6. CQS has been enhanced to support not only shared queue structures, but also the IMS V8 resource structure.



This is a diagram of the CSL architecture showing the new address spaces and the new resource structure. There may be (probably are) additional images on other LPARs. All CSL address spaces communicate using SCI.

Operations Management

IMS exploits CSL services to enhance

Operations Management

- Operations Manager services
  - OM provides new command entry API (OM API)
    - Enables programmable command entry
    - Consolidates responses from multiple command processors (IMS)

INIT OLC

UPD LE

- IMS supports new IMSplex commands
  - Entered through OM API
  - INITiate, TERMinate, UPDate, DELete, QueRY

QRY STRUCTURE

- TSO Single Point of Control (SPOC)
  - Uses OM API to enter commands retrieve consolidated responses
    - Classic or IMSplex commands

/DIS TRAN

  - Displays responses on TSO terminal

QRY TRAN

- User/vendor may write own AOP using OM API

IMS

The Operations Manager supports the operations management function by providing an interface into the IMSplex from a single point of control to all members of the IMSplex. New command have been added, and a TSO SPOC application provided to get you started. It is expected that the DB2 Control Center tool will offer a GUI interface to OM and support management of both IMS and DB2 from the same SPOC.

Resource Management

IMS exploits CSL services to enhance

Resource Management

- Resource Manager services
  - Uses CQS to maintain resource name and status information in CF list structure
- Sysplex Terminal Management function of IMS exploits RM with resource structure to provide ...
  - Resource name and type consistency
  - Resource name uniqueness
  - Terminal and user status recovery
  - Support for VTAM generic resource session level affinities
  - Support for global callable services
- Coordinated Global Online Change function of IMS exploits RM with or without resource structure to ...
  - Ensure all IMSs use same OLC data sets (requires structure)
  - Coordinate prepare and commit phases across all participating IMSs

IMS

The Resource Manager provides the infrastructure for resource management, to include Sysplex Terminal Management and Global Online Change.

Systems Management

The IMSplex with a Common Service Layer provides the benefits of Parallel Sysplex

- Capacity
- Performance
- Availability

... and enhances the system manageability of the IMSplex

- Resource management
- Operations management

IMS

So there you have it - improved capacity, performance, and availability along with improved systems management.