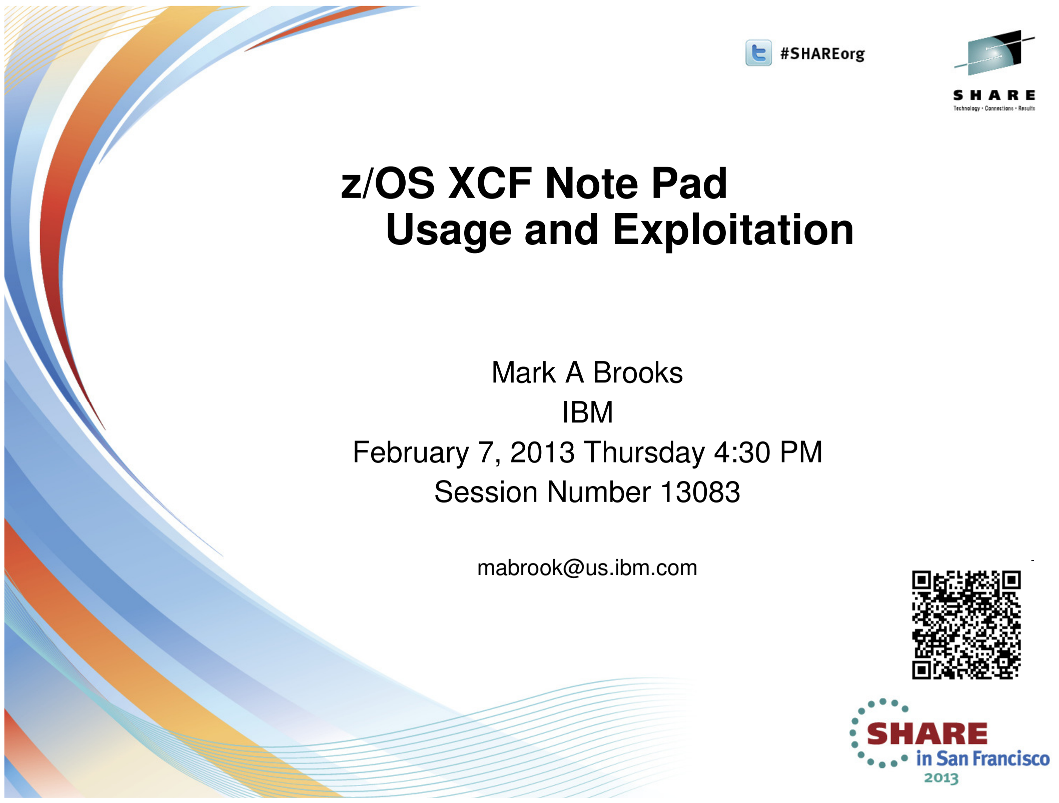


#SHAREorg

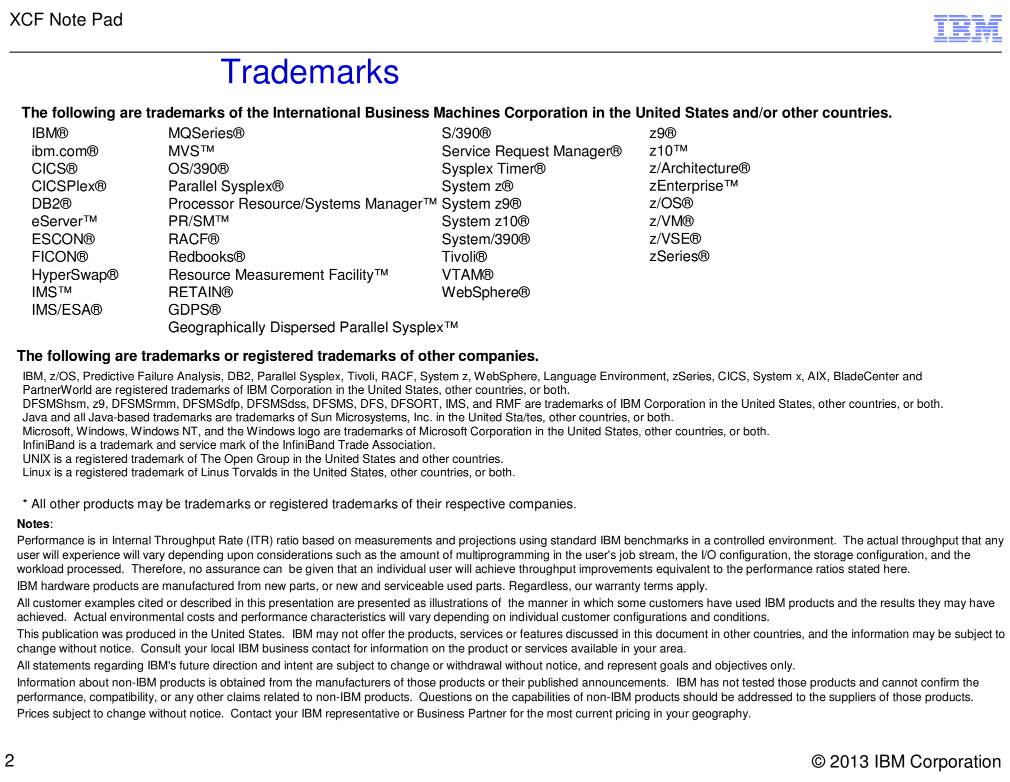
z/OS XCF Note Pad Usage and Exploitation

Mark A Brooks
IBM
February 7, 2013 Thursday 4:30 PM
Session Number 13083

mabrook@us.ibm.com

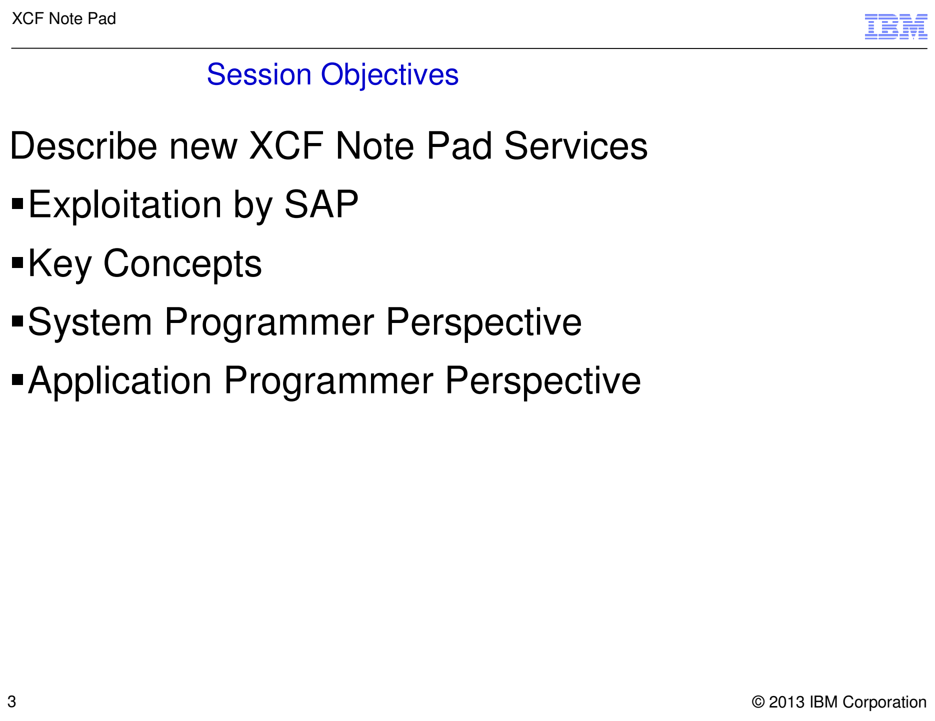
SHARE in San Francisco 2013

We are providing a new XCF programming interface via APAR OA38450 on z/OS V1R13 that will be exploited by SAP. The speaker will focus on two things: the XCF programming interface itself, and the various things that the system programmer needs to do to configure the system for note pads (security, CFRM policy, structure sizings and the like). The speaker will also explain SAP exploitation to provide an example of its use as well as explain the benefits to customers of running SAP with this support enabled. The programming interface supports unauthorized callers and it gives applications virtually painless access to a CF list structure (though the services offered by the note pad programming model are not nearly as extensive as the traditional XES list structure interfaces).



Trademarks

For a complete list of IBM Trademarks, see www.ibm.com/legal/copytrade.shtml



XCF Note Pad

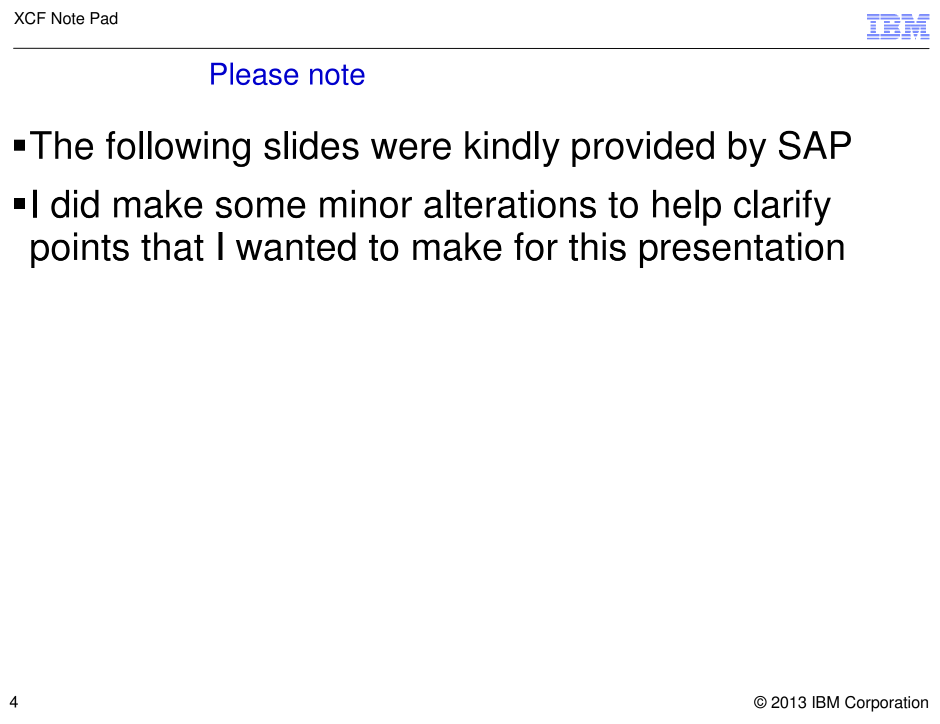
Session Objectives

Describe new XCF Note Pad Services

- Exploitation by SAP
- Key Concepts
- System Programmer Perspective
- Application Programmer Perspective

3 © 2013 IBM Corporation

Our intent is to provide an overview of the XCF Note Pad Services. To whet your appetite, we first describe how SAP makes use of an XCF Note Pad. With that example in mind, we then discuss some of the high level concepts in order to establish terminology and a foundation for the remainder of the presentation. We can then look at note pads from the perspective of the system programmer who needs to configure the sysplex to enable their use. Finally, we briefly discuss XCF Note Pads from the perspective of the application programmer who might like to exploit the service.



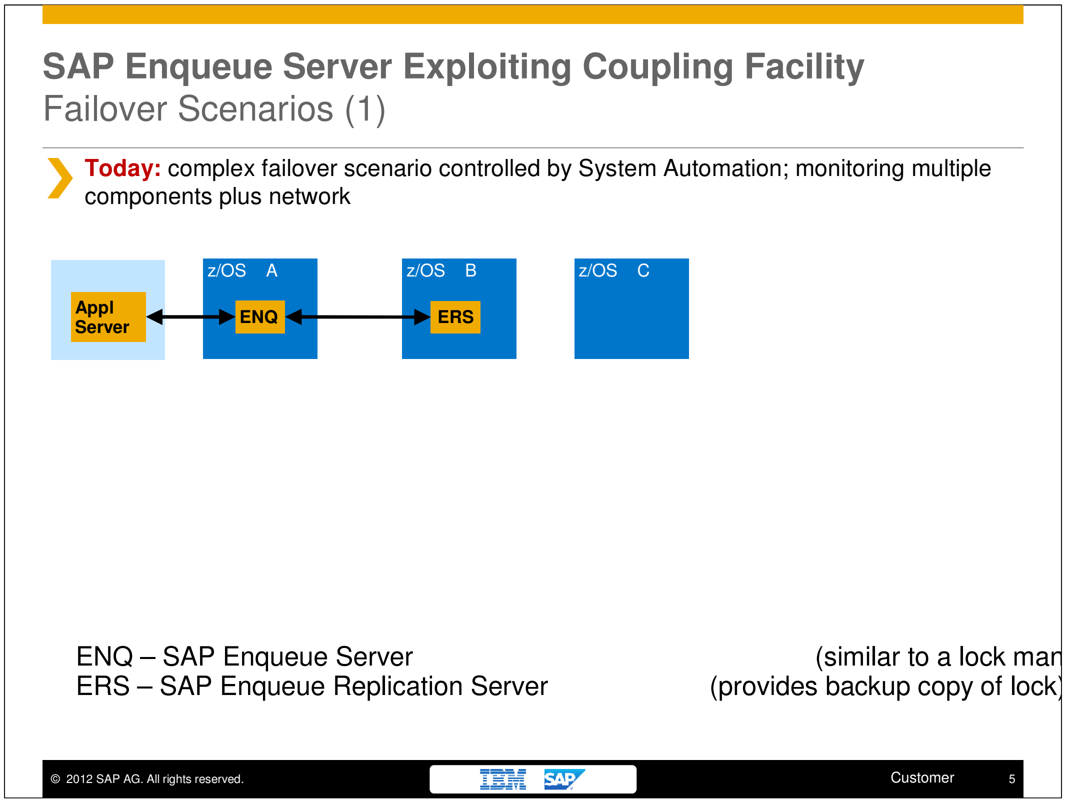
XCF Note Pad

Please note

- The following slides were kindly provided by SAP
- I did make some minor alterations to help clarify points that I wanted to make for this presentation

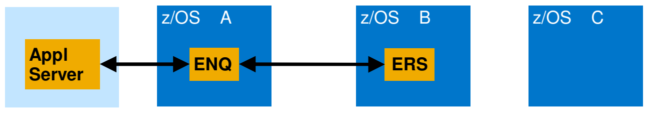
4

It should also be noted that I created the speaker notes for these slides. The notes express my interpretation of the slides. I am not an SAP expert, so there may well be some content that is not entirely accurate from the perspective of one who is an SAP expert. Nothing in the notes should be construed as any sort of statement or commitment by SAP.



SAP Enqueue Server Exploiting Coupling Facility
Failover Scenarios (1)

❯ **Today:** complex failover scenario controlled by System Automation; monitoring multiple components plus network

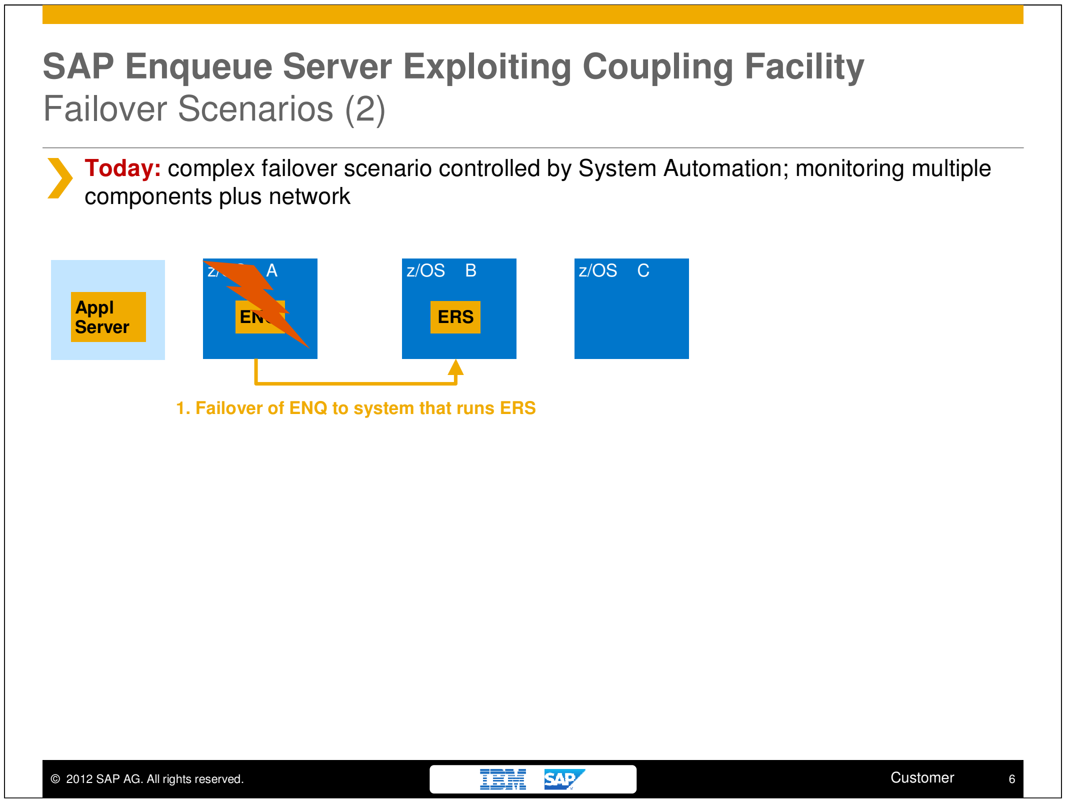


ENQ – SAP Enqueue Server (similar to a lock man
ERS – SAP Enqueue Replication Server (provides backup copy of lock)

© 2012 SAP AG. All rights reserved. Customer 5

The Application Server sends a request to the SAP Enqueue Server (ENQ) to obtain a lock to serialize some resource. ENQ receives the request and obtains the lock. The "lock held" state is sent to the SAP Enqueue Replication Server (ERS) to store information that will enable the lock to be recovered if ENQ should happen to fail. ERS sends a response to ENQ to confirm that the state information has been preserved. After receiving the confirmation, ENQ responds to the application server.
Note that the application server cannot continue until ENQ responds. ENQ does not respond until ERS responds. So use of ERS can elongate transaction response time. Installations that configure an ERS for better availability will generally have longer running transactions and less transaction throughput.

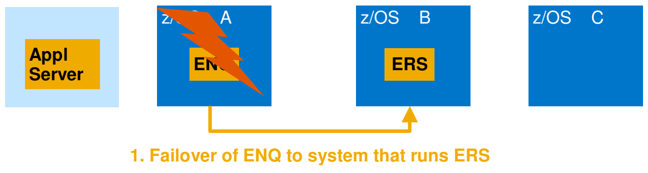
So that is our background. What happens if ENQ fails?



SAP Enqueue Server Exploiting Coupling Facility

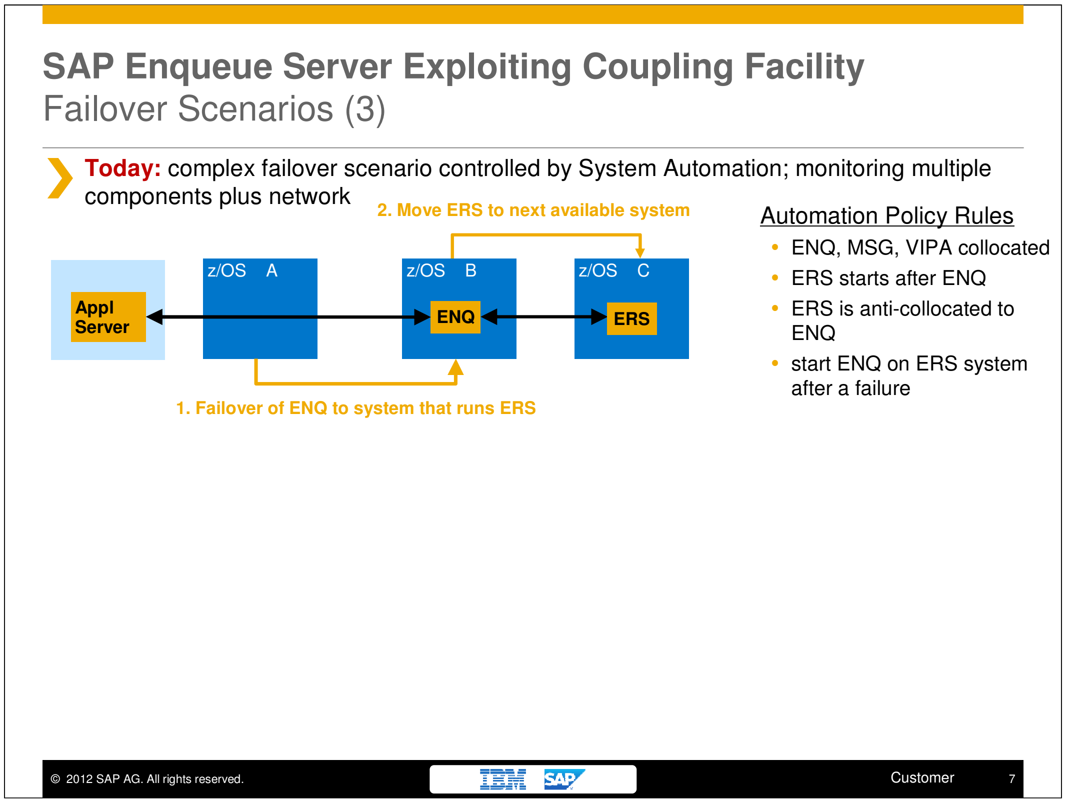
Failover Scenarios (2)

❯ **Today:** complex failover scenario controlled by System Automation; monitoring multiple components plus network



Customer 6

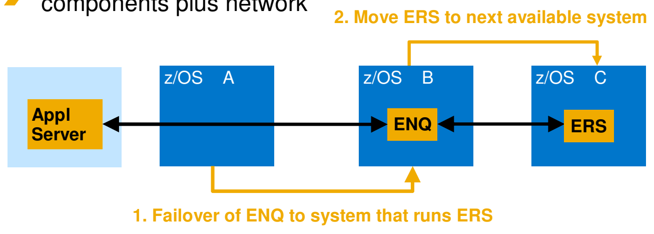
We envision a scenario where the SAP Enqueue Server (ENQ) fails. ENQ must be restarted on the system that contains the SAP Enqueue Replication Server (ERS). ERS has the backup copy of the state information that is needed to allow ENQ to re-establish the "lock held" states for all the Application Servers.



SAP Enqueue Server Exploiting Coupling Facility

Failover Scenarios (3)

Today: complex failover scenario controlled by System Automation; monitoring multiple components plus network


2. Move ERS to next available system
z/OS A
z/OS B
z/OS C
Appl Server
ENQ
ERS
1. Failover of ENQ to system that runs ERS

Automation Policy Rules

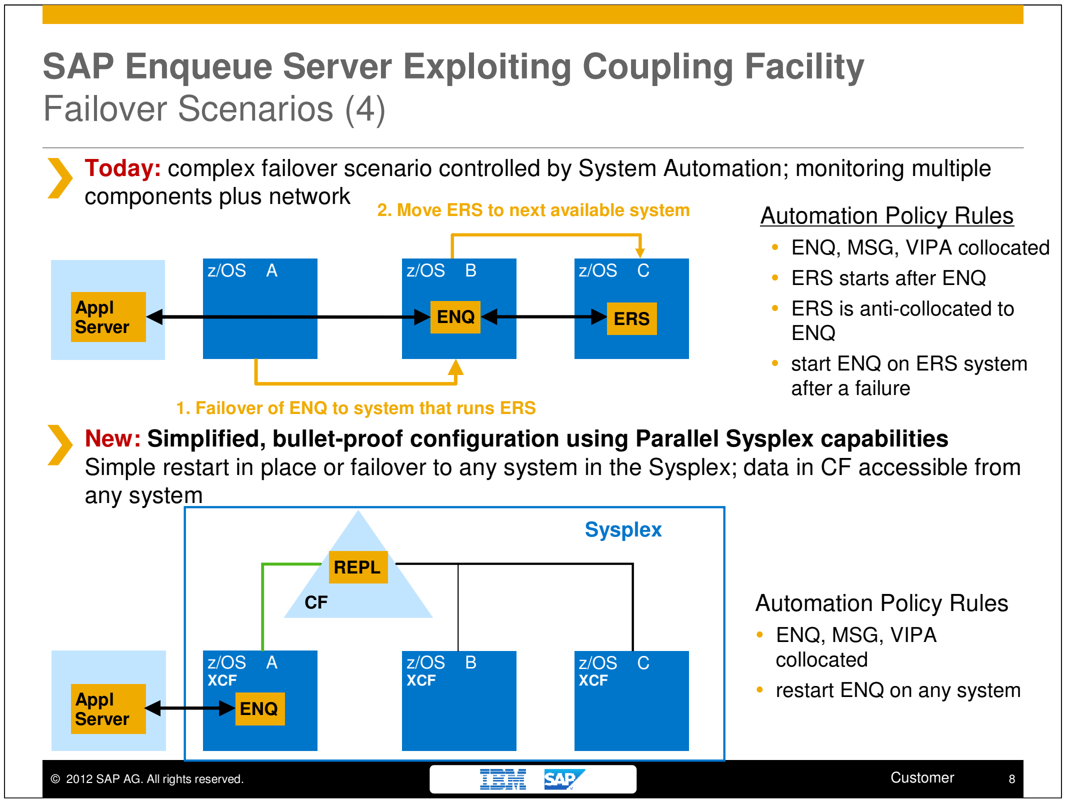
- ENQ, MSG, VIPA collocated
- ERS starts after ENQ
- ERS is anti-collocated to ENQ
- start ENQ on ERS system after a failure

© 2012 SAP AG. All rights reserved. Customer 7

Continuing with recovery, a new instance of the Enqueue Replication Server (ERS) is started on yet another system so as to provide replication services for the newly restarted instance of the Enqueue Server (ENQ). Communication is re-established with the Application Server and normal processing resumes. Once again the Application Server can send requests to ENQ, which are then replicated by ERS.

But this is fairly complex. The automation needs to ensure that ENQ gets restarted on the one system in the sysplex that was running the ERS. The ERS then needs to be moved to yet another system to ensure that the replicated state data will survive another ENQ outage. Finally, the automation then needs to "remember" where everything landed so that ENQ will be restarted on the appropriate system after the next failure.

How might this look if we were to exploit the coupling facility for the replicated state data?



SAP Enqueue Server Exploiting Coupling Facility
Failover Scenarios (4)

- **Today:** complex failover scenario controlled by System Automation; monitoring multiple components plus network

2. Move ERS to next available system
z/OS A
z/OS B
z/OS C
Appl Server
ENQ
ERS
1. Failover of ENQ to system that runs ERS

Automation Policy Rules
- ENQ, MSG, VIPA collocated
- ERS starts after ENQ
- ERS is anti-collocated to ENQ
- start ENQ on ERS system after a failure

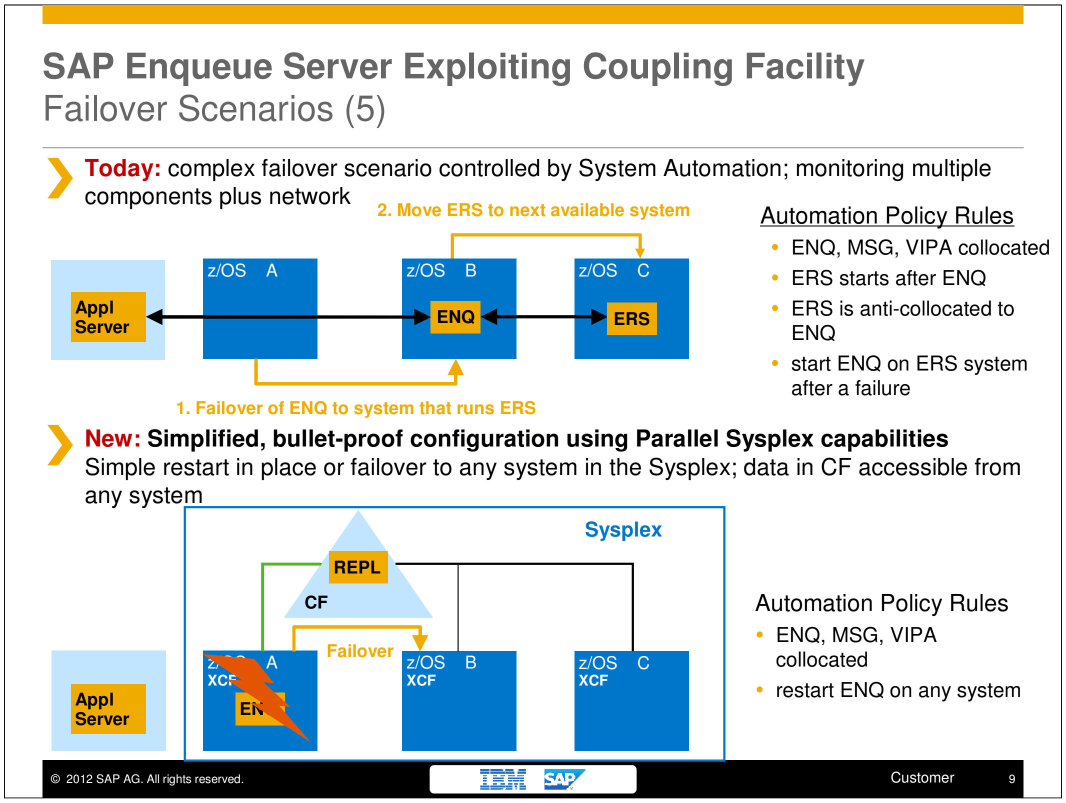
- **New: Simplified, bullet-proof configuration using Parallel Sysplex capabilities**
Simple restart in place or failover to any system in the Sysplex; data in CF accessible from any system

Automation Policy Rules
- ENQ, MSG, VIPA collocated
- restart ENQ on any system

© 2012 SAP AG. All rights reserved. Customer 8

The top half of the chart depicts the state after automation has restored service after the failure of the Enqueue Server (ENQ).
The bottom half of the chart suggests how the coupling facility might be used to store the replication state data. Let us reconsider the mainline transaction in this environment. The Application Server sends a request to the SAP Enqueue Server (ENQ) to obtain a lock to serialize some resource. ENQ receives the request and obtains the lock. The “lock held” state is written to the replication table (REPL) in the coupling facility. This information will enable the lock to be recovered if ENQ should happen to fail. Upon return from the (synchronous) CF write operation, ENQ responds to the application server.
It is still true that the application server cannot continue until ENQ responds. ENQ does not respond until the CF write operation completes. So replication will still elongate transaction response time. However, the time to complete a CF write operation is significantly less than the time it takes to communicate with the traditional Enqueue Replication Server (ERS) via TCP/IP. Thus installations that exploit the coupling facility for replication will have better transaction response times and more transaction throughput than it would with a traditional ERS.
Also note that there is no ERS running on any z/OS image. Thus fewer z/OS MIPS are consumed.

What happens if ENQ fails in this environment?



SAP Enqueue Server Exploiting Coupling Facility

Failover Scenarios (5)

- **Today:** complex failover scenario controlled by System Automation; monitoring multiple components plus network

2. Move ERS to next available system
Appl Server
z/OS A
z/OS B
ENQ
z/OS C
ERS
1. Failover of ENQ to system that runs ERS

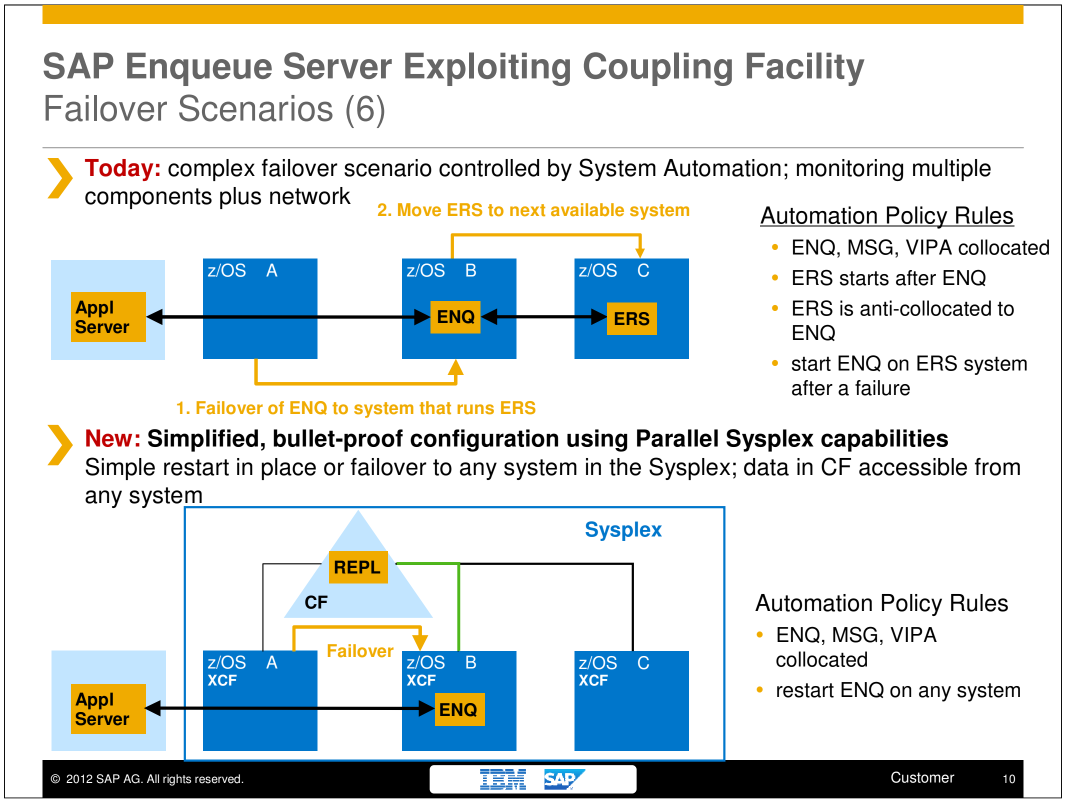
Automation Policy Rules
- ENQ, MSG, VIPA collocated
- ERS starts after ENQ
- ERS is anti-collocated to ENQ
- start ENQ on ERS system after a failure

- **New: Simplified, bullet-proof configuration using Parallel Sysplex capabilities**
  Simple restart in place or failover to any system in the Sysplex; data in CF accessible from any system

Automation Policy Rules
- ENQ, MSG, VIPA collocated
- restart ENQ on any system

© 2012 SAP AG. All rights reserved. Customer 9

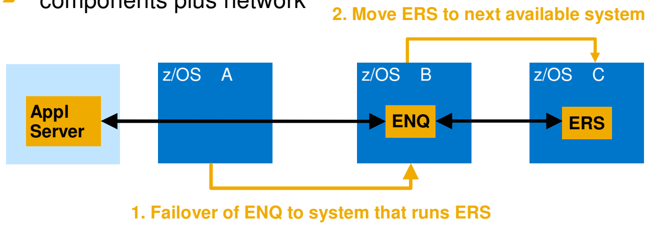
We envision a scenario where the SAP Enqueue Server (ENQ) fails with lock held states being replicated in the coupling facility (CF). ENQ can be restarted on any system in the sysplex that has connectivity to the coupling facility that contains the replication data (REPL). In practical terms, that means ENQ can be restarted on any system in the sysplex. Contrast this with traditional replication via the Enqueue Replication Server (ERS) where ENQ had to be restarted on the one system where ERS was running since the ERS had the backup copy of the state information needed to allow ENQ to re-establish the “lock held” states for all the Application Servers. In the new configuration, the state information needed to allow ENQ to re-establish the”lock held” states for the Application Servers resides in REPL in the coupling facility, which is accessible to any system in the sysplex.



SAP Enqueue Server Exploiting Coupling Facility

Failover Scenarios (6)

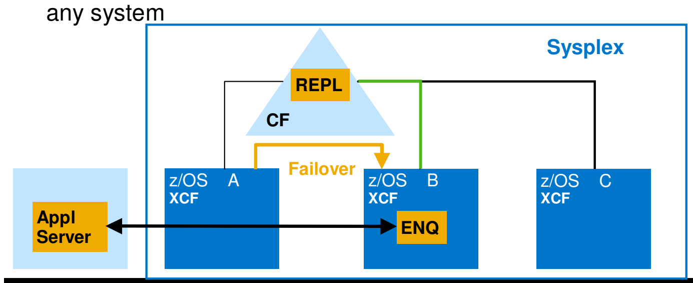
- **Today:** complex failover scenario controlled by System Automation; monitoring multiple components plus network


2. Move ERS to next available system
z/OS A
z/OS B
z/OS C
Appl Server
ENQ
ERS
1. Failover of ENQ to system that runs ERS

Automation Policy Rules

- ENQ, MSG, VIPA collocated
- ERS starts after ENQ
- ERS is anti-collocated to ENQ
- start ENQ on ERS system after a failure

- **New: Simplified, bullet-proof configuration using Parallel Sysplex capabilities**
  Simple restart in place or failover to any system in the Sysplex; data in CF accessible from any system



Automation Policy Rules

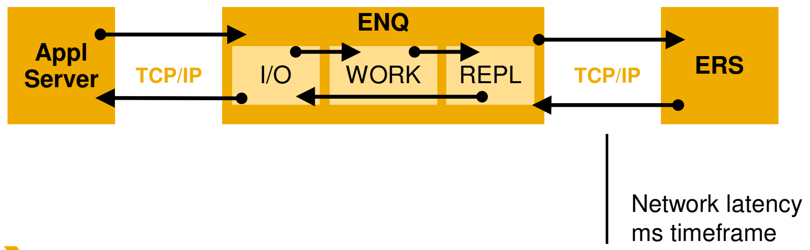
- ENQ, MSG, VIPA collocated
- restart ENQ on any system

© 2012 SAP AG. All rights reserved. Customer 10

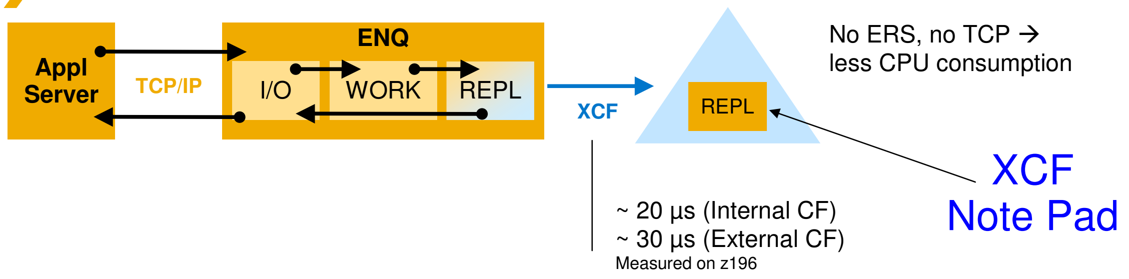
Continuing with recovery, we see that the restarted instance of the Enqueue Server (ENQ) reads the state information from the replication data (REPL) in the coupling facility to re-establish the lock held states for the applications. As soon as communication is re-established with the Application Servers, normal processing can be resumed.
Use of the coupling facility for replication simplifies the recovery process by making it possible to start ENQ on any system in the sysplex.

SAP Enqueue Server Exploiting Coupling Facility
Implementation Details

Using Enqueue Replication Server



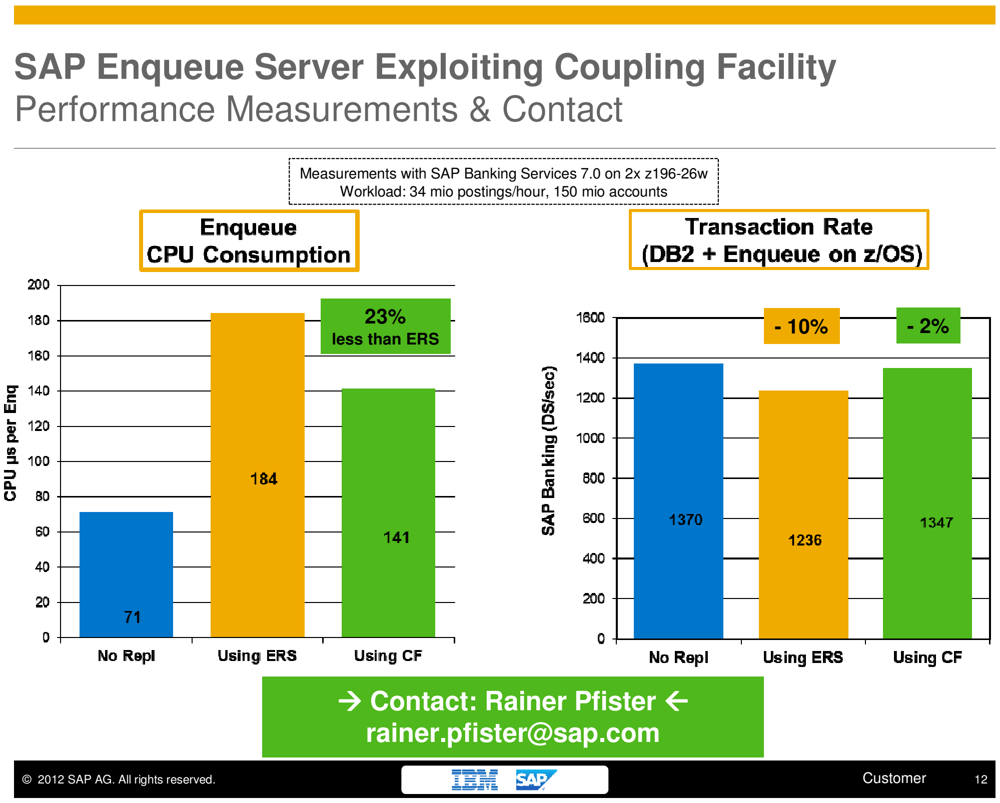
Exploiting Coupling Facility



Available on z/OS 1.13 with SAP 7.21 kernel (downward compatible to 7.00, 7.01, 7.10, 7.11l)

This chart provides a more detailed view of the transaction flow within the Enqueue Server (ENQ).

It also highlights some of the benefits: reduced latency (XCF Note Pad access is faster than communication with ERS via TCP/IP) and less CPU consumption since the z/OS image is not running ERS.



SAP Enqueue Server Exploiting Coupling Facility
Performance Measurements & Contact

Measurements with SAP Banking Services 7.0 on 2x z196-26w
Workload: 34 mio postings/hour, 150 mio accounts

→ Contact: Rainer Pfister ←
rainer.pfister@sap.com

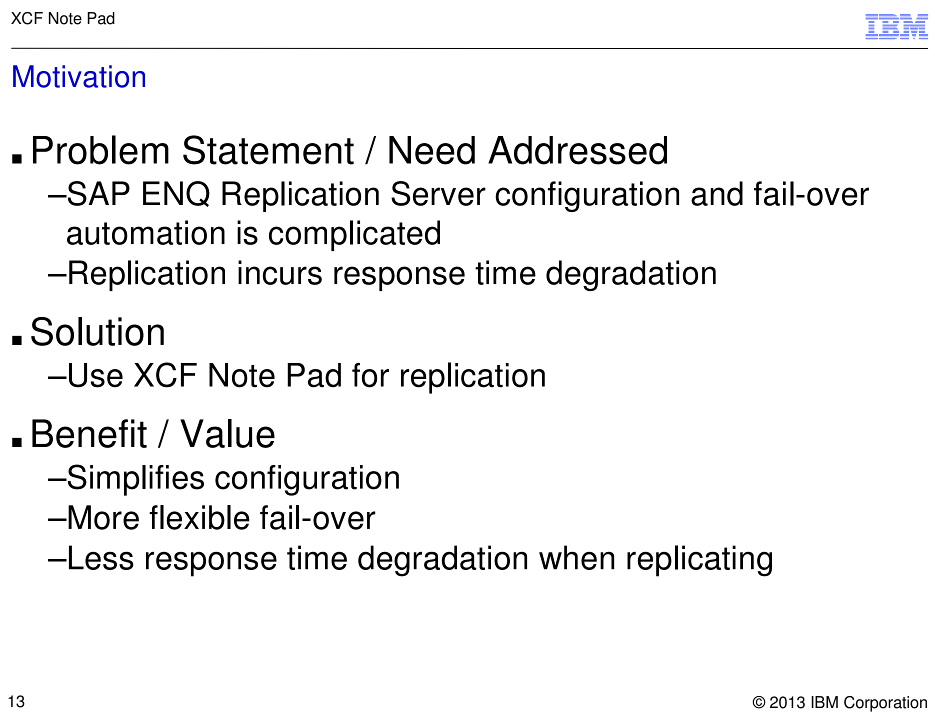
© 2012 SAP AG. All rights reserved. Customer 12

Use of the XCF Note Pad Services provides for less complex failure recovery. But there are other benefits as well. If SAP is configured so that the Enqueue Server (ENQ) uses an XCF Note Pad to provide the Enqueue Replication Server (ERS) functionality, the cost of servicing an ENQ request is reduced as compared to the traditional ERS configuration. Similarly, there is an improvement to transaction throughput when XCF Note Pads are used.

The usual caveats apply. Your results may vary.

Refer to the following paper for a detailed performance study.
IBM Enterprise System: Sap Enqueue Replication on System z Coupling Facility Performance Paper

http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP102248



XCF Note Pad

Motivation

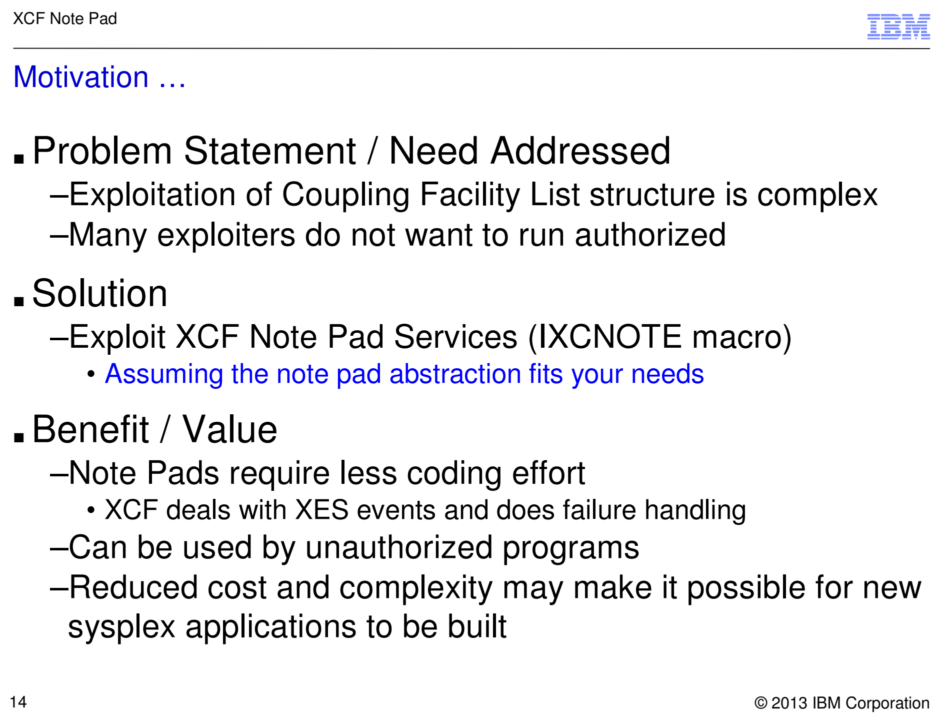
- Problem Statement / Need Addressed
  - –SAP ENQ Replication Server configuration and fail-over automation is complicated
  - –Replication incurs response time degradation
- Solution
  - –Use XCF Note Pad for replication
- Benefit / Value
  - –Simplifies configuration
  - –More flexible fail-over
  - –Less response time degradation when replicating

13 © 2013 IBM Corporation

The immediate goal of the IXCNOTE interface is to provide services that will enable the SAP Enqueue Server to easily exploit a coupling facility for replication. Today, this replication is accomplished by sending requests via TCP/IP to some other system who is then responsible for maintaining a backup copy of an ENQ request (not to be confused with a GRS ENQ). Today, some fairly sophisticated configuration and System Automation setup is required to recognize failures and get a new instance of the server started on the system that has the backup copies when the ENQ server fails. The automation also needs to deal with communication failures, and failure of the replication server.

SAP can now be optionally configured to use the XCF Note Pad Services to replicate their ENQ requests in a CF structure. Doing so will greatly simplify the configuration and management of this high availability option. For example, when the SAP Enqueue server fails, it could be restarted on any system in the sysplex that has access to the coupling facility that contains the note pad that is being used to replicate the ENQ requests. Today, the server needs to be restarted on the very system that was doing the replication. The performance impact of enqueue replication will in general be less. Replication is always performed synchronously to the enqueue request. With replication via TCP/IP, a typical enqueue request has a response time of around 920 microseconds, versus a typical response time of 380 microseconds without replication. When replicated in a CF structure, it is estimated that the response time for an enqueue request with replication will be around 410 microseconds. Finally, this solution provides a nice differentiator for running SAP on System z in that it will enable customers to achieve higher availability with less of a performance penalty for replication relative to the traditional TCP/IP replication.

Note: Response times can vary depending on such factors as configuration and workload. These were our results. Yours may be different.



XCF Note Pad

Motivation …

- Problem Statement / Need Addressed
  - Exploitation of Coupling Facility List structure is complex
  - Many exploiters do not want to run authorized
- Solution
  - Exploit XCF Note Pad Services (IXCNOTE macro)
    - Assuming the note pad abstraction fits your needs
- Benefit / Value
  - Note Pads require less coding effort
    - XCF deals with XES events and does failure handling
  - Can be used by unauthorized programs
  - Reduced cost and complexity may make it possible for new sysplex applications to be built

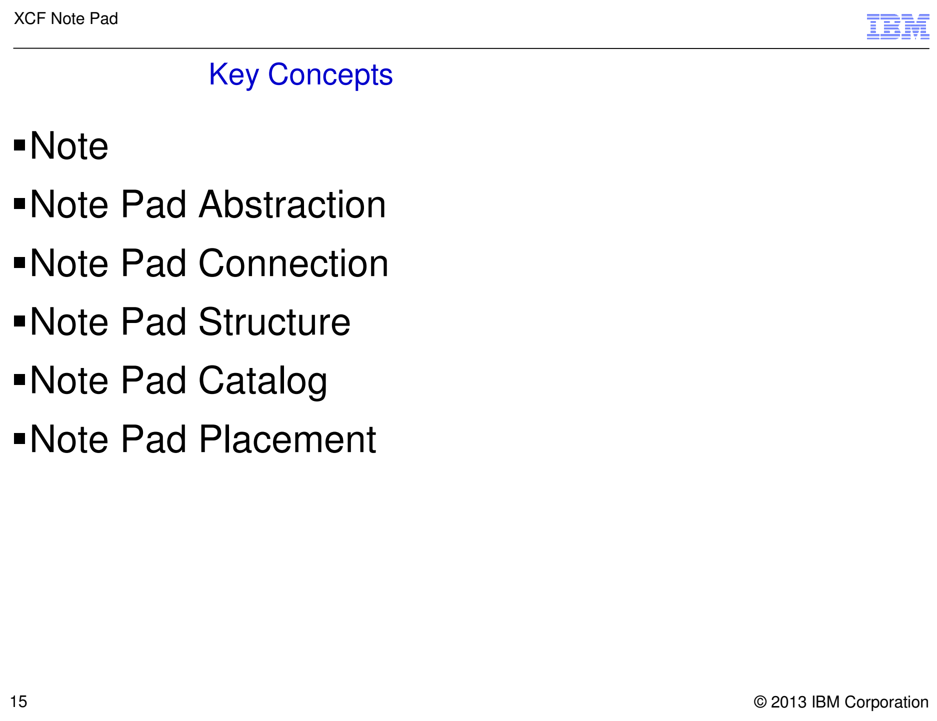
14

© 2013 IBM Corporation

The IXCNOTE interface is intended to provide externals that are much easier to exploit than the IXLLIST services on which they are built. Reducing the cost and complexity of exploiting the CF services may enable additional sysplex applications to be built.

The interface was certainly designed to meet needs of SAP ENQ replication. But it is also sufficiently general that it may have broader appeal. We certainly built it with that idea in mind. It is not really useful as platform for communication as there are no notifications issued when a new note "arrives". However, it should have appeal for applications that need to record state information or data.

The fact that it can be used by unauthorized programs may also help broaden its appeal.



XCF Note Pad

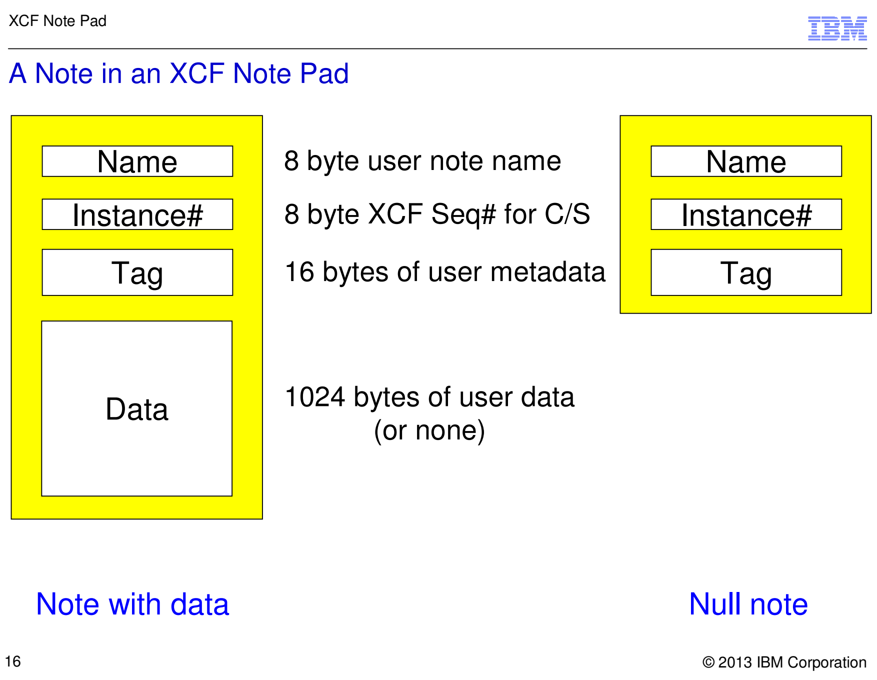
Key Concepts

- Note
- Note Pad Abstraction
- Note Pad Connection
- Note Pad Structure
- Note Pad Catalog
- Note Pad Placement

15

We now want to lay out the key note pad concepts that will be the basis for the remainder of the presentation.

A note is really application data. Notes live in a note pad, which is a named collection of notes. To manipulate notes in a note pad, the application must establish a connection to the note pad. Note pads reside in a note pad structure (which resides in a coupling facility). XCF needs to know whether a given note pad exists and if so, which structure is being used to host the note pad. So there is a note pad catalog to track and manage the note pads. The catalog is just another CF structure. Finally we want to understand how XCF decides which structure should be used to host the note pad.



XCF Note Pad

A Note in an XCF Note Pad

Note with data		Null note
Name	8 byte user note name	Name
Instance#	8 byte XCF Seq# for C/S	Instance#
Tag	16 bytes of user metadata	Tag
Data	1024 bytes of user data (or none)	

16 © 2013 IBM Corporation

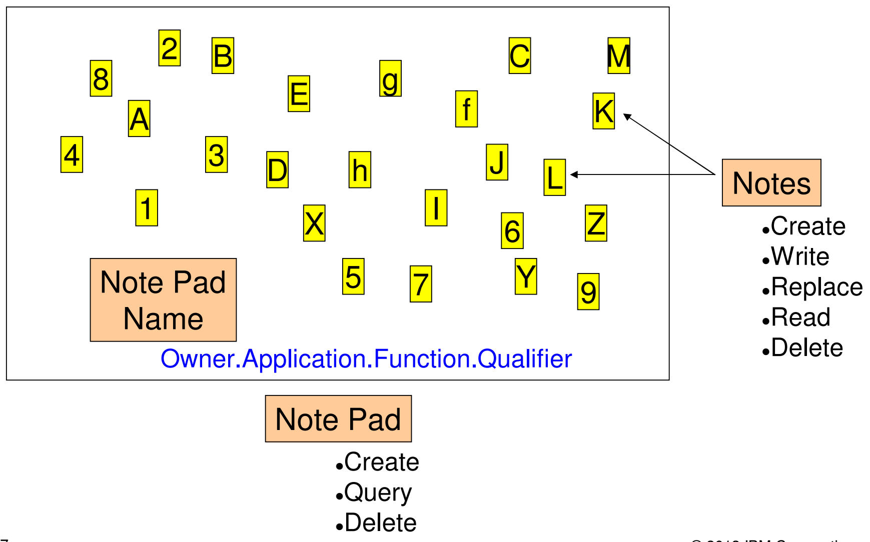
A **note** is a **list entry** containing either no data or 1024 bytes of **entry data**. A note without any data is said to be a **null** note. Each note has an 8 byte **note name** that identifies the note when creating, reading, updating, or deleting the note. Names must be unique within a given note pad. The application picks the names. Two different note pads can use the same note name without conflict. Each note has an **instance number** that is changed every time the note is updated. XCF manipulates the instance number. The exploiter can optionally do instance number comparisons to ensure that updates are being made to the expected instance of the note, but otherwise has no control over what value is used for an instance number.

The note **tag** is 16 bytes of additional data to be associated with the note. The creator of the Note Pad determines whether the tags are assigned by XCF or by the user. For **user assigned tags**, the creator of the XCF Note Pad determines whether the tag values are arbitrary data, or whether the tag values for a given note must be non-decreasing (so long as the note exists). **XCF assigned tags** form an ever increasing sequence that is global to the note pad. The next incremental sequence number is assigned whenever a note is created or updated.

XCF Note Pad

IBM

Abstract View of an XCF Note Pad



17

© 2013 IBM Corporation

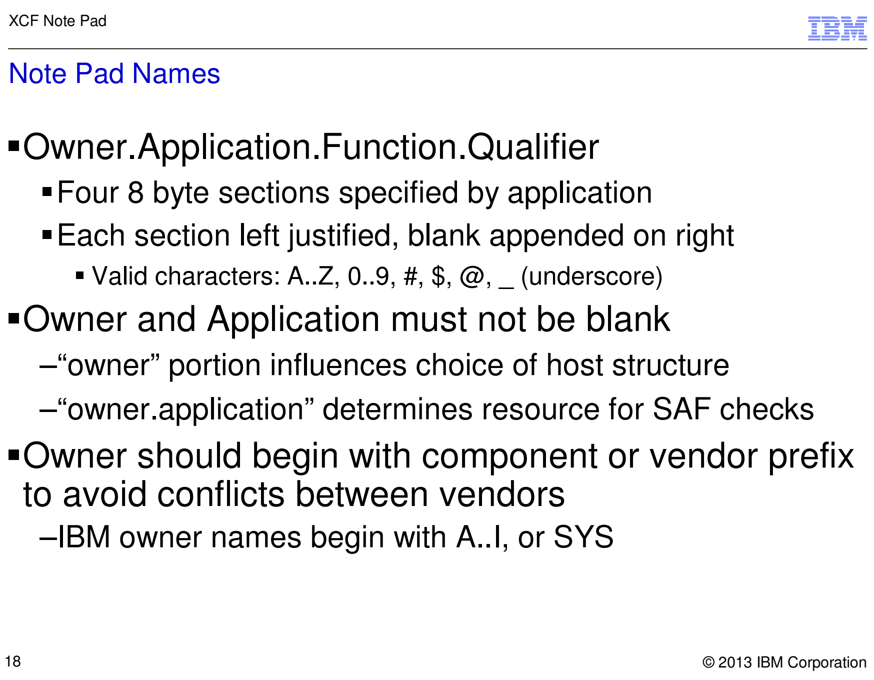
This chart depicts the note pad abstraction. A note pad is a named collection of notes. Each yellow box is a note. Each note has a name, such as “A” or “B” or “9”, and may contain 1024 bytes of data. The names of the notes are determined by the application. The note pad name, which is chosen by the application, has a particular form that is intended to prevent collisions between applications and still allow for a variety of different note pads to be created. The application can create a note pad, query a note pad to get information about the note pad, and delete a note pad. A note in the note pad can be created, or deleted. The note content can be read or replaced. A write request will create the note if it does not already exist, and replace the note content if it does exist.

The XCF Note Pad Services provide an interface that allows an application to store and retrieve data from a coupling facility list structure. The XCF Note Pad macro (IXCNOTE) encapsulates the interfaces that allow an application to interact with the XCF Note Pad Services. With these interfaces, an application can create or delete a "note pad". A note pad contains zero or more notes. A "note" contains up to 1024 bytes of application data and is identified by an application provided "name". An application that is "connected" to a note pad can create, read, modify, or delete notes in the note pad. Depending on the various attributes specified when the note pad is created, one might use a note pad to:

o Store data that would allow a backup instance of an application to resume processing on some other system in the sysplex after the primary instance had failed

o Share data between systems in the sysplex, using a simple "compare and swap" like serialization to ensure the integrity of updates.

Applications that use IXCNOTE need not run authorized.



Note Pad Names

- Owner.Application.Function.Qualifier
  - Four 8 byte sections specified by application
  - Each section left justified, blank appended on right
    - Valid characters: A..Z, 0..9, #, $, @, _ (underscore)
- Owner and Application must not be blank
  - –"owner" portion influences choice of host structure
  - –"owner.application" determines resource for SAF checks
- Owner should begin with component or vendor prefix to avoid conflicts between vendors
  - –IBM owner names begin with A..I, or SYS

Note Pad names consist of four 8 byte sections: "owner", "application", "function", and "qualifier". The intended purpose of the ***owner*** section is to provide uniqueness so that the note pad names used by different software vendors will not conflict with each other. The intended purpose of the ***application*** section is to provide uniqueness so that the note pad names used by different applications from a given software vendor will not conflict with each other. The intended purpose of the ***function*** section is to enable a given vendor application to use multiple note pads. The intended purpose of the ***qualification*** section is to enable a given application function to make use of more than one note pad. Alternatively, this section might be used to distinguish among multiple instances of an application that might be running in the same sysplex. You can think of the sections in any way you like. However, "owner" and "application" are required and must be documented so that the installation can be configured to support your note pad. You need to understand how "owner" and "application" are used by XCF with respect to structure assignment and authority checking (more later).

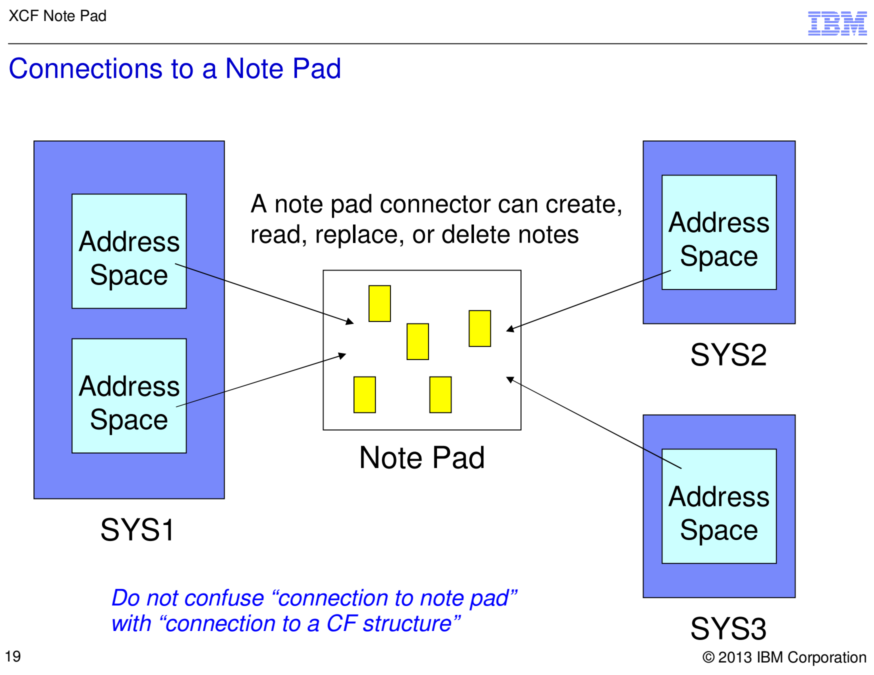
Each 8 byte section must be left justified, padded on the right with EBCDIC blanks as needed. Each section can contain any upper case alphabetic (A-Z), numeric (0-9), national (@,#,$), or underscore (_) character. The first two sections, which identify the owner (vendor) and application, must not be all blanks. The remaining sections can be all blank. Externally, we use "dot qualification" to demarcate the sections of the name for readability.

To avoid names used by IBM, do not begin Note Pad names ("owner") with the letters A through I or the character string SYS.

Note that if the installation so chooses, it can define coupling facility structure names of the form IXCNP_ownerxx (where xx is the EBCDIC representation of a two digit hex number) to have XCF allocate Note Pads for the indicated "owner" in one of the designated structures. In the absence of any such definition, the Note Pad will be allocated in a default Note Pad structure defined for XCF (IXCNP_SYSXCFxx).

To enable a program to access a note pad, the installation defines System Authorization Facility (SAF) profiles based on "owner" and "application".

Note pad names are mapped by ixcynote_tNotePadName in macro IXCYNOTE.



XCF Note Pad

Connections to a Note Pad

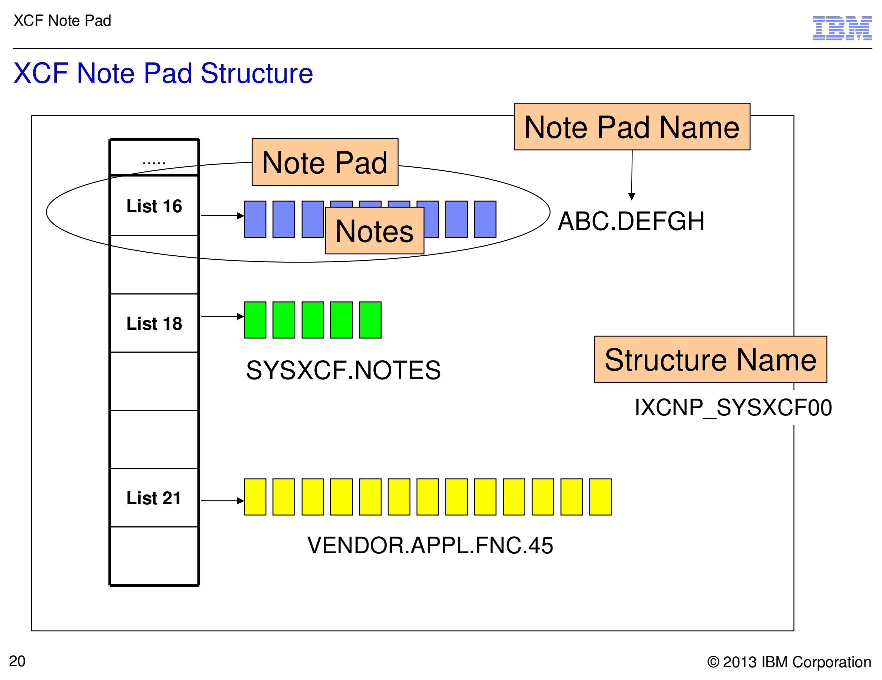
A note pad connector can create, read, replace, or delete notes
Address Space
Address Space
SYS1
Note Pad
Address Space
SYS2
Address Space
SYS3
Do not confuse "connection to note pad" with "connection to a CF structure"
19
© 2013 IBM Corporation

To use an XCF Note Pad, one must "connect" to the Note Pad. Generally, each address space that wants to use the XCF Note Pad will need to establish a connection. One can connect from any system in the sysplex*. Any given system can have multiple connections.

After a connection is created, the connector can manipulate notes in the Note Pad.

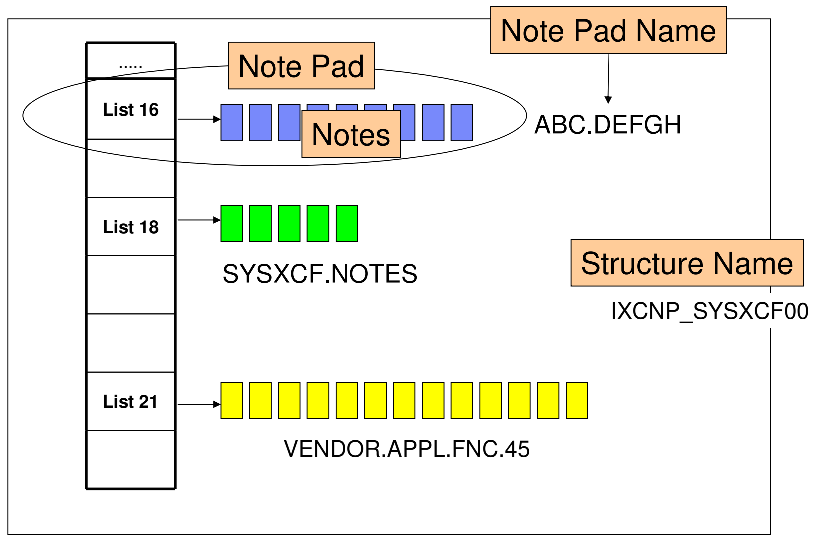
Do not confuse a connection to a note pad with a XES connection to a CF structure. Under the covers, XCF does the necessary IXLCONN to connect to the CF structure that contains the note pad. There will be at most one such connection per system. The note pad exploiters are unaware of the XES connection.

* More precisely, any system running with XCF Note Pad support. The system will need to be connected to the appropriate coupling facilities in order for the connection to be viable.



XCF Note Pad

XCF Note Pad Structure



20

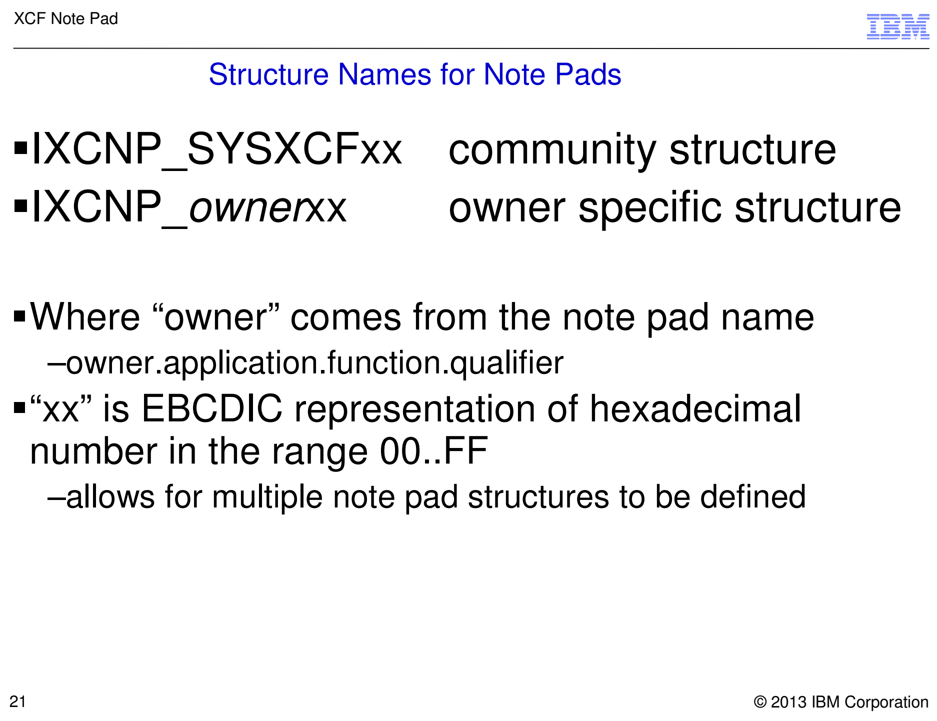
© 2013 IBM Corporation

So as you might suspect, a Note Pad is really just a list structure in a coupling facility.

Actually the list structure, shall we call it a “Note Pad structure”, can contain multiple Note Pads. A Note Pad is really just a collection of list entries residing on some one list in the list structure. Each list is a Note Pad. The list entries on the list are the notes.

In the slide, we depict a note pad structure containing three note pads. Each note pad has its own name and its own collection of notes. The names are totally made up for illustration purposes only. In particular, XCF itself does not currently make use of a note pad.

The note pad structure has a name, IXCNP_SYSXCF00 in this case. The installation would need to define this structure in the Coupling Facility Resource Management (CFRM) policy to make it available for use. XCF assumes that any structure name beginning with “IXCNP_” is intended for note pad use.



Structure Names for Note Pads

- IXCNP_SYSXCFxx community structure
- IXCNP_*owner*xx owner specific structure

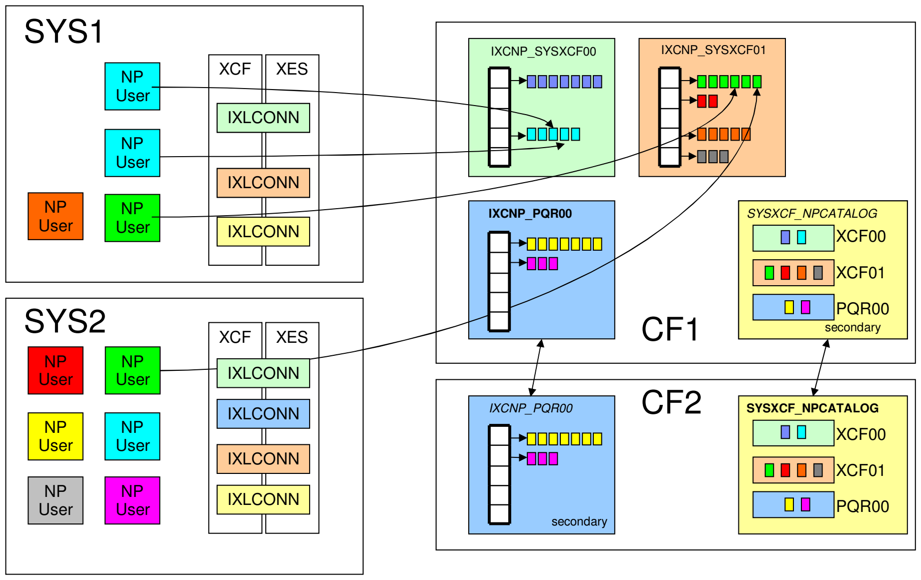
- Where "owner" comes from the note pad name
  - owner.application.function.qualifier
- "xx" is EBCDIC representation of hexadecimal number in the range 00..FF
  - allows for multiple note pad structures to be defined

The note pad structure has a name. All note pad structure names begin with "IXCNP_". The remainder of the name takes on one of two forms, SYSXCFxx or Ownerxx. The "xx" is the EBCDIC representation of a two digit hexadecimal number in the range 00..FF. The "owner" would correspond to the "owner" portion of a note pad name. Multiple note pad structures are supported.

We think of structures with names of the form IXCNP_SYSXCFxx as being for community use. That is, the structure can contain note pads from all sorts of different applications. We think of structures with names of the form IXCNP_*owner*xx as being vendor (or perhaps application) specific. We call these owner specific structures. In general, there is no need to define an owner specific structure for any given application. The XCF Note Pad Services were designed with the intent that note pads be comingled in the same structure. The owner specific structures are intended for use on an exception basis.

The XCF Note Pad Services was designed to support thousands of note pads. It just isn't practical to define a unique structure for every note pad, especially given that CFRM has a relatively low limit on the number of structures that can be defined in the sysplex. It is expected that any given note pad structure will contain multiple note pads.

XCF Note Pads in the Sysplex



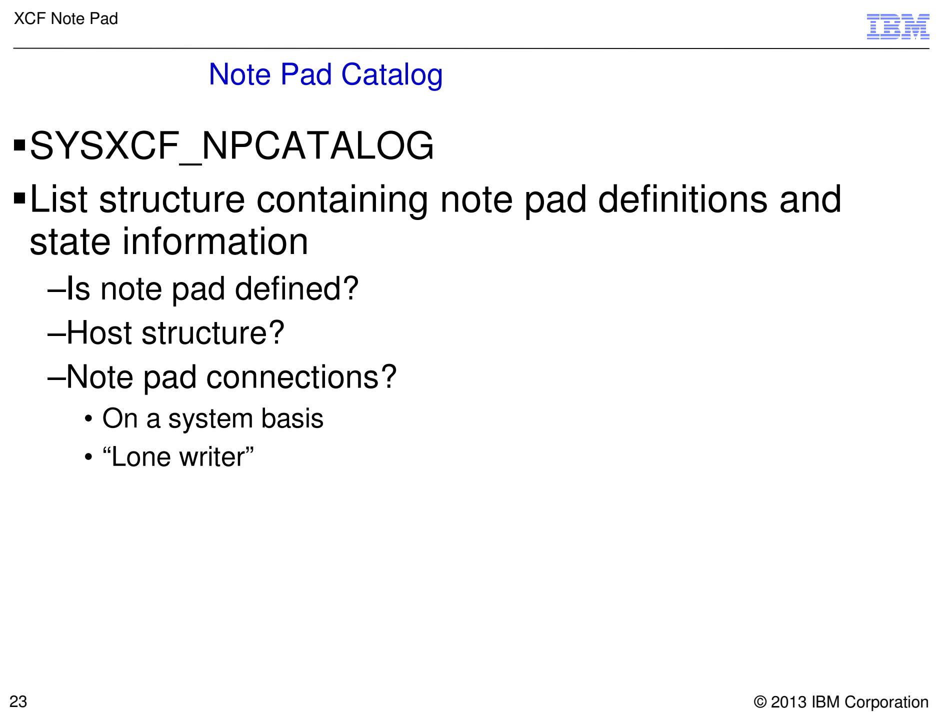
This slide tries to depict the complete package. We show two of the systems in the sysplex. Each system has various note pad connectors. On those systems, XCF uses XES services to connect (IXLCONN) to the note pad structures of interest. SYS1 has a connection to each of the note pad structures IXCNP_SYSXCF00 and IXCNP_SYSXCF01. However, it does not have a connection to IXCNP_PQR00 because it does not have any users connected to any of the note pads in that structure. On the other hand, SYS2 does have a connection to all three of the note pad structures because it does have users connected to the note pads in each of the structures.

Observe that structure IXCNP_SYSXCF01 has multiple note pads. Both SYS1 and SYS2 have users connected to those various note pads. However, the systems have but one XES connection to the structure. Observe that SYS1 has two note pad connections to the note pad in structure IXCNP_SYSXCF00, but again there is but one XES connection to the structure.

Observe that structure IXCNP_SYSXCF00 contains a (purple) note pad which does not currently have any note pad connectors, thus illustrating the point that a note pad persists even though the application is not currently using it.

Observe that note pad structure IXCNP_PQR00 is a duplexed structure with a primary instance in CF1 and a secondary instance in CF2. The applications using the note pads in that structure indicated that they needed to have their note pads duplexed. The installation accommodated the request. Structures IXCNP_SYSXCF00 and IXCNP_SYSXCF01 are not duplexed. The installation can configure duplexing (or not) for any note pad structure. The needs of the applications determine whether this is needed or not.

How does XCF know which structure hosts which note pad? If a user on SYS1 wanted to connect to the yellow note pad in IXCNP_PQR00, how would XCF know that it needed to establish a XES connection to that structure? That is where the XCF Note Pad Catalog comes in. Structure SYSXCF_NPCATALOG is the note pad catalog. The catalog has an entry for every note pad that indicates where the note pad resides. The slide attempts to depict this by showing the note pads that reside in each note pad structure. The catalog is a single point of failure since if the catalog is lost, XCF loses track of all the note pads. To mitigate that risk, it is strongly suggested that the catalog structure be duplexed, as depicted in the slide.

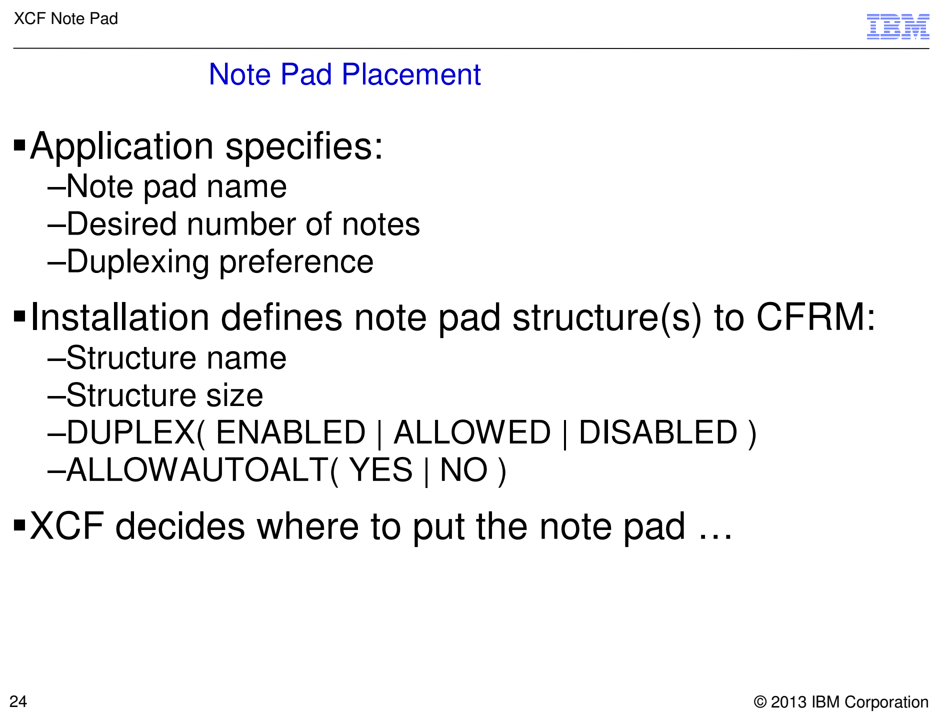


Note Pad Catalog

- SYSXCF_NPCATALOG
- List structure containing note pad definitions and state information
  - –Is note pad defined?
  - –Host structure?
  - –Note pad connections?
    - On a system basis
    - “Lone writer”

The XCF Note Pad Catalog structure named SYSXCF_NPCATALOG is required. This structure contains all the note pad definitions for the sysplex. In particular, it indicates which note pad structure is being used to host each note pad. It also identifies which systems happen to have users connected to the note pad. It does not have information about each connector on that system. Each system is responsible for its own connectors.

The creator of the note pad can specify MULTIWRITE=NO to indicate that there should be at most one connector with write access to the note pad. We in XCF refer to such a connector as the “Lone Writer”. SAP exploits this option for its note pads. The catalog structure records the existence of such a connector.



XCF Note Pad

Note Pad Placement

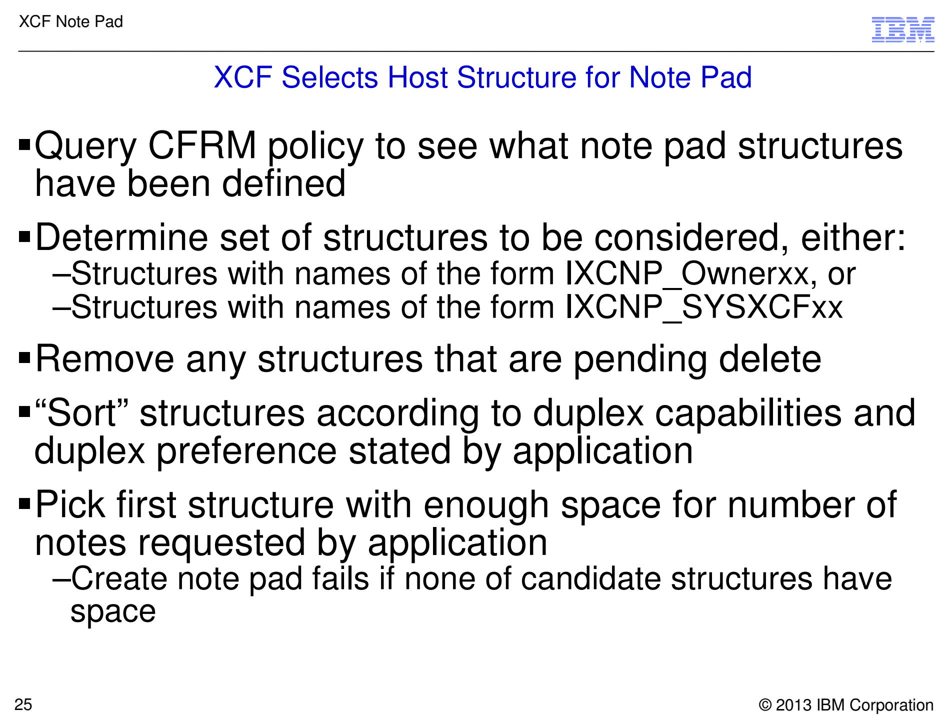
- Application specifies:
  - –Note pad name
  - –Desired number of notes
  - –Duplexing preference
- Installation defines note pad structure(s) to CFRM:
  - –Structure name
  - –Structure size
  - –DUPLEX(ENABLED | ALLOWED | DISABLED)
  - –ALLOWAUTOALT(YES | NO)
- XCF decides where to put the note pad …

24 © 2013 IBM Corporation

The XCF Note Pad Services chooses a note pad structure to host a note pad based on several factors. Some factors are specified by the application, others are in effect specified by the installation as a consequence of defining the note pad structures in the Coupling Facility Resource Management (CFRM) policy.

When the application invokes the IXCNOTE macro to create a note pad, it specifies the name of the note pad, the number of notes the note pad needs to have, and a preference as to whether the note pad ought to be duplexed. The application should document this information so that the installation can ensure that the necessary note pad structures are defined.

The installation needs to define note pad structures in the CFRM policy for XCF to use for hosting note pads. The CFRM policy specifies the name of the structure, its size, whether structure duplexing is used, and whether the structure can be automatically altered. Alter processing makes dynamic adjustments to the structure so that the various objects in the structure “match” the work load. Note that XCF does not actually use the ALLOWAUTOALT specification when choosing a host structure, so technically I should not have put it on the slide. However, the system programmer might or might not choose to specify ALLOWAUTOALT. We’ll have more to talk about on that topic as there are some concerns (maybe).



XCF Note Pad

XCF Selects Host Structure for Note Pad

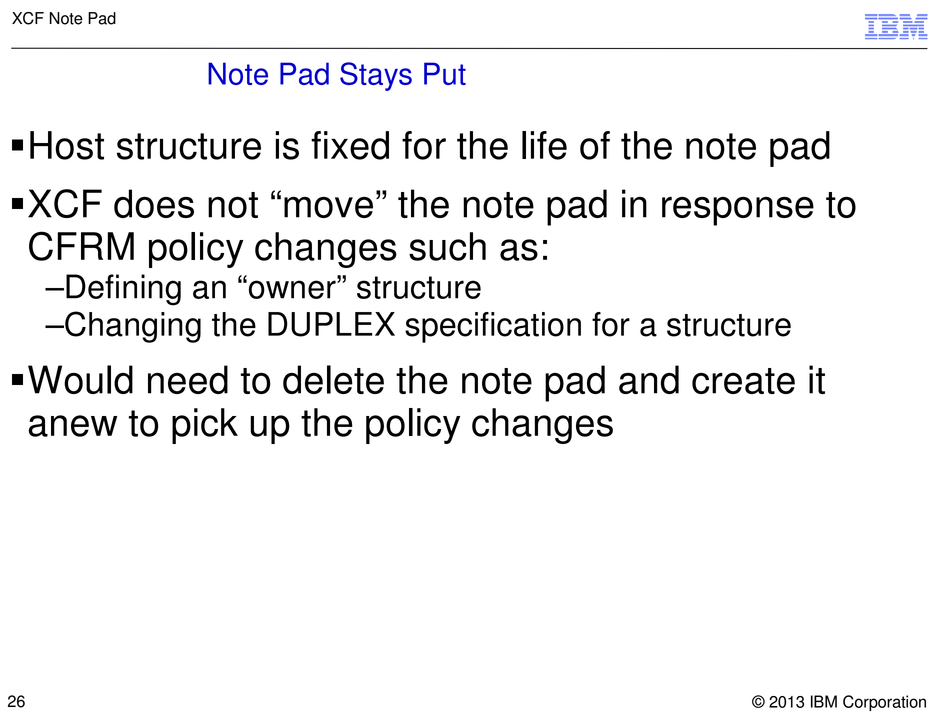
- Query CFRM policy to see what note pad structures have been defined
- Determine set of structures to be considered, either:
  - Structures with names of the form IXCNP_Ownerxx, or
  - Structures with names of the form IXCNP_SYSXCFxx
- Remove any structures that are pending delete
- “Sort” structures according to duplex capabilities and duplex preference stated by application
- Pick first structure with enough space for number of notes requested by application
  - Create note pad fails if none of candidate structures have space

25 © 2013 IBM Corporation

If owner specific note pad structures (that are not pending delete) are defined in the Coupling Facility Resource Management (CFRM) policy when note pad is created, the note pad goes in one of those structures. No other structures will be considered. If those structures are full or inaccessible, the create of the note pad fails.

If there are no owner specific note pad structures (that are not pending delete) defined in the CFRM policy when the note pad is created, the note pad will go in one of the community structures (that are not pending delete). No other structures will be considered. If those structures are full or inaccessible, the create of the note pad fails.

When creating the note pad, the application can indicate whether it prefers a duplexed structure or a non-duplexed structure. Based on that specification, XCF sorts the list of candidate structures according to their duplexing capabilities. For example, if the application prefers a duplexed structure, XCF would give precedence to duplexed structures over non-duplexed structures. Having sorted the list of candidate structures according to their duplex attributes, XCF would then consider each structure in turn until it finds one that has enough space to accommodate the requested number of notes. So the note pad could end up in a simplex structure if none of the duplexed ones had room for the note pad.



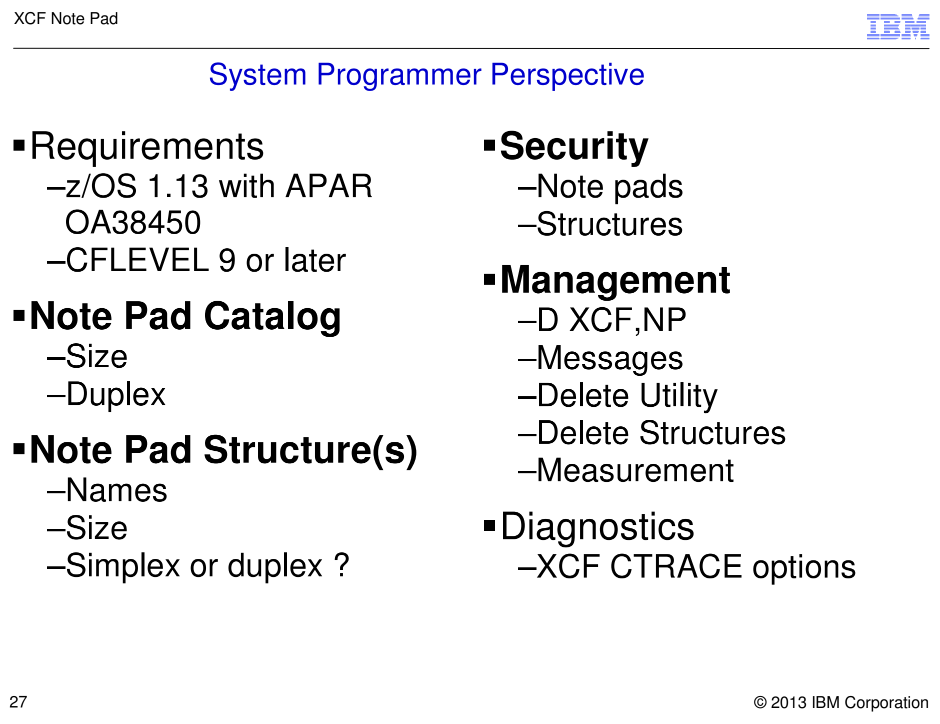
Note Pad Stays Put

- Host structure is fixed for the life of the note pad
- XCF does not “move” the note pad in response to CFRM policy changes such as:
  - –Defining an “owner” structure
  - –Changing the DUPLEX specification for a structure
- Would need to delete the note pad and create it anew to pick up the policy changes

During the create note pad process, XCF might consider several different structures. These candidates might be visible via the DISPLAY XCF,NOTEPAD command as XCF cycles through them. When the note pad is finally created, it will reside in one specific host structure. XCF has no ability to move the note pad to any other structure. Thus the host structure is fixed for the life of the note pad.

In particular, subsequent changes to the Coupling Facility Resource Management (CFRM) policy do not cause XCF to go back and rework the host structure selection process. For example, the note pad might have been created when the policy only had community structures defined (IXCNP_SYSXCFxx). If the policy is then later changed to define an owner specific structure (IXCNP_ownerxx), the note pad persists in the community structure. In order for the owner specific structure to become a candidate for hosting the note pad, the note pad would need to be deleted and then created again. During the create process, the new owner specific structure in the CFRM policy would be used.

How to get the note pad deleted? Hopefully the application provides documentation to indicate whether, when, and how a note pad can be safely deleted. You’d want to use those procedures. XCF provides a delete utility to allow you to delete a note pad, but you really need to understand the impact to the application.



XCF Note Pad

System Programmer Perspective

- Requirements
  - –z/OS 1.13 with APAR OA38450
  - –CFLEVEL 9 or later
- **Note Pad Catalog**
  - –Size
  - –Duplex
- **Note Pad Structure(s)**
  - –Names
  - –Size
  - –Simplex or duplex ?
- **Security**
  - –Note pads
  - –Structures
- **Management**
  - –D XCF,NP
  - –Messages
  - –Delete Utility
  - –Delete Structures
  - –Measurement
- Diagnostics
  - –XCF CTRACE options

27 © 2013 IBM Corporation

The system programmer needs to configure the sysplex to support note pads. During the presentation, I'll touch on the topics in bold. The other topics are either discussed here, or have hidden slides that you can read at your leisure.

XCF Note Pad Services are available z/OS V1R13 when APAR OA38450 is installed. The coupling facility functionality needs to be CFLEVEL=9 or later (which should not be an issue).

The Coupling Facility Resource Management (CFRM) policy must be updated to define the necessary coupling facility structures, the note pad catalog structure and the note pad structures used to host the note pads needed by the applications.

Appropriate System Authorization Facility (SAF) profiles need to be defined to control access to the note pads and the structures used by the XCF Note Pad Services.

We want to look at the various tools available for managing note pads.

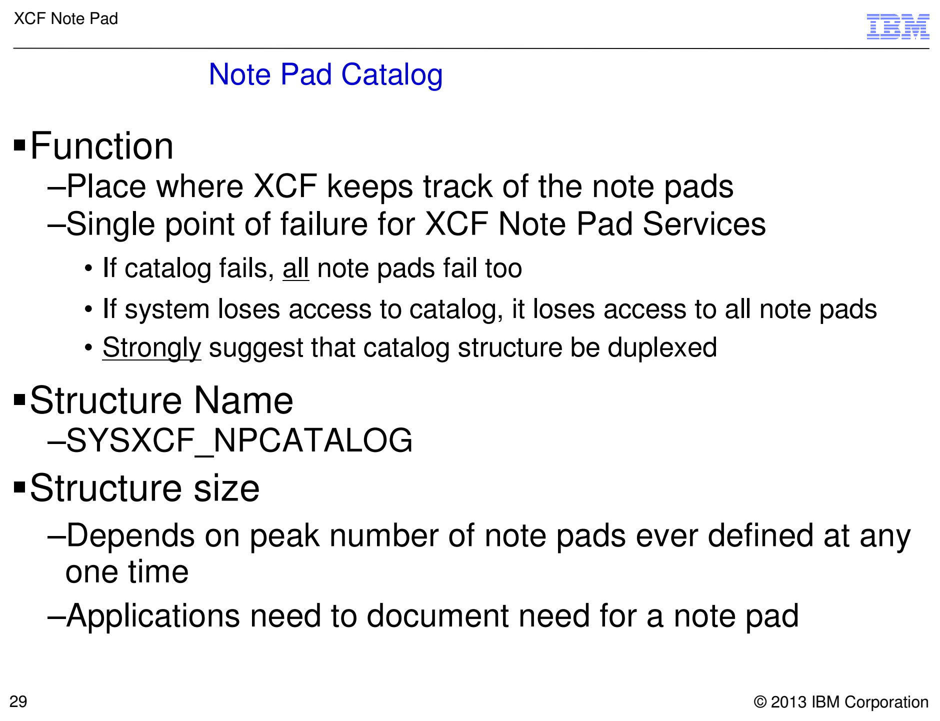
I don't discuss diagnostics, as you'd likely be working with IBM Service Personnel as needed. There are new options for the TRACE CT command to enable detailed XCF component tracing for note pad support. The traces can be filtered for subsets of note pads.

RACF APAR OA38720

- Potentially needed on z/OS V1R13 if there should happen to be an exploiter issuing IXCNOTE requests from MASTER
  - No such exploitation today
  - Rather doubt there will be such any time soon, if ever
- Fixes some inconsistencies between FASTAUTH (which XCF uses) and AUTH.

The issue appears to be related to how FASTAUTH deals with the address space level ACEE for MASTER. This ACEE is a "default" ACEE generated by SAF, and its (invalid) userid of +MASTER+ is not defined to RACF as a user. The UTOKEN for this "user" has the following flags set: TOKDFLT and TOKTRST. Without OA38720, FASTAUTH is always returning RC=8 for such cases, indicating that the "user" was not allowed to access the resource (even if we set up UACC of CONTROL!). In discussion with RACF, it was determined that this behavior was wrong and not consistent with how the non-XMEM check using AUTH behaves (AUTH permits the access). So RACF APAR OA38720 updated FASTAUTH to deal with MASTER address space in a way that is consistent with AUTH. For cases where the ACEE of the MASTER address space is used, we expect SAF to permit all accesses since MASTER is "trusted".

We rather doubt there will be any applications written to exploit XCF Note Pads that are intended to run out of MASTER (or similar spaces). If true, the RACF APAR is irrelevant. But if there are some, the RACF APAR will need to be installed to enable them to make it past the SAF checks.



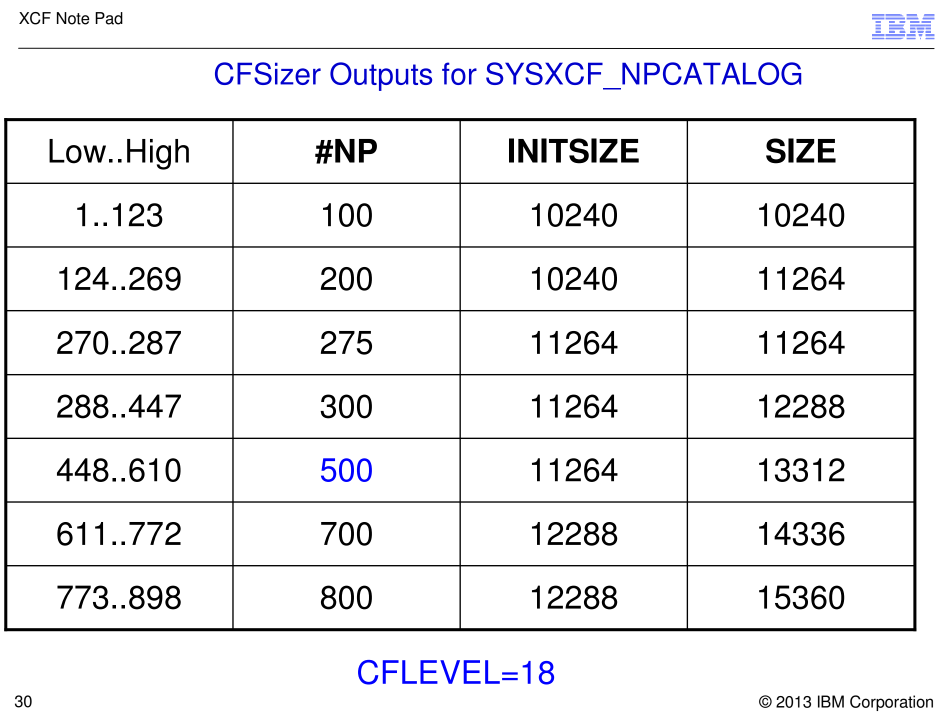
Note Pad Catalog

- Function
  - –Place where XCF keeps track of the note pads
  - –Single point of failure for XCF Note Pad Services
    - If catalog fails, all note pads fail too
    - If system loses access to catalog, it loses access to all note pads
    - Strongly suggest that catalog structure be duplexed
- Structure Name
  - –SYSXCF_NPCATALOG
- Structure size
  - –Depends on peak number of note pads ever defined at any one time
  - –Applications need to document need for a note pad

XCF maintains an entry in the note pad catalog structure for each note pad defined in the sysplex. The size of the note pad catalog structure is determined by the number of note pads that can be defined in the sysplex at any one time.

Peak number of note pads: If you should ever run consistently with the peak number of note pads defined, you might need to allow for small “float” to allow new instance of note pad to be created while the old instance is in the midst of being deleted. In such cases there could actually be two entries in the catalog for the same note pad. Should not generally be an issue as the delete of the old instance will likely be finished before the application is able to initiate the request to create a new instance. The entry for the deleted note pad can potentially linger, for example, if there were lots of notes to be deleted or if there was a loss of connectivity to the coupling facility that contains the note pad structure (or the note pad catalog). I rather doubt this concern will ever become a practical issue.

The note pad catalog is critical to the XCF Note Pad Services. Loss of the catalog induces loss of all the note pads. Loss of access to the catalog induces loss of access to all the note pads. To avoid this single point of failure, it is strongly suggested that the catalog structure be duplexed. There should not be lots of activity to this structure since it is only accessed in response to things like create, query, or delete note pad requests which should be relatively rare. The catalog is also accessed when the first connection to a note pad from a given system is created and when the last connection is deleted. Removing a system from the sysplex will also induce catalog accesses.



XCF Note Pad

CFSizer Outputs for SYSXCF_NPCATALOG

Low..High	#NP	INITSIZE	SIZE
1..123	100	10240	10240
124..269	200	10240	11264
270..287	275	11264	11264
288..447	300	11264	12288
448..610	500	11264	13312
611..772	700	12288	14336
773..898	800	12288	15360

CFLEVEL=18

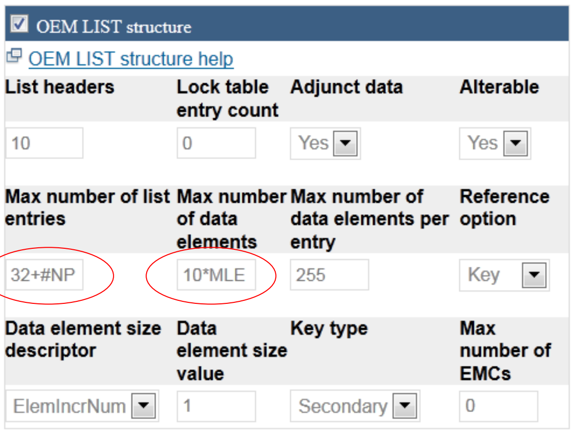
30

© 2013 IBM Corporation

Something in the neighborhood of a 12MB catalog structure will likely suffice for quite some time. Size it for 500 note pads and wait for the world to catch up in a few years.

#NP is the number of note pads to be supported by the note pad catalog. **INITSIZE** and **SIZE** are the CFSizer outputs that would then be used for the respective specifications in the CFRM policy. When I was playing with the sizer, I noticed that I was getting the same values for different numbers of note pads. I got curious as to where the cutoffs were, so the **Low..High** column shows the range for the number of note pads that yield the same INITSIZE and SIZE values from the CFSizer (as computed for a CFLEVEL=18 coupling facility). You really don't need such precision.

CFSizer Inputs for SYSXCF_NPCATALOG



☑ OEM LIST structure

OEM LIST structure help

List headers	Lock table entry count	Adjunct data	Alterable
10	0	Yes	Yes
Max number of list entries	**Max number of data elements**	**Max number of data elements per entry**	**Reference option**
32+#NP	10*MLE	255	Key
Data element size descriptor	**Data element size value**	**Key type**	**Max number of EMCs**
ElemIncrNum	1	Secondary	0

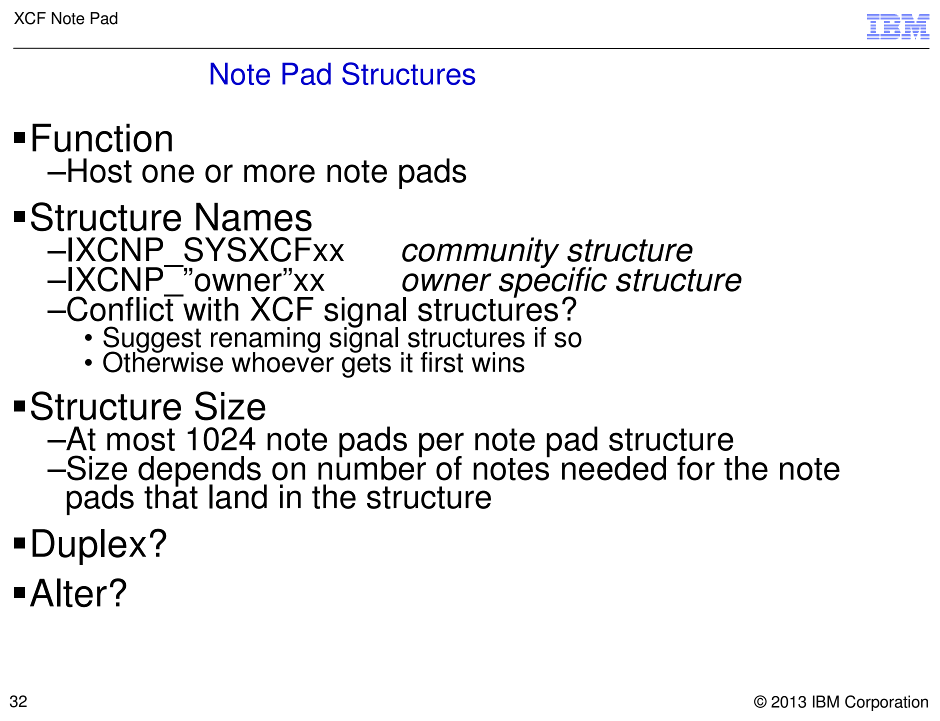
#NP = number of note pads to be defined in the sysplex
MLE = value entered in Max number of list entries (32+#NP)

There is no direct support within CFSizer for the note pad catalog. So you have to use the OEM option and size it as an OEM List Structure.

The circled items are the only variable input that might need "research". Both fields are determined by the number of note pads that the catalog needs to support. All other fields should be entered as shown.

.

But the previous slide, and *Setting Up a Sysplex* already did the work for you.



Note Pad Structures

- Function
  - –Host one or more note pads
- Structure Names
  - –IXCNP_SYSXCFxx *community structure*
  - –IXCNP_"owner"xx *owner specific structure*
  - –Conflict with XCF signal structures?
    - Suggest renaming signal structures if so
    - Otherwise whoever gets it first wins
- Structure Size
  - –At most 1024 note pads per note pad structure
  - –Size depends on number of notes needed for the note pads that land in the structure
- Duplex?
- Alter?

XCF Note Pad Services are designed to support way more note pads than XES supports structures. In particular, we intend for many note pads to reside in the same note pad structure. XCF controls the access to note pads and ensures that note pads are isolated from each other. So in general, a community structure that hosts one or more note pads should be adequate for most installations. You might make use of an owner specific structure in order to accomplish some unique purpose. Note that an owner specific structure could also host more than one note pad as well. But that would likely depend on how the application names its note pads.

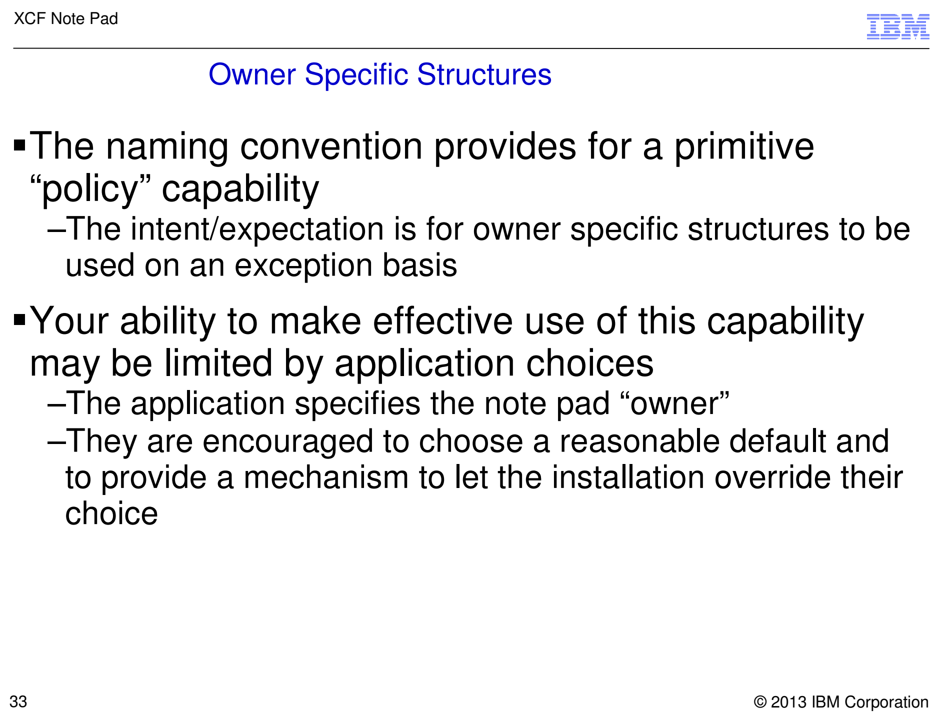
XCF signalling structures also begin with the prefix IXC. Perhaps you have an XCF signal structure whose name begins with "IXCNP_". If so there is a potential conflict and I suggest that you rename the signal structure. If there is a conflict, whichever service allocates the structure first "wins". The loser defers to the winner in a gentlemanly fashion.

The size of the note pad structure is determined by the number of notes it needs to hold, which is the sum of the number of notes needed by each of the note pads hosted by the structure. Not so bad at the moment, as there is one well defined exploiter. But over time the number of applications making use of note pads might well grow. I rather doubt you want to spend time hunting them down and doing the math. In addition, if there are multiple note pad structures defined, you may not be able to reliably predict which structure will be selected by XCF to host the note pad. So I believe you really want to think of the note pad structures as providing a pool of note resources for applications to use. Don't try to micromanage the size. Make it large enough to provide reasonable capacity. Monitor it if you must, or just wait for XCF to complain when it is unable to allocate a new note pad.

A given note pad structure can support at most 1024 note pads. Some day you might need to define a another note pad structure to support more note pads.

The user of the note pad needs to document the number of notes it requires and whether it wants the note pad to be hosted by a duplexed structure. At this time, there are no exploiters that desire duplexing. But should that occur, you would need to configure a duplexed note pad structure to satisfy that need.

In general it is good practice to permit AUTOALTER to alter the structure. The concern for note pad structures is that when the CF becomes full, alter processing might reduce the size of the note pad structure. Doing so reduces the number of notes in the structure, which in turn could impact the note pad users if XCF is unable to create a note as promised. We doubt that this will be a practical issue any time soon.

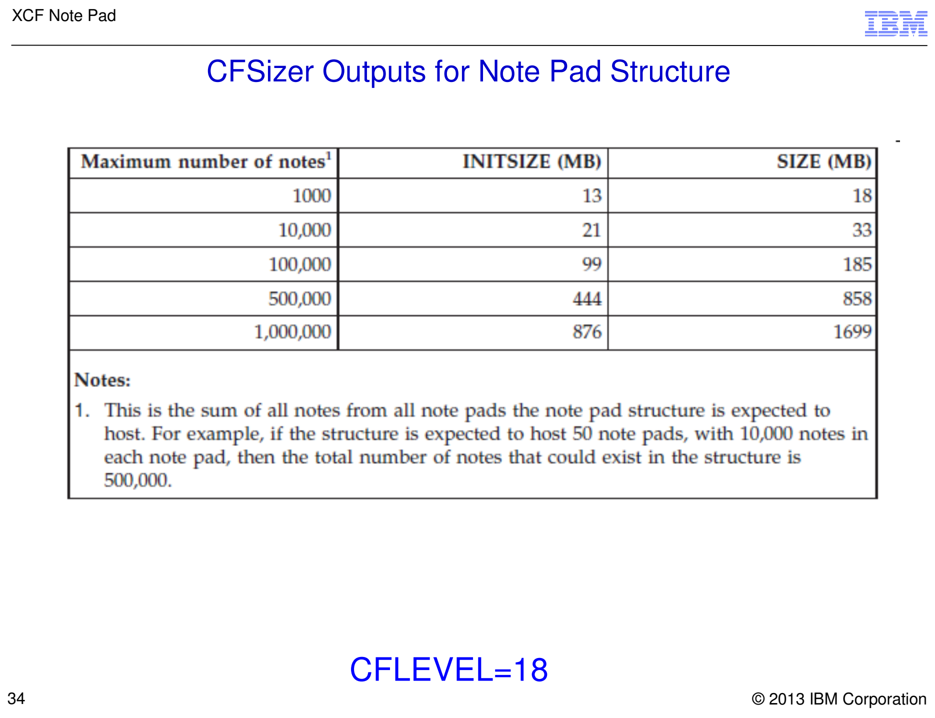


Owner Specific Structures

- The naming convention provides for a primitive “policy” capability
  - –The intent/expectation is for owner specific structures to be used on an exception basis
- Your ability to make effective use of this capability may be limited by application choices
  - –The application specifies the note pad “owner”
  - –They are encouraged to choose a reasonable default and to provide a mechanism to let the installation override their choice

In general, “community” structures will likely be fine and owner specific structures would only be needed in order to accomplish some unique purpose.

Time will tell whether we need true policy capability to provide more precise control over note pad placement.



XCF Note Pad

IBM

CFSizer Outputs for Note Pad Structure

Maximum number of notes[1]	INITSIZE (MB)	SIZE (MB)
1000	13	18
10,000	21	33
100,000	99	185
500,000	444	858
1,000,000	876	1699

Notes:

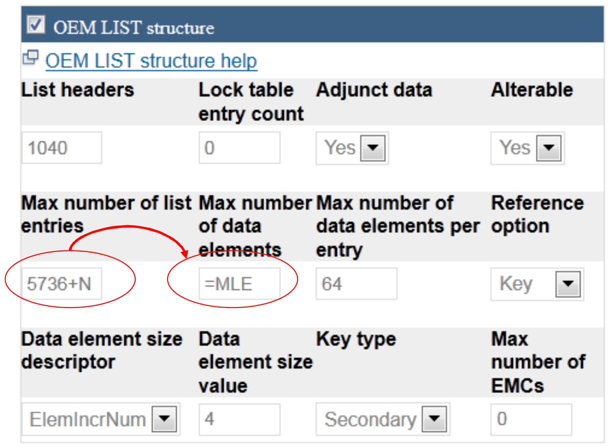
1. This is the sum of all notes from all note pads the note pad structure is expected to host. For example, if the structure is expected to host 50 note pads, with 10,000 notes in each note pad, then the total number of notes that could exist in the structure is 500,000.

CFLEVEL=18

34 © 2013 IBM Corporation

From *Setting Up A Sysplex,* here are some representative sizings for note pad structures at CFLEVEL=18.

CFSizer Inputs for Note Pad Structure(s)



☑ OEM LIST structure

OEM LIST structure help

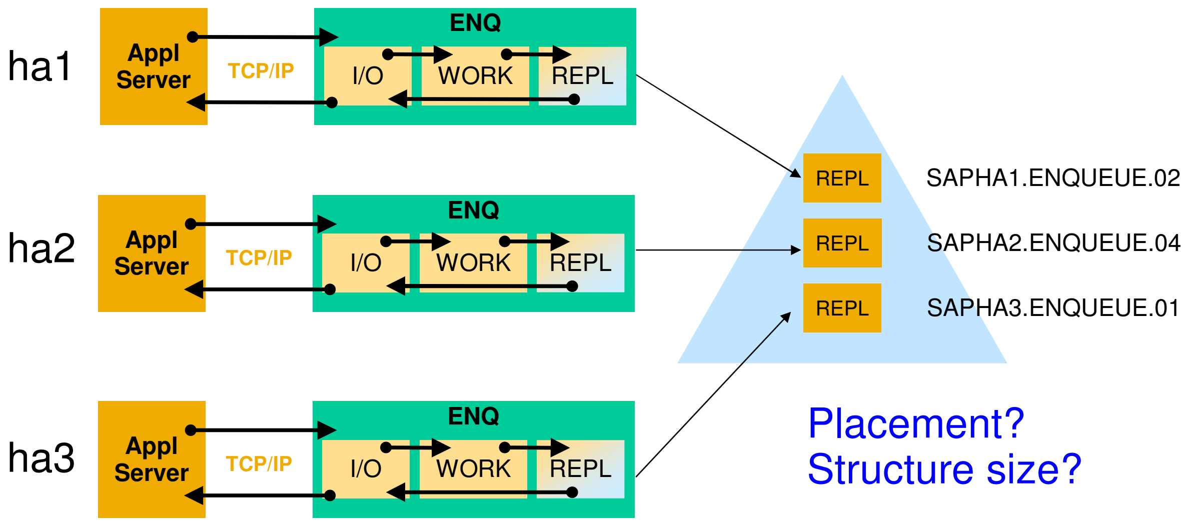
List headers	Lock table entry count	Adjunct data	Alterable
1040	0	Yes	Yes
Max number of list entries	**Max number of data elements**	**Max number of data elements per entry**	**Reference option**
5736+N	=MLE	64	Key
Data element size descriptor	**Data element size value**	**Key type**	**Max number of EMCs**
ElemIncrNum	4	Secondary	0

N – Total number of notes to be supported by structu
=MLE – Use same value as for Max number of list entries

CFSizer does not have specific support for note pad structures, so need to use the OEM List Structure option. Your challenge is to determine the number of notes needed. The circled fields are the only variable inputs. The same value should be entered in both places. All other fields should be set as indicated.

I really would not bother with computing a precise number of notes "N". Get a ball park figure, perhaps increase it be some percentage or fixed amount to allow some room for growth, and be done with it.

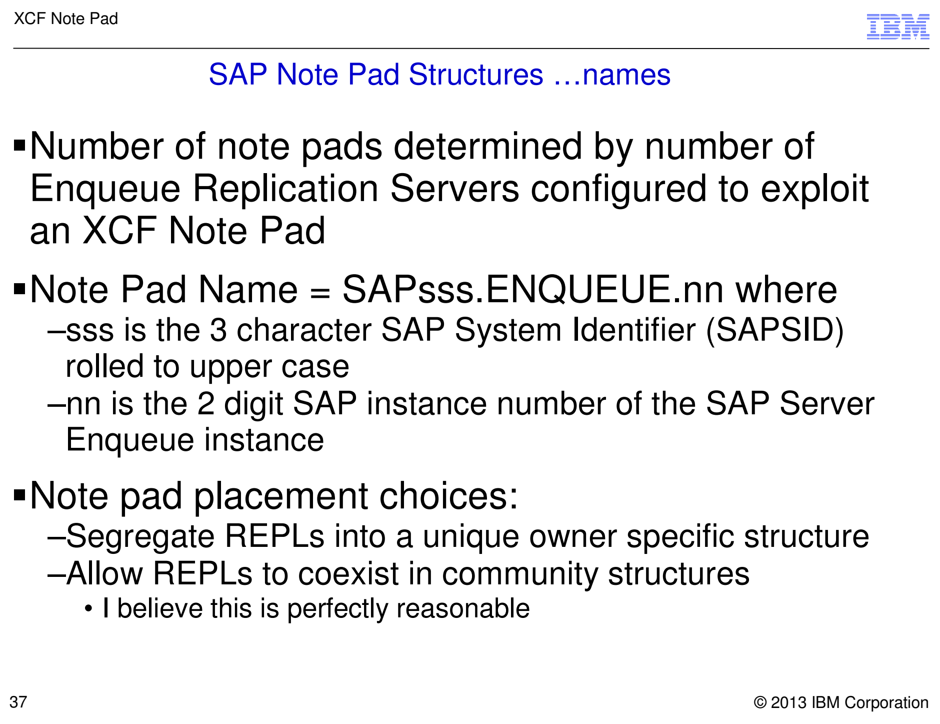
SAP Note Pad Structures



Placement?
Structure size?

Your installation may have one or more SAP instances. Each instance has a unique SAP System Identifier (SAPSID). For example ha1, ha2, and ha3 as depicted in the slide. Each instance would have its own ENQ server. You might choose to reconfigure one or more of the ENQ so that an XCF Note Pad is used to replicate the state information for ENQ. There would be a unique note pad for each.

Questions that arise: do I want to put each note pad (REPL) in its own note pad structure, put them all in the same structure, or some combination of both? Having decided on how I want the note pads distributed across the structures, what size should those structures be?



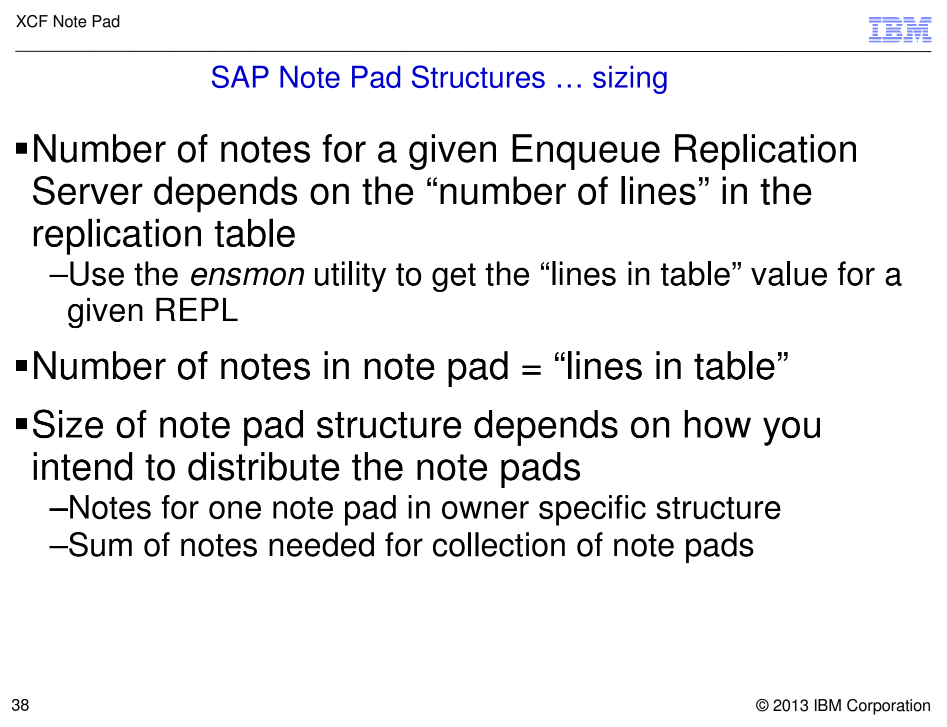
XCF Note Pad

SAP Note Pad Structures …names

- Number of note pads determined by number of Enqueue Replication Servers configured to exploit an XCF Note Pad
- Note Pad Name = SAPsss.ENQUEUE.nn where
  - –sss is the 3 character SAP System Identifier (SAPSID) rolled to upper case
  - –nn is the 2 digit SAP instance number of the SAP Server Enqueue instance
- Note pad placement choices:
  - –Segregate REPLs into a unique owner specific structure
  - –Allow REPLs to coexist in community structures
    - I believe this is perfectly reasonable

37 © 2013 IBM Corporation

The SAP installation guide suggests that the REPL for production be isolated into its own structure. That is, define an owner specific structure named IXCNP_SAPsss01 (for example) to host the note pad. In effect this means the note pad structure will contain exactly one note pad. In a large installation, there might be dozens of SAP "systems" each with its own ENQ server and therefore its own unique note pad. I really can't see that you'd want to define and manage the dozens of structures needed to accomplish that. And as argued on a subsequent slide, co-location of note pads in the same community structure should not give rise to any sort of concerns with respect to performance or interference between note pads. I suggest that you simply make use of community structures.



XCF Note Pad

SAP Note Pad Structures … sizing

- Number of notes for a given Enqueue Replication Server depends on the “number of lines” in the replication table
  - –Use the *ensmon* utility to get the “lines in table” value for a given REPL
- Number of notes in note pad = “lines in table”
- Size of note pad structure depends on how you intend to distribute the note pads
  - –Notes for one note pad in owner specific structure
  - –Sum of notes needed for collection of note pads

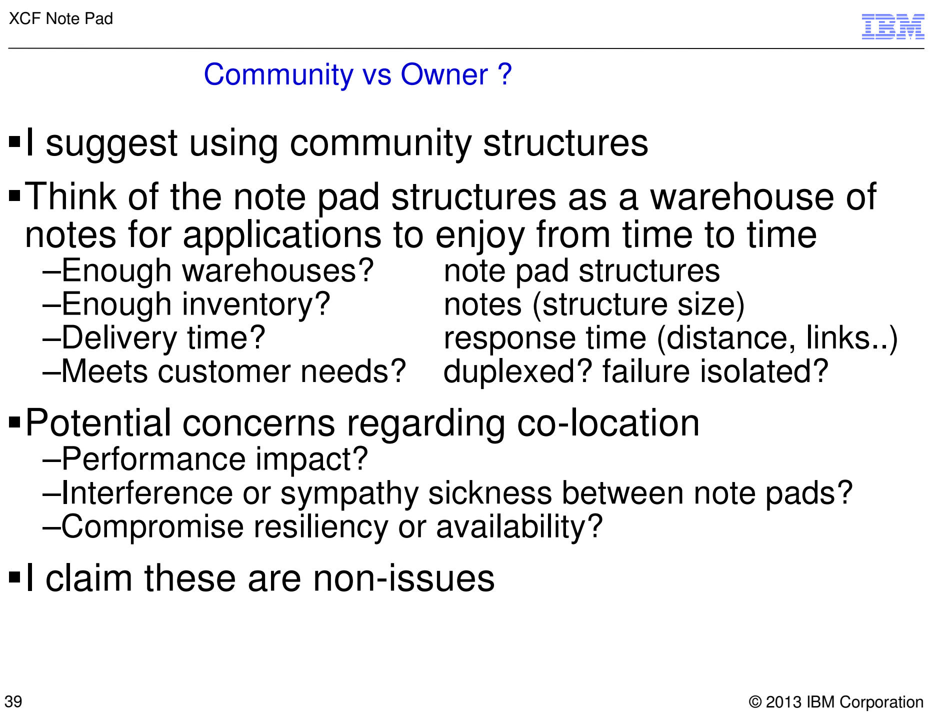
38 © 2013 IBM Corporation

The SAP enqueue table size is defined in the SAP (A)SCS profile via the following parameter:

enque/table_size = 64000 (for example)

You might think that the size of the note pad would need to allow for that many notes as well. However, the replication table (REPL) only needs to record information about the locks that are actually held. Thus there really only needs to be enough notes to cover the maximum number of locks that might be held at one time. Well, SAP has a utility that you can run to determine this value. After SAP is up and running, use the *ensmon* utility which produces a report that includes information about the replication table. The value reported for “lines in table” is the number of notes required for the note pad. In this example, 57033.

```
ensmon pf=<profile of SAP ENQ instance> 2

.....

static description of the replication table:

-------------------------------------------

line size : 744 Bytes

table size : 43573360 Bytes

lines in table : 57033 lines
```



Community vs Owner ?

- I suggest using community structures
- Think of the note pad structures as a warehouse of notes for applications to enjoy from time to time

–Enough warehouses?	note pad structures
–Enough inventory?	notes (structure size)
–Delivery time?	response time (distance, links..)
–Meets customer needs?	duplexed? failure isolated?

- Potential concerns regarding co-location
  - –Performance impact?
  - –Interference or sympathy sickness between note pads?
  - –Compromise resiliency or availability?
- I claim these are non-issues

I think there would be these concerns when co-locating note pads in the same structure:

(1) Might co-locating note pads in the same structure lead to some sort of overload within the structure that somehow impacts performance?

(2) Might co-locating note pads in the same structure lead to some sort of interference such that the activity of one note pad compromises the ability of another note pad to accomplish its activity?

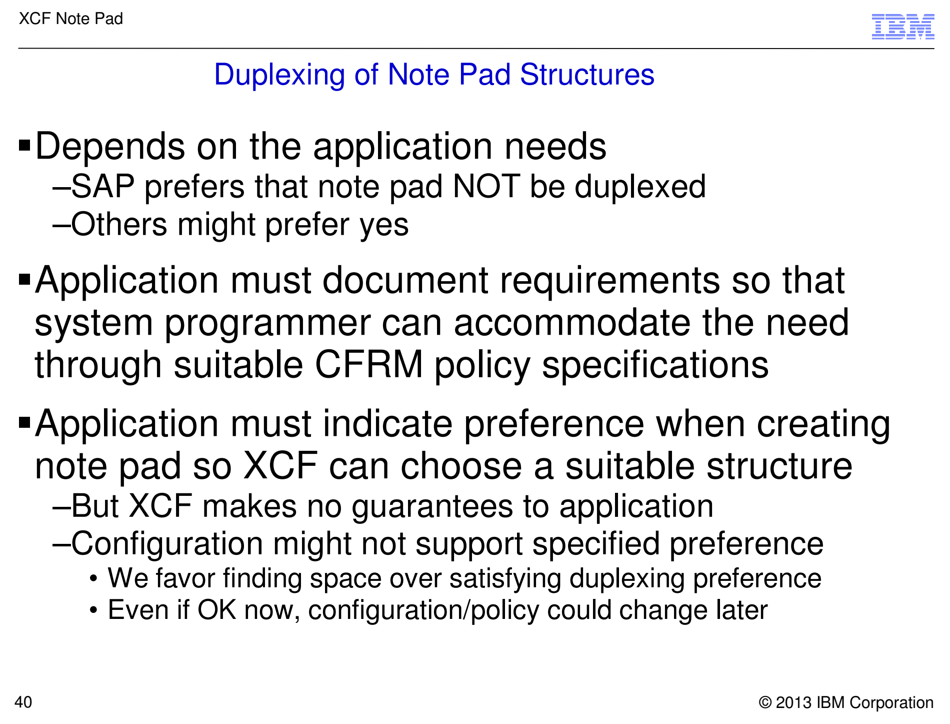
(3) Might we be compromising resiliency and/or availability by co-locating the note pads in the same structure?

Frankly, the answer to (1) is “unknown” as we do not have sufficient customer experience with these environments to say. However, past experience with various coupling facility structures suggests that the CF can indeed maintain excellent performance even at high rates. In general, serialization within a list structure is list centric. Thus if contention were to arise, it would do so for a given list (ie, note pad). In other words, if a given note pad is experiencing contention within the structure, it will experience that contention even if it were the only note pad in the structure.

Since XCF ensures isolation between note pads within the structure, question (2) is essentially about whether one note pad can consume list entries in the structure such that a different note pad would be denied the list entry needed for its note. But XCF protocols ensure that the structure chosen to host the note pad has enough capacity to provide every note pad with the requested (promised) number of notes. That promise can be broken if the structure is altered to reduce its size. But the reduction could occur for any note pad structure (assuming you permit it). Thus isolating a note pad in its own structure affords no extra protection.

For (3), the coupling facility rarely has a structure fail. The more typical scenario is where the entire coupling facility fails. Thus all the structures in that coupling facility would fail. So having one note pad structure or lots of them in the coupling facility does not provide any additional availability. So the key is to spread the note pad structures across multiple facilities so that loss of one CF does not also cause a loss of all note pads.

Installations that are service providers for other companies may have concerns as they often need to isolate their clients from each other within the same sysplex. Might be tricky as the applications may not offer sufficient control over the “owner” portion of the note pad name to make it possible to direct the note pads into an owner (company) specific note pad structure. There is no direct control over XCF placement of note pads in a community structure, though one might be able to make this work if there is a coupling facility that is only accessible to a given “company” (subset of systems). In that case, the preference list could be used to force a community note pad into the isolated CF. The note pads would land in the right community note pad because it would be the only one accessible to the company systems. But in the end, I rather suspect that additional XCF support will be needed to make this easier to manage.

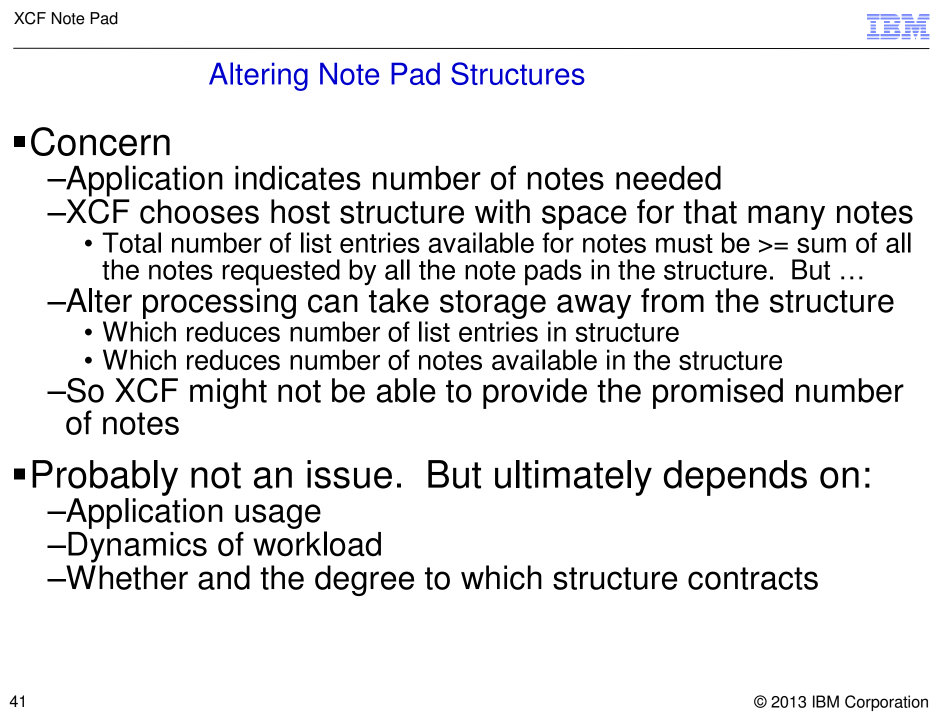


Duplexing of Note Pad Structures

- Depends on the application needs
  - –SAP prefers that note pad NOT be duplexed
  - –Others might prefer yes
- Application must document requirements so that system programmer can accommodate the need through suitable CFRM policy specifications
- Application must indicate preference when creating note pad so XCF can choose a suitable structure
  - –But XCF makes no guarantees to application
  - –Configuration might not support specified preference
    - We favor finding space over satisfying duplexing preference
    - Even if OK now, configuration/policy could change later

When we presented the XCF Note Pad Services, vendors requested that it be possible to get the note pad hosted by a duplexed structure. Thus the creator of the note pad can state a preference as to whether the host structure be duplexed or not. It is only a preference and XCF does not guarantee that the preference will be honored. SAP has flatly stated that they do not want the note pad to be in a duplexed structure as their primary interest is the improved performance.

In the future, applications will need to document their preference so that installations can provide the desired configuration. When that time comes, I would expect there to be pools of note pad structures, some duplexed and some not.



Altering Note Pad Structures

- Concern
  - –Application indicates number of notes needed
  - –XCF chooses host structure with space for that many notes
    - Total number of list entries available for notes must be >= sum of all the notes requested by all the note pads in the structure. But …
  - –Alter processing can take storage away from the structure
    - Which reduces number of list entries in structure
    - Which reduces number of notes available in the structure
  - –So XCF might not be able to provide the promised number of notes
- Probably not an issue. But ultimately depends on:
  - –Application usage
  - –Dynamics of workload
  - –Whether and the degree to which structure contracts

In general ALTER processing is desirable as it lets the system tailor the resources based on observed behavior. SAP usage tends to be "sparse", meaning the number of notes that exist in the note pad at any one time is far less than the maximum number of notes requested for the note pad (although this can vary based on the dynamics of the workload). The sparse note usage implies that it is likely possible for the structure to decrease in size without adverse impact. Also note that in current practice, structures do not usually get altered smaller until such time as the coupling facility itself starts to run out of storage (for example, 90% in use).

In this regard, it helps to have multiple note pads in the same structure as this increases the odds that the structure will have list entries in use when alter processing runs. Alter processing generally considers the number of entries in use when determining how far it can contract the structure. Alter will set aside a certain percentage of the in use entries as white space. So as the number of in use entries increases, the likelihood that the contraction will constrain the note pads decreases.

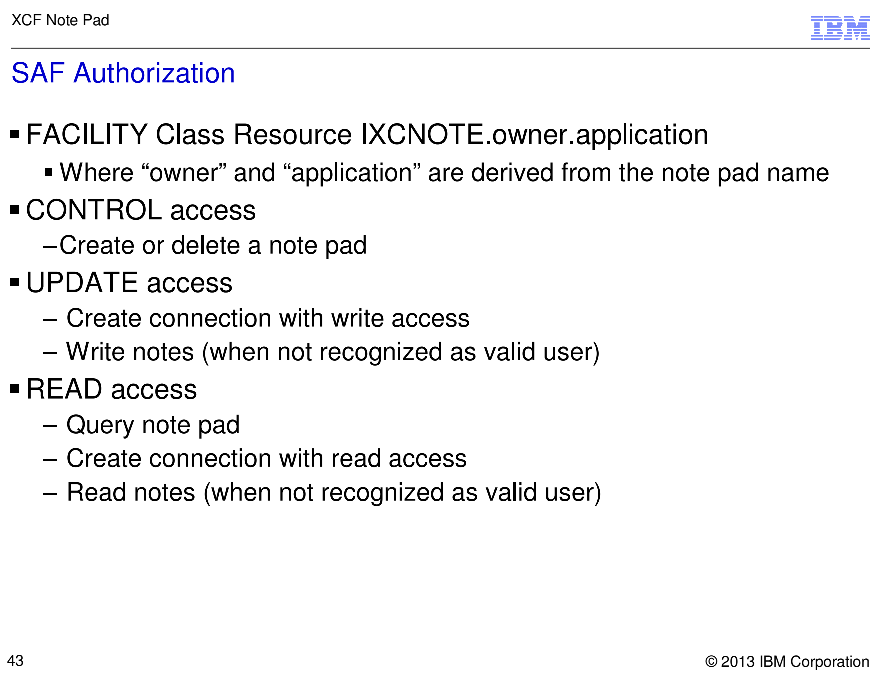
In general XCF cannot know the nature of the application usage. The XCF Note Pad Services are available for general use and XCF will not in general be aware of the exploiters that might arise in the future. So even if there are no issues today, it could become a concern in the future as the number and nature of the exploiters changes.

Security - Note Pad Access

- Requests from unauthorized programs rejected if:
  - SAF not installed, or
  - No SAF profile exists for the note pad, or
  - The profile does not permit the requested access
- Requests from authorized programs rejected if:
  - SAF is installed, and
  - A SAF profile exists for the note pad, and
  - The profile does not permit the requested access
- The Security Administrator needs to know the name of the note pad and the type of access needed by the program in order set up the SAF profile

The XCF Note Pad interface supports both authorized and unauthorized callers. The security administrator might have to define System Authorization Facility (SAF) profiles for certain note pads to grant access to the unauthorized callers. For authorized callers, XCF will honor the SAF profile if one is defined. If SAF is not installed, or the installation has not defined a SAF profile for the note pad, XCF rejects the request made by an unauthorized caller and permits the request to go forward for an authorized caller.

SAP runs unauthorized. So security profiles will certainly be required for them.



XCF Note Pad

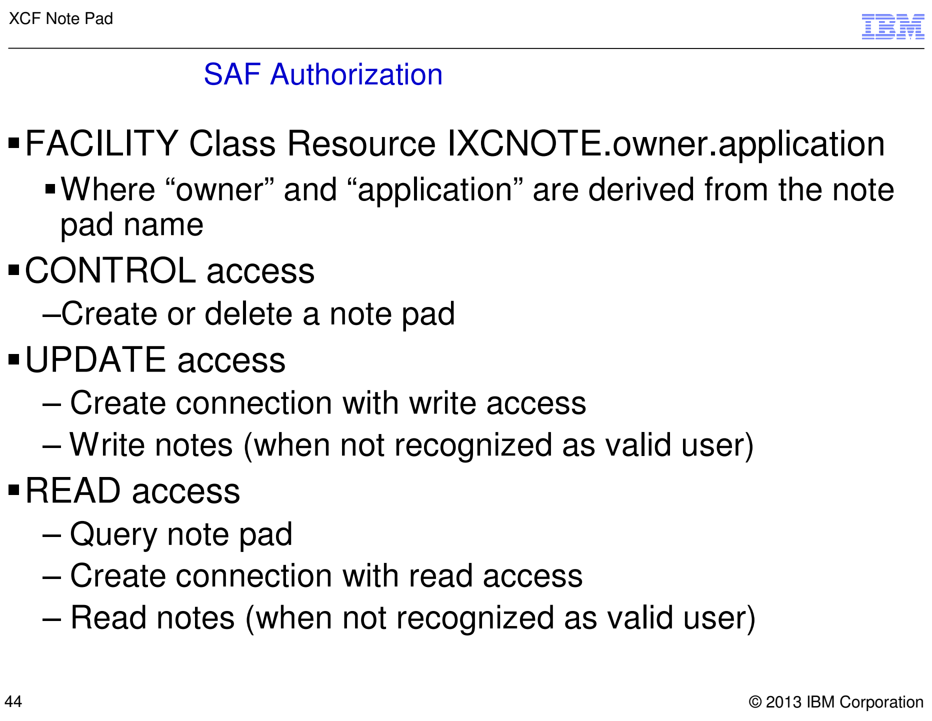
SAF Authorization

- FACILITY Class Resource IXCNOTE.owner.application
  - Where “owner” and “application” are derived from the note pad name
- CONTROL access
  - Create or delete a note pad
- UPDATE access
  - Create connection with write access
  - Write notes (when not recognized as valid user)
- READ access
  - Query note pad
  - Create connection with read access
  - Read notes (when not recognized as valid user)

43 © 2013 IBM Corporation

The application needs to document the name of the note pad so that the installation can set up the appropriate System Authorization Facility (SAF) profiles. The “owner” and “application” portions of the note pad name are used to define the profile. Note that a given application could have several different note pads with the same “owner” and “application” name. The one profile would provide access to them all.

I suspect that many programs will need CONTROL access so that the application has the full range of note pad services available to it. It is conceivable that an application might be decomposed into different parts such that one part is responsible for creating and deleting notes, while other parts are responsible for manipulating notes in the note pad. In such cases, it might be reasonable to establish profiles for the various parts so that each part has no more access rights than it requires.



XCF Note Pad

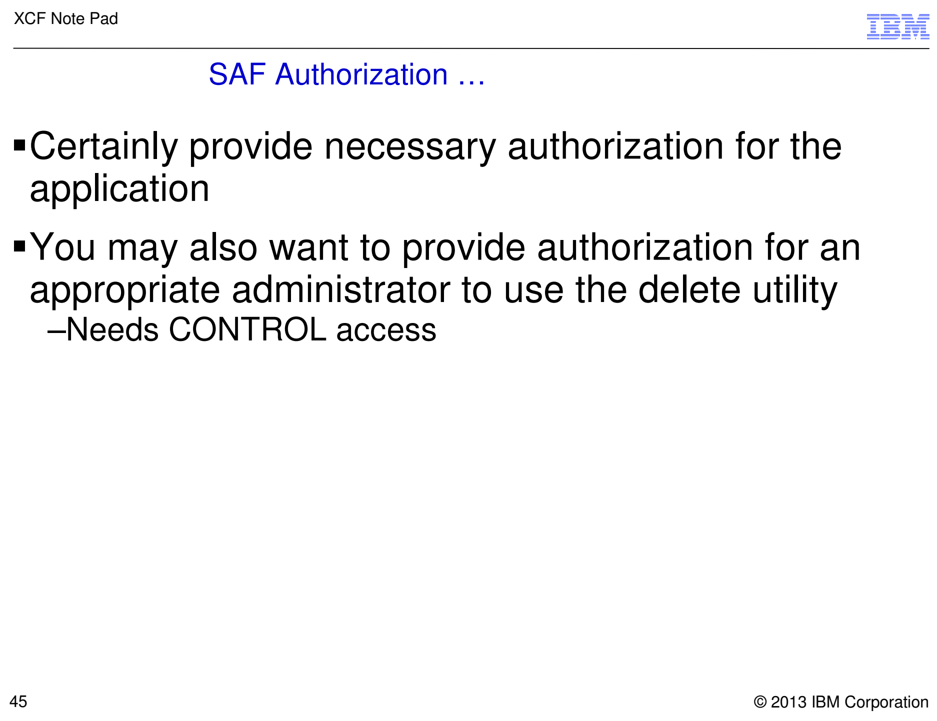
SAF Authorization

- FACILITY Class Resource IXCNOTE.owner.application
  - Where “owner” and “application” are derived from the note pad name
- CONTROL access
  – Create or delete a note pad
- UPDATE access
  – Create connection with write access
  – Write notes (when not recognized as valid user)
- READ access
  – Query note pad
  – Create connection with read access
  – Read notes (when not recognized as valid user)

44 © 2013 IBM Corporation

The application needs to document the name of the note pad so that the installation can set up the appropriate System Authorization Facility (SAF) profiles. The “owner” and “application” portions of the note pad name are used to define the profile. Note that a given application could have several different note pads with the same “owner” and “application” name. The one profile would provide access to them all.

I suspect that many programs will need CONTROL access so that the application has the full range of note pad services available to it. It is conceivable that an application might be decomposed into different parts such that one part is responsible for creating and deleting note pads (which requires CONTROL access), while other parts are responsible for manipulating notes in the note pad (which requires UPDATE access). In such cases, it might be reasonable to establish profiles for the various parts so that each part has no more access rights than it requires. The application documentation needs to provide the necessary guidance.



SAF Authorization …

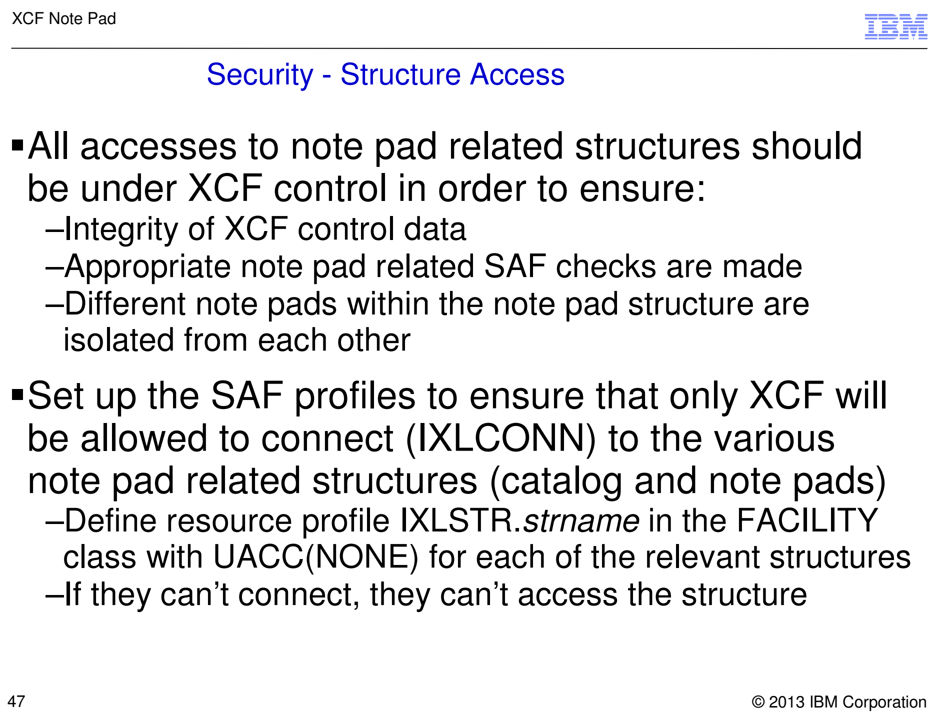
- Certainly provide necessary authorization for the application
- You may also want to provide authorization for an appropriate administrator to use the delete utility
  - –Needs CONTROL access

As we shall see in a subsequent slide, XCF provides a delete utility that might be needed to delete a note pad. Whoever runs the utility would need to have the appropriate authority for deleting the note relevant note pad.

SAF for SAP Note Pads

- SAP runs unauthorized
- Thus a SAF profile for the SAP note pad must be defined in order for SAP to access the note pad
- The SAP administrator that starts the ENQ server needs to have CONTROL access to the relevant note pad

The SAP administrator user <sapsid>adm that starts the enqueue server needs to be authorized to use the note pad. <sapsid> is the SAP System ID.



Security - Structure Access

- All accesses to note pad related structures should be under XCF control in order to ensure:
  - –Integrity of XCF control data
  - –Appropriate note pad related SAF checks are made
  - –Different note pads within the note pad structure are isolated from each other
- Set up the SAF profiles to ensure that only XCF will be allowed to connect (IXLCONN) to the various note pad related structures (catalog and note pads)
  - –Define resource profile IXLSTR.*strname* in the FACILITY class with UACC(NONE) for each of the relevant structures
  - –If they can’t connect, they can’t access the structure

By default, any programs that run in supervisor state or PKM allowing keys 0 to 7 can use the IXLCONN macro to establish XES connections to the XCF catalog and note pad structures. However, the data in these structures is managed solely by XCF, and allowing other programs to access the data directly through established XES connections could cause serious data integrity issues. IBM suggests that installations use Security Authorization Facility (SAF) to restrict access to the XCF catalog and note pad structures. No additional action is needed to grant XCF access. Note that restricting access to a note pad structure does not prohibit a program from accessing the note pads hosted in that note pad structure. Those accesses are made under XCF control via the IXCNOTE interface macro.

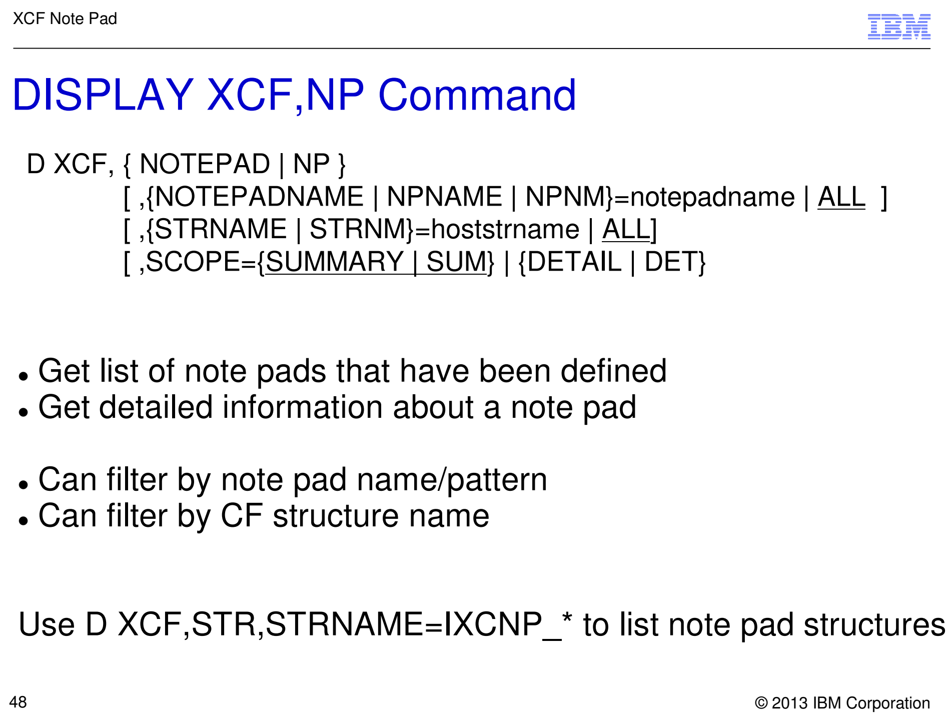
The following steps describe how the RACF security administrator can define RACF profiles to control the use of XCF catalog and note pad structures:

1. Define resource profile IXLSTR.structure-name in the FACILITY class for each of the note pad structures as well as the note pad catalog structure. The Universal Access Authority (UACC) should be set to NONE to prohibit access from any programs other than XCF.

2. Make sure the FACILITY class is active, and generic profile checking is in effect. If in-storage profiles are maintained for the FACILITY class, refresh them. For example, if an installation wants to restrict access to the XCF catalog structure, the security administrator can use the following commands:

```
RDEFINE FACILITY IXLSTR.SYSXCF_NPCATALOG UACC(NONE)
SETROPTS CLASSACT(FACILITY)
```

Similar steps should be taken for each note pad structure.



DISPLAY XCF,NP Command

```
D XCF, { NOTEPAD | NP }
        [ ,{NOTEPADNAME | NPNAME | NPNM}=notepadname | ALL  ]
        [ ,{STRNAME | STRNM}=hoststrname | ALL]
        [ ,SCOPE={SUMMARY | SUM} | {DETAIL | DET}
```

- Get list of note pads that have been defined
- Get detailed information about a note pad

- Can filter by note pad name/pattern
- Can filter by CF structure name

Use D XCF,STR,STRNAME=IXCNP_* to list note pad structures

Use D XCF,NP to get information about the note pads. The SCOPE keyword determines whether summary information or detailed information is to be provided. You can specify a note pad name or a note pad name pattern to get information for one specific note pad or a set of note pads. You can specify a structure name to limit the output to just those note pads hosted by the indicated structure.

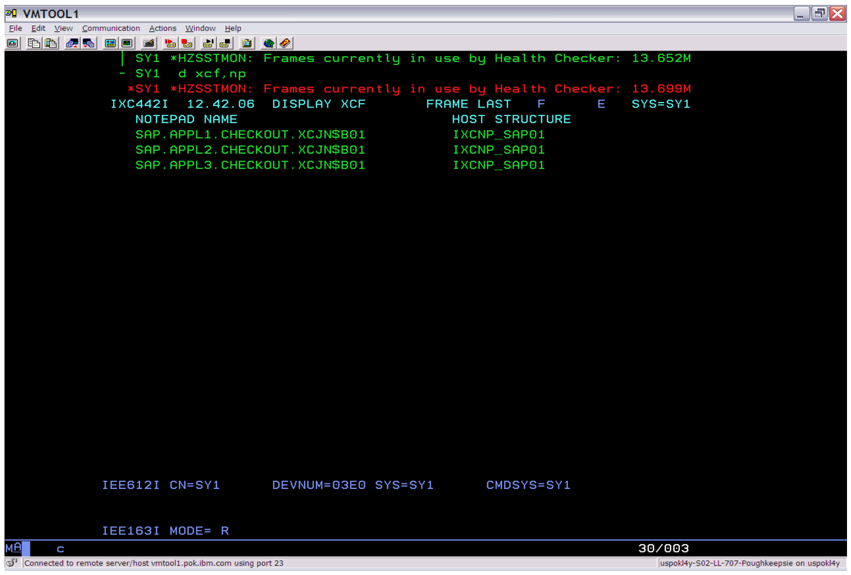
Examples:

```
 d xcf,np,npname=s.y,scope=sum
 d xcf,np,strname=sy,scope=sum
 d xcf,np,strname=sy,npname=a.b
 d xcf,np,strname=sy,npname=a.b,scope=sum
 d xcf,np,strname=sy,npname=a.b,scope=det
 d xcf,np,npname=a.b
 d xcf,np,strname=ralph
 d xcf,np,strname=ralph,scope=det
 d xcf,notepad
 d xcf,notepad,scope=sum
 d xcf,notepad,npname=all
 d xcf,notepad,strname=all
```

The DISPLAY XCF,STR command is an existing command used to get information about a given structure. Since all note pad names begin with the prefix "IXCNP_", you could specify STRNAME=IXCNP_* to get information about each of the note pad structures. Of course you could specify a specific structure name to get information about that one note pad structure.

XCF Note Pad

D XCF,NP



```
           | SY1 *HZSSTMON: Frames currently in use by Health Checker: 13.652M
           - SY1  d xcf,np
            *SY1 *HZSSTMON: Frames currently in use by Health Checker: 13.699M
          IXC442I  12.42.06  DISPLAY XCF        FRAME LAST   F       E   SYS=SY1
             NOTEPAD NAME                          HOST STRUCTURE
             SAP.APPL1.CHECKOUT.XCJN$B01           IXCNP_SAP01
             SAP.APPL2.CHECKOUT.XCJN$B01           IXCNP_SAP01
             SAP.APPL3.CHECKOUT.XCJN$B01           IXCNP_SAP01

         IEE612I CN=SY1       DEVNUM=03E0 SYS=SY1       CMDSYS=SY1

         IEE163I MODE= R
MA    c                                                                30/003
```

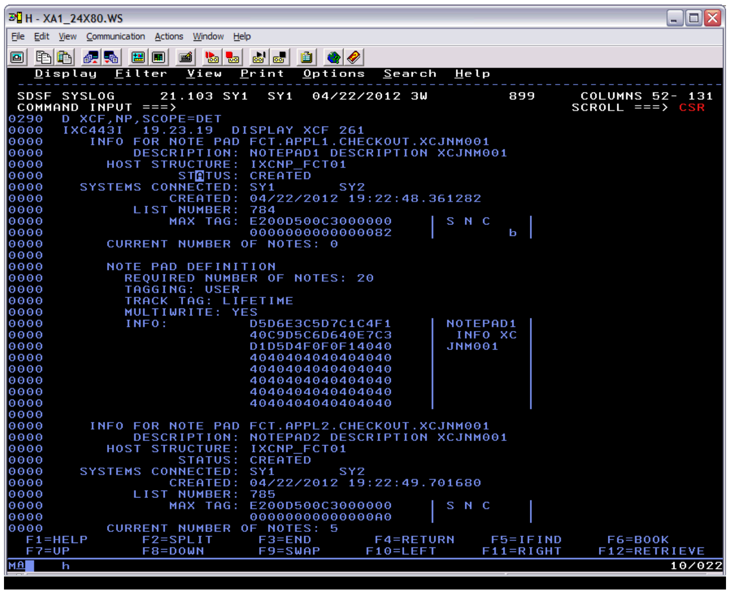
VMTOOL1
File Edit View Communication Actions Window Help
Connected to remote server/host vmtool1.pok.ibm.com using port 23
uspokl4y-S02-LL-707-Poughkeepsie on uspokl4y

49 © 2013 IBM Corporation

Sample output from the DISPLAY XCF,NOTEPAD command. Lists the names of the note pads that are currently defined along with the note pad structure that hosts the note pad. You might issue D XCF,NP,STRNAME=*strname* to get a list of all the note pads defined in the structure named *strname*.

Despite all appearances, the indicated names are not actual SAP note pad names. These were generated by XCF testers looking to verify that we were correctly hosting the note pads in an owner specific note pad. So perhaps it is worth pointing out the “owner” portion of these note pad names is SAP and the host structure IXCNP_SAP01 is an owner specific note pad.

D XCF,NP,SCOPE=DETAIL


H - XA1_24X80.WS
File Edit View Communication Actions Window Help
Display Filter View Print Options Search Help

```
 SDSF SYSLOG      21.103 SY1   SY1   04/22/2012 3W         899      COLUMNS 52- 131
 COMMAND INPUT ===>                                                 SCROLL ===> CSR
0290  D XCF,NP,SCOPE=DET
0000  IXC443I  19.23.19  DISPLAY XCF 261
0000     INFO FOR NOTE PAD FCT.APPL1.CHECKOUT.XCJNM001
0000          DESCRIPTION: NOTEPAD1 DESCRIPTION XCJNM001
0000       HOST STRUCTURE: IXCNP_FCT01
0000               STATUS: CREATED
0000    SYSTEMS CONNECTED: SY1      SY2
0000              CREATED: 04/22/2012 19:22:48.361282
0000          LIST NUMBER: 784
0000              MAX TAG: E200D500C3000000    | S N C     |
0000                       0000000000000082    |        b  |
0000       CURRENT NUMBER OF NOTES: 0
0000
0000       NOTE PAD DEFINITION
0000         REQUIRED NUMBER OF NOTES: 20
0000         TAGGING: USER
0000         TRACK TAG: LIFETIME
0000         MULTIWRITE: YES
0000         INFO:         D5D6E3C5D7C1C4F1    | NOTEPAD1  |
0000                       40C9D5C6D640E7C3    |  INFO XC  |
0000                       D1D5D4F0F0F14040    | JNM001    |
0000                       4040404040404040    |           |
0000                       4040404040404040    |           |
0000                       4040404040404040    |           |
0000                       4040404040404040    |           |
0000                       4040404040404040    |           |
0000
0000     INFO FOR NOTE PAD FCT.APPL2.CHECKOUT.XCJNM001
0000          DESCRIPTION: NOTEPAD2 DESCRIPTION XCJNM001
0000       HOST STRUCTURE: IXCNP_FCT01
0000               STATUS: CREATED
0000    SYSTEMS CONNECTED: SY1      SY2
0000              CREATED: 04/22/2012 19:22:49.701680
0000          LIST NUMBER: 785
0000              MAX TAG: E200D500C3000000    | S N C     |
0000                       00000000000000A0    |           |
0000       CURRENT NUMBER OF NOTES: 5
  F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=IFIND     F6=BOOK
  F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
MA    h                                                                   10/022
```

This slide depicts the output of a DISPLAY XCF,NOTEPAD command requesting detailed information about the note pad. For a given note pad, the first part of the output identifies the note pad and provides status information. The second portion lists the remainder of the various parameter specifications from the IXCNOTE request that was used to create the note pad. Message IXC443I provides the information.

For the note pad named FCT.APPL1.CHECKOUT.XCJNM001, we see the application provided note pad description and the name of the note pad structure that hosts it (IXCNP_FCT01). The current status of the note pad is created, indicating that XCF has successfully finished creating the note pad. The status could also indicate, for example, that the note pad was still in the midst of being created. In that case, the host structure would be the name of the candidate note pad structure that is currently being considered as a host. Until the status indicates that the note pad is created, the indicated host structure could change. A status of delete pending would indicate that the note pad is in the midst of being deleted.

The systems connected lists SY1 and SY2, indicating that each of these systems has at least one connector to the note pad. There could be more than one connector on one or more of the indicated systems. The created timestamp indicates when the note pad was deemed to have been created. This TOD can be used to identify a unique instance of the note pad.

Within the indicated host note pad, list number 784 is being used for this note pad. The current maximum tag value for the note pad is shown. Finally a count of the number of notes currently residing in the note pad is provided. As described later in the presentation, the delete of a note might be pending. Such a note is still included in the current number of notes.

The remainder of the information describes the note pad definition. The output indicates the number of notes that the note pad needs to support, whether XCF or the user is responsible for assigning note tag values, whether the maximum tag value is to be tracked, whether more than one connection is permitted to create, update, or delete notes in the note pad. <Sorry, this is actually down level output. Subsequent to this test run, we added a new line after MULTIWRITE to indicate whether the creator of the note pad specified DUPLEX=AVOID or DUPLEX=FAVOR.> Finally, the 64 bytes of application provided data referred to as INFO is reported in both hexadecimal and EBCDIC format.

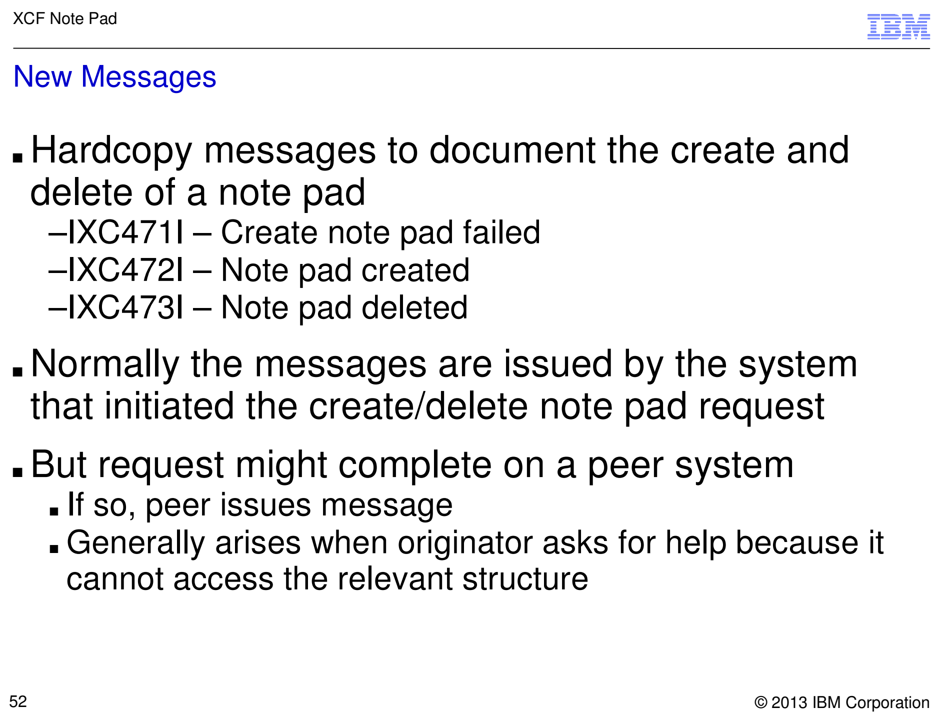
The display output then provides information for each of the remaining note pads that was selected.

Display Messages

- Messages related to DISPLAY XCF,NOTEPAD

- IXC441I – bad note pad name filter specification
- IXC442I – summary output
- IXC443I – detail output
- IXC444I – nothing matched what you asked for
- IXC445I – command failed
- IXC328I – bad syntax, add insert for bad note pad name/pattern

The messages in blue are issued in response to the DISPLAY XCF,NOTEPAD command. IXC328I is an old message, but updated with a new insert.



New Messages

- Hardcopy messages to document the create and delete of a note pad
  - –IXC471I – Create note pad failed
  - –IXC472I – Note pad created
  - –IXC473I – Note pad deleted
- Normally the messages are issued by the system that initiated the create/delete note pad request
- But request might complete on a peer system
  - If so, peer issues message
  - Generally arises when originator asks for help because it cannot access the relevant structure

The indicated messages are new. Messages IXC471I and IXC472I are issued in response to a create note pad request. Message IXC473I issued in response to a delete note pad request.

Message IXC471I explains why XCF was unable to create a note pad due to issues related to the coupling facility structures. The message indicates the name of the note pad, the name of the system that issued the request, the name of the job that issued the request, and an explanation as to what the problem was and with which structure. If multiple note pad structures were considered, the message will list each one and explain why it could not be used to host the note pad. A create note pad request could be rejected for a variety of reasons. For example: note pad already exists, invalid parameters, not authorized to create the note pad, etc. The message is not issued for those sorts of problems. This message is issued to explain failures arising from conditions that the IXCNOTE macro describes as “no structure resources” (return code x’0C’ reason ‘xxxx0CA5’).

Message IXC472I is issued when a note pad is created. In general there will be one instance of the message issued for the note pad to indicate that it was successfully created. However, there are cases where a system might be able to define the note pad but is unable to finish instantiating it in a host note pad structure. Typically these cases arise when the system experiences a loss of connectivity to a structure while in the midst of creating the note pad. In these cases, the system issues message IXC472I with an indication that the note pad is being created. The note pad is logically defined but is not yet usable. To the application, the note pad remains quiesced and inaccessible until XCF is able to finish creating the note pad (or until the note pad fails or is otherwise deleted). If a system is unable to finish creating the note pad, it will ask other systems in the sysplex for help. If one of those systems is able to finish creating the note pad, that system will issue message IXC472I to indicate that the note pad has finally been created. Alternatively, connectivity to the relevant structure might be restored, in which case the local system might be the one to finish creating the note pad and in so doing, issue message IXC472I to indicate that the note pad was created and available for use. Message IXC472 indicates the name of the note pad, the name of the system that initiated the create note pad request, the name of the job that issued the request, the time of day when the note pad was created (which can be used to uniquely identify an instance of a note pad), the number of notes in the note pad, and the name of the structure selected to host the note pad.

Message IXC473I issued to indicate that a note pad has been deleted. Normally the delete of the note pad can be completed and the message so indicates. Again, it might be the case that the system loses access to the relevant structures during the delete. In such cases message IXC473I indicates that the note pad is being deleted. From the perspective of the application the note pad is deleted and a new instance could be created. However, the physical resources associated with the note pad might still exist. Other systems in the sysplex could be asked to finish deleting the note pad. When cleanup of the resources is finally complete, message IXC473I is issued to so indicate. The message indicates the name of the note pad, the name of the system and the name of the job that initiated the delete request, the time of day when the note pad was created (to uniquely identify the physical instance of the note pad that is being deleted), as well as the reason the note pad was deleted. Typically the note pad is deleted because of an application request. But it will also be deleted, for example, if the host structure is deallocated.

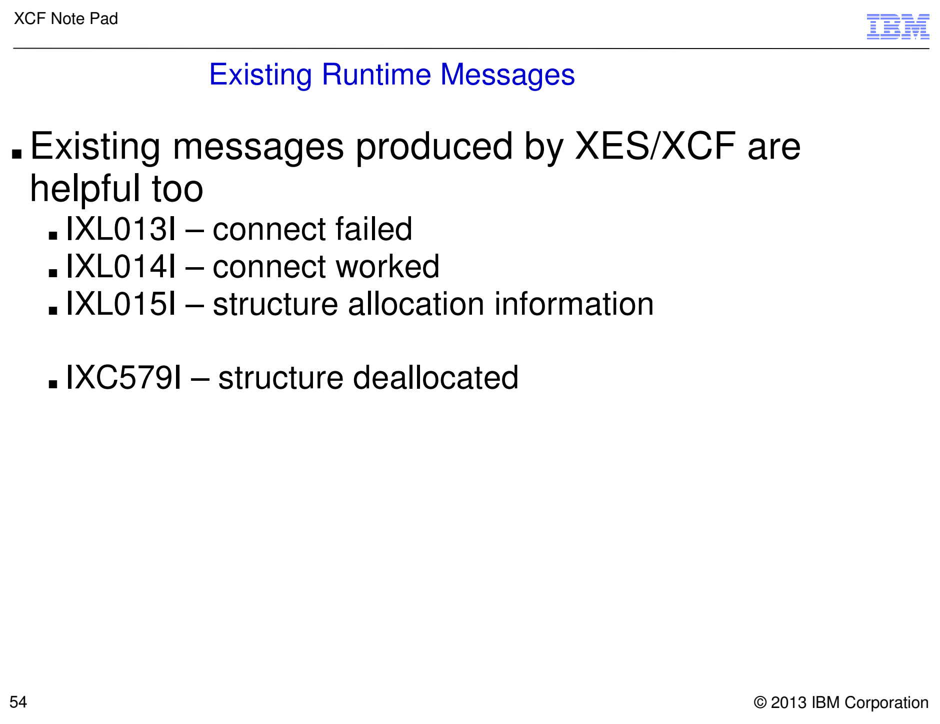
- Hardcopy messages to document the create and delete of a note pad
  - –IXC471I – Create note pad failed
  - –IXC472I – Note pad created
  - –IXC473I – Note pad deleted
- Normally the messages are issued by the system that initiated the create/delete note pad request
- But request might complete on a peer system
  - If so, peer issues message
  - Generally arises when originator asks for help because it cannot access the relevant structure

The indicated messages are new. Messages IXC471I and IXC472I are issued in response to a create note pad request. Message IXC473I issued in response to a delete note pad request.

Message IXC471I explains why XCF was unable to create a note pad. A create note pad request could be rejected for a variety of reasons. For example: note pad already exists, invalid parameters, not authorized to create the note pad, etc. The message is not issued for those sorts of problems. It is issued to explain failures arising from conditions that the IXCNOTE macro describes as "no structure resources" (return code x'0C' reason 'xxxx0CA5'). The message indicates the name of the note pad, the name of the system that issued the request, the name of the job that issued the request, and an explanation as to what the problem was and with which structure. If multiple note pad structures were considered, the message will list each one and explain why it could not be used to host the note pad.

Message IXC472I is issued when a note pad is created. In general there will be one instance of the message issued for the note pad to indicate that it was successfully created. However, there are cases where a system might be able to define the note pad but is unable to finish instantiating it in a host note pad structure. Typically these cases arise when the system experiences a loss of connectivity to a structure while in the midst of creating the note pad. In these cases, the system issues message IXC472I with an indication that the note pad is being created. The note pad is logically defined but is not yet usable. To the application, the note pad remains quiesced and inaccessible until XCF is able to finish creating the note pad (or until the note pad fails or is otherwise deleted). If a system is unable to finish creating the note pad, it asks other systems in the sysplex for help. If one of those systems can finish creating the note pad, it issues message IXC472I to indicate that the note pad has finally been created. Alternatively, connectivity to the relevant structure might be restored, in which case the local system might finish creating the note pad and issue message IXC472I to indicate that the note pad was created. Message IXC472 indicates the name of the note pad, the name of the system that initiated the create note pad request, the name of the job that issued the request, the time of day when the note pad was created (which can be used to uniquely identify an instance of a note pad), the number of notes requested for the note pad, and the name of the host structure.

Message IXC473I issued to indicate that a note pad has been deleted. Normally the delete of the note pad can be completed and the message so indicates. Again, it might be the case that the system loses access to the relevant structures during the delete. In such cases message IXC473I indicates that the note pad is being deleted. From the perspective of the application the note pad is deleted and a new instance could be created. However, the physical resources associated with the note pad might still exist. Other systems in the sysplex could be asked to finish deleting the note pad. When cleanup of the resources is finally complete, message IXC473I is issued to so indicate. The message indicates the name of the note pad, the name of the system and the name of the job that initiated the delete request, the time of day when the note pad was created (to uniquely identify the physical instance of the note pad that is being deleted), as well as the reason the note pad was deleted. Typically the note pad is deleted because of an application request. But it will also be deleted, for example, if the host structure is deallocated.

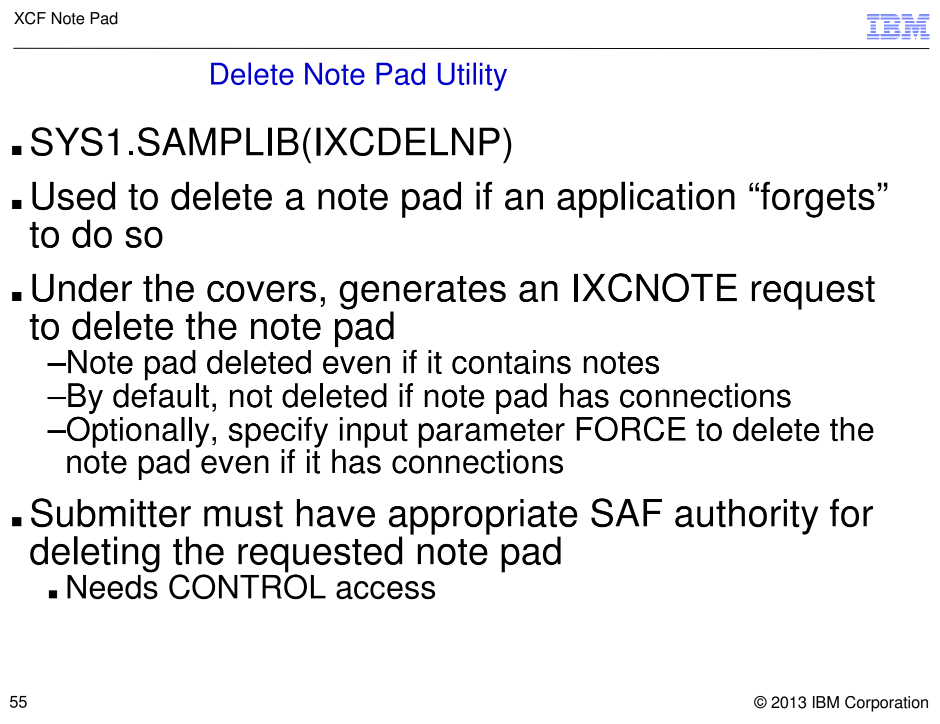


Existing Runtime Messages

- Existing messages produced by XES/XCF are helpful too
  - IXL013I – connect failed
  - IXL014I – connect worked
  - IXL015I – structure allocation information

  - IXC579I – structure deallocated

XES and XCF issue messages related to structures and structure connectors. These existing messages will be issued as XCF Note Pad Services establishes connections (via IXLCONN macro) to note pad structures and as note pad structures are created. Message IXL015I is issued when a successful connect causes an instance of the structure to be allocated.



Delete Note Pad Utility

- SYS1.SAMPLIB(IXCDELNP)
- Used to delete a note pad if an application “forgets” to do so
- Under the covers, generates an IXCNOTE request to delete the note pad
  - –Note pad deleted even if it contains notes
  - –By default, not deleted if note pad has connections
  - –Optionally, specify input parameter FORCE to delete the note pad even if it has connections
- Submitter must have appropriate SAF authority for deleting the requested note pad
  - Needs CONTROL access

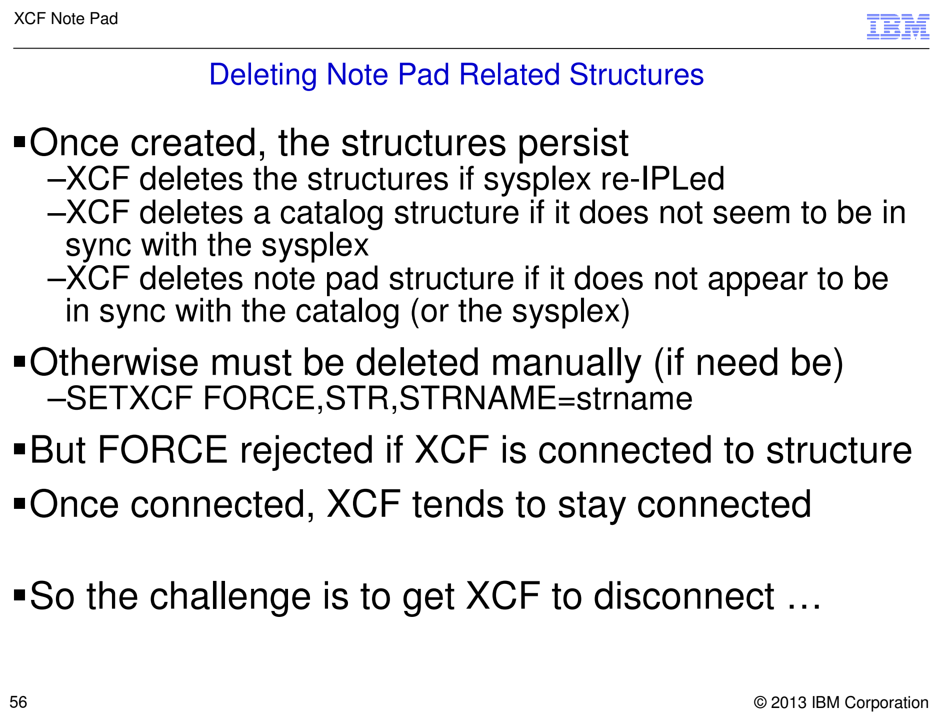
Applications expect the note pad to persist even if no system has a connection to the note pad. The onus is on the application to delete the note pad when it is no longer needed. But there may be occasions where the application fails to delete the note pad. If you are certain that the note pad is no longer needed, you can use the delete utility to delete the note pad.

The application needs to document the conditions under which it is safe to have the note pad deleted. You would not want to delete a note pad if it would result in severe negative consequences.

The delete utility is a program supplied by IBM. See member IXCDELNP in SYS1.SAMPLIB. You supply the name of the note pad to be deleted and submit the job. The user that submits the job must be authorized to delete the note pad, which is to say the user needs CONTROL access to the FACILITY class resource IXCNOTE.owner.application where “owner” and “application” are derived from the note pad name.

The utility will delete the note pad even if it contains notes. By submitting the utility, you tacitly indicate that the content of the note pad is no longer needed. If you only want to delete the note pad if it is empty, you can use the DISPLAY XCF,NP,NOTEPAD=*notepadname* to see whether the note pad has any notes (or connections for that matter). However, be aware that there is a time of check to time of use window. You would need to ensure that the application cannot create any new notes between the time you observed that the note pad was empty and the time you submit the delete utility.

By default, the utility will not delete the note pad if it has connections. The existence of a note pad connection suggests that the note pad is actively in use. If you truly want to rip the note pad out from under its users, you can specify the FORCE option to make that happen. Ongoing or subsequent note requests issued by the active users will be rejected. The connections will be deleted along with the note pad.

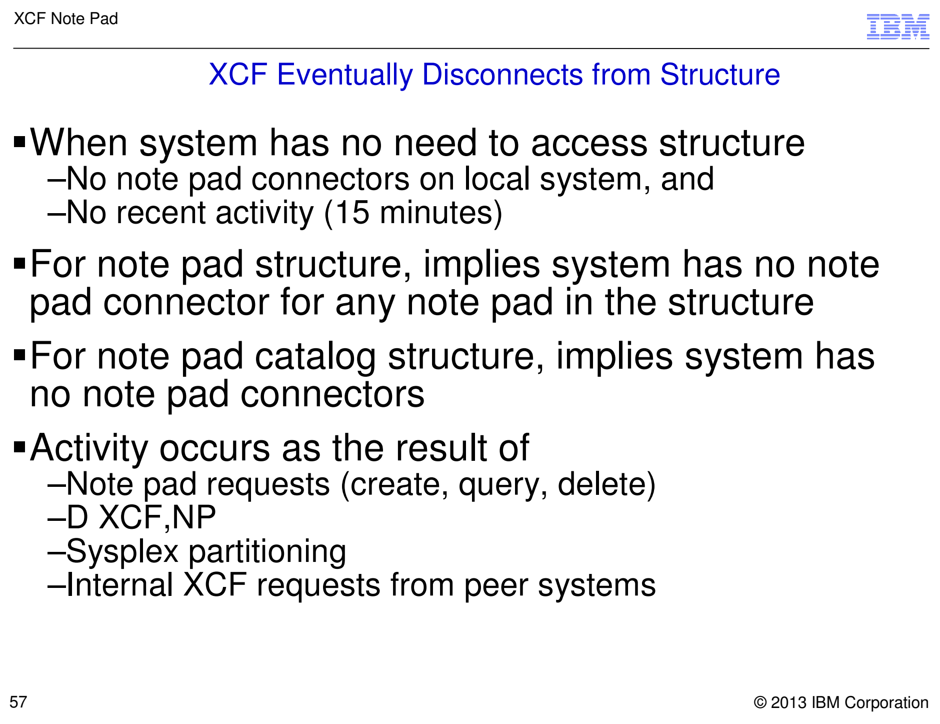


Deleting Note Pad Related Structures

- Once created, the structures persist
  - –XCF deletes the structures if sysplex re-IPLed
  - –XCF deletes a catalog structure if it does not seem to be in sync with the sysplex
  - –XCF deletes note pad structure if it does not appear to be in sync with the catalog (or the sysplex)
- Otherwise must be deleted manually (if need be)
  - –SETXCF FORCE,STR,STRNAME=strname
- But FORCE rejected if XCF is connected to structure
- Once connected, XCF tends to stay connected
- So the challenge is to get XCF to disconnect …

There may come a time when you decide that you want to delete a note pad structure, and perhaps even the note pad catalog. XCF tells XES to create these structures as "persistent structures". A persistent structure remains in existence even if it has no connections (IXLCONN). Barring events such as the outright failure of the host coupling facility, the structure will persist until it is explicitly deleted. XCF does not delete these structures during the normal course of operation. It will delete the structure as the result of error events such as re-IPL of the sysplex or in cases where it appears that the catalog structure and the note pad structure are not self-consistent. So in general, that means you must take explicit action to delete these structures.

The existing SETXCF FORCE,STR,STRNAME=*strname* command is used to delete a structure. However it should be noted that this command is rejected if the structure has any connectors. If it works, well and good. But if it is rejected because connectors exist, and you really want to get the structure deleted, you will need to get XCF to disconnect from the structure. So you need to understand what causes XCF to establish connections to these structures, and when XCF will disconnect from the structures. There is no command to make XCF disconnect. So XCF has to be encouraged to disconnect of its own accord.



XCF Note Pad

XCF Eventually Disconnects from Structure

- When system has no need to access structure
  - –No note pad connectors on local system, and
  - –No recent activity (15 minutes)
- For note pad structure, implies system has no note pad connector for any note pad in the structure
- For note pad catalog structure, implies system has no note pad connectors
- Activity occurs as the result of
  - –Note pad requests (create, query, delete)
  - –D XCF,NP
  - –Sysplex partitioning
  - –Internal XCF requests from peer systems

57 © 2013 IBM Corporation

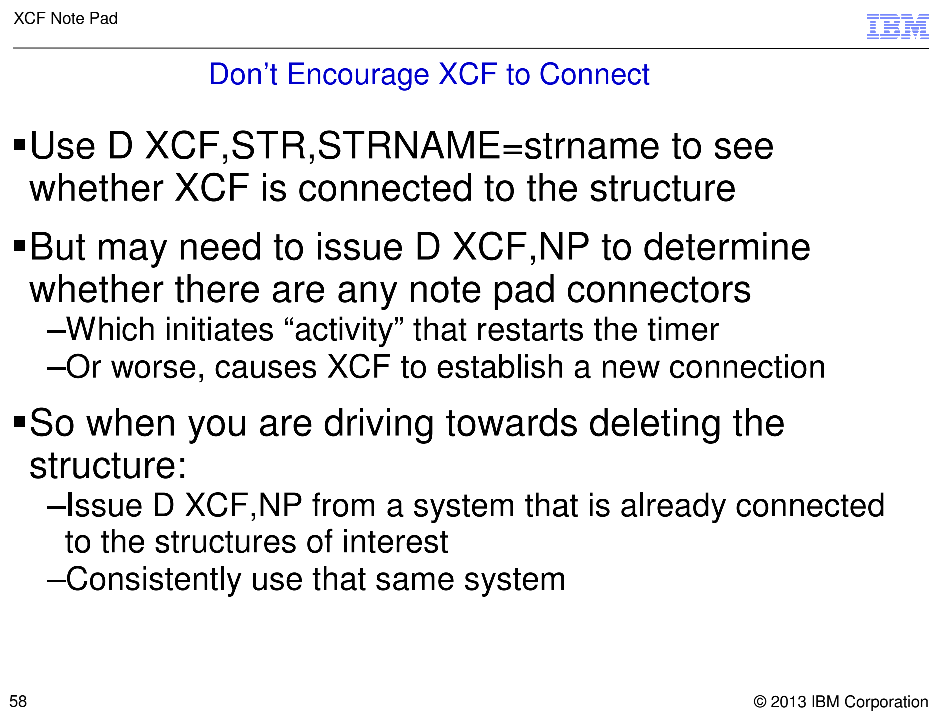
Once we establish a connection to a structure, we tend to want to keep it. Connections are relatively expensive to establish and tear down. So we don’t want to do that on a frequent basis – as might occur if we were to connect and disconnect on demand. So the general rule is that any given system will remain connected to a note pad structure and the note pad catalog structure so long as that system has an application with a connection to a note pad in the given note pad structure.

So if you want XCF to disconnect from its structures, you will first need to shut down the applications that are connected to note pads in those structures. You would also need to ensure that no new instances of applications that exploit note pads are started in the sysplex.

If a system does not have any applications with connections to note pads in a given note pad structure, that system will disconnect from the note pad structure approximately 15 minutes after the last occasion XCF had to access the structure. There are a variety of things that can cause a system to access a note pad structure, application requests being the most obvious. So if you want XCF to disconnect from the note pad structure, you would need to make sure that no new note pad applications are started on the system. DISPLAY XCF,NP can induce structure accesses since the query note pad processing used to provide the information for the display output may try to read information from the note pad structure. Sysplex partitioning can cause structure accesses as surviving systems perform cleanup processing on behalf of their now departed peer. Finally, if a system does not have connectivity to a given note pad structure, it may ask a peer system to perform some task on its behalf. That request can induce a structure accesses. Perhaps worse, if the system does not currently have a connection to the relevant structure, it will in fact connect to the structure in order to accomplish the task.

The discussion above focused on note pad structures. But it applies to the catalog structure as well. You should assume that any request that induces access to a note pad structure is likely going to cause the catalog structure to be accessed as well. If you want to get XCF to disconnect from the catalog structure, you will have to get XCF to disconnect from all the note pad structures first.

For completeness, I should mention that XCF will disconnect from a structure if the system loses connectivity to the coupling facility that hosts the structure. Furthermore, if XCF disconnects from the catalog structure, it will also disconnect from the note pad structures as well. However, I rather doubt that you will be entertaining the notion of pulling cables to lose connectivity to a coupling facility in order to get XCF to disconnect from its structures.



XCF Note Pad

Don’t Encourage XCF to Connect

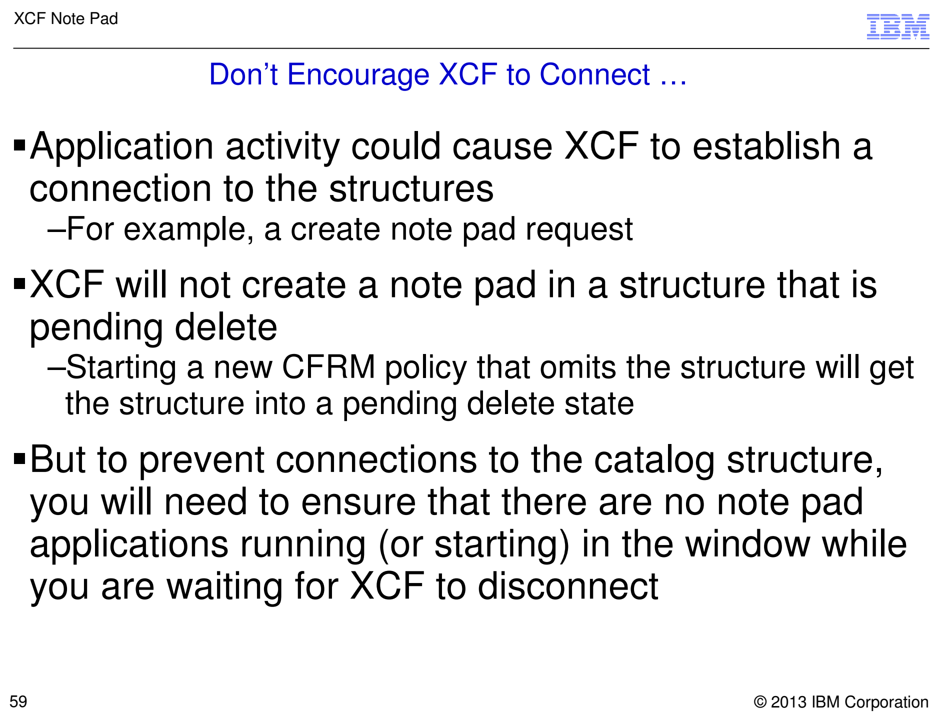
- Use D XCF,STR,STRNAME=strname to see whether XCF is connected to the structure
- But may need to issue D XCF,NP to determine whether there are any note pad connectors
  - Which initiates “activity” that restarts the timer
  - Or worse, causes XCF to establish a new connection
- So when you are driving towards deleting the structure:
  - Issue D XCF,NP from a system that is already connected to the structures of interest
  - Consistently use that same system

58

© 2013 IBM Corporation

So if you are trying to get XCF to disconnect from either a note pad structure or the note pad catalog structure, you do not want to perform any actions that will cause XCF to access the structures. Any such accesses will in effect cause XCF to restart its 15 minute “no activity” timer. In particular, note that the DISPLAY XCF,NP will cause XCF to access the catalog structure for sure, and probably the note pad structures as well. This can get us into a sort of catch-22 situation where we might need to get information to see where things stand only to have the act of getting the information keep the connection alive.

So I would first start with issuing the SETXCF FORCE,STR command and see if it works. If it is rejected because the structure still has connections, issue the DISPLAY XCF,STR,STRNAME=*strname* command to see which systems have connections. You will likely need to issue the DISPLAY XCF,NOTEPAD command to see which note pads have connections so that you can shut down the appropriate applications. Since the D XCF,NP command induces “activity”, you should issue that command consistently from the same system and that system should already have a connection to the structure. That way the “activity” only occurs on one system. You certainly don’t want to issue the command from a system that does not have connectivity to the structure because XCF will then establish a connection in order to gather the information for the display.



XCF Note Pad

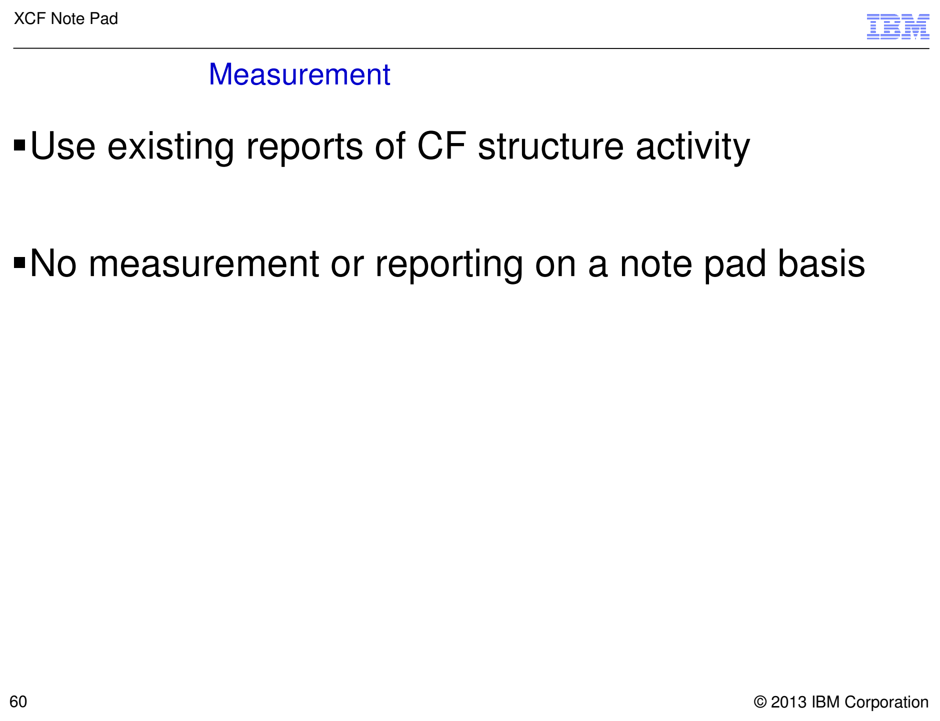
Don’t Encourage XCF to Connect …

- Application activity could cause XCF to establish a connection to the structures
  - For example, a create note pad request
- XCF will not create a note pad in a structure that is pending delete
  - Starting a new CFRM policy that omits the structure will get the structure into a pending delete state
- But to prevent connections to the catalog structure, you will need to ensure that there are no note pad applications running (or starting) in the window while you are waiting for XCF to disconnect

59 © 2013 IBM Corporation

“Activity” could arise form application requests. If you are trying to delete a note pad structure, you don’t want application activity to cause XCF to try to access the structure since such accesses will cause XCF to create and/or maintain its connection to the structure. If you update the Coupling Facility Resource Management (CFRM) policy so that the structure is no longer defined in the policy, the structure will become pending delete. If a note pad structure is pending delete, XCF Note Pad Services will not create any new note pads in that structure. So if an application tries to create a new note pad, a structure that is pending delete will not be included in the list of candidate structures. Thus XCF would not need to create a connection to the structure, and if it already has a connection, the attempt to create the note pad would not cause XCF to try to access the structure. So removing the structure from the CFRM policy could be helpful (perhaps required) for preventing accesses (activity) for a note pad structure.

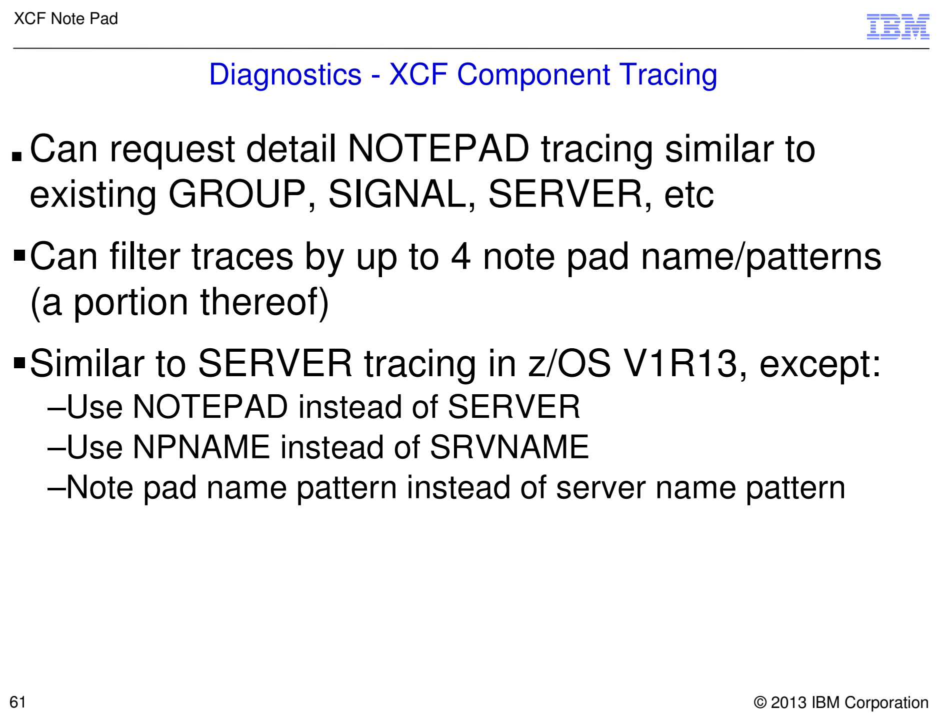
This same trick does not really help for the catalog structure. If the catalog is needed, it will be used regardless of whether it is pending delete from the CFRM policy perspective. So if you are trying prevent XCF from accessing the catalog structure, you might have to ensure that there are no note pad applications running.



Measurement

- Use existing reports of CF structure activity
- No measurement or reporting on a note pad basis

The existing Resource Measurement Facility (RMF) reports of CF structure activity, or similar reports from an equivalent product of your choice, can be used to monitor use of note pad structures. We do not have any measurements or reporting for individual note pads within those note pad structures. I’m wondering if that level of detail would really be needed. Please speak up if you want it.



Diagnostics - XCF Component Tracing

- Can request detail NOTEPAD tracing similar to existing GROUP, SIGNAL, SERVER, etc
- Can filter traces by up to 4 note pad name/patterns (a portion thereof)
- Similar to SERVER tracing in z/OS V1R13, except:
  - Use NOTEPAD instead of SERVER
  - Use NPNAME instead of SRVNAME
  - Note pad name pattern instead of server name pattern

FYI. I suspect most installations would not need to bother with setting up XCF component trace options unless they happen to be trying to diagnose a note pad related problem. In that case, IBM service personnel would be involved in providing some direction here. So the point of this slide is to just make you aware that there is detailed tracing for note pad related activity. The XCF traces can be filtered so that only those traces related to a specific note pad or the set of note pads whose name matches a given pattern will be recorded. Such filtering is useful if you want to exclude traces for note pads that are not related to the investigation.

Commands – NPNAME Filter on TRACE CT for SYSXCF

- NPNAME
  - Filters NOTEPAD traces based on note pad name
  - Not case sensitive (in contrast to SERVER and SRVNAME)
  - Four (4) note pad names or patterns can be active

```
 TRACE CT,ON,COMP=SYSXCF
*0021 ITT006A SPECIFY OPERAND(S) FOR TRACE CT COMMAND.
 R 21,OPTIONS=(NPNAME=(SYS*.ABC*,XYZ.FRED*)),END


 ------------------------------------------------
SYSXCF        ON   0003M  HEAD    0
   ASIDS      *NOT SUPPORTED*
   JOBNAMES   *NOT SUPPORTED*
   OPTIONS    NPNAME=(SYS*.ABC*,XYZ.FRED*)
   WRITER     *NONE*
```

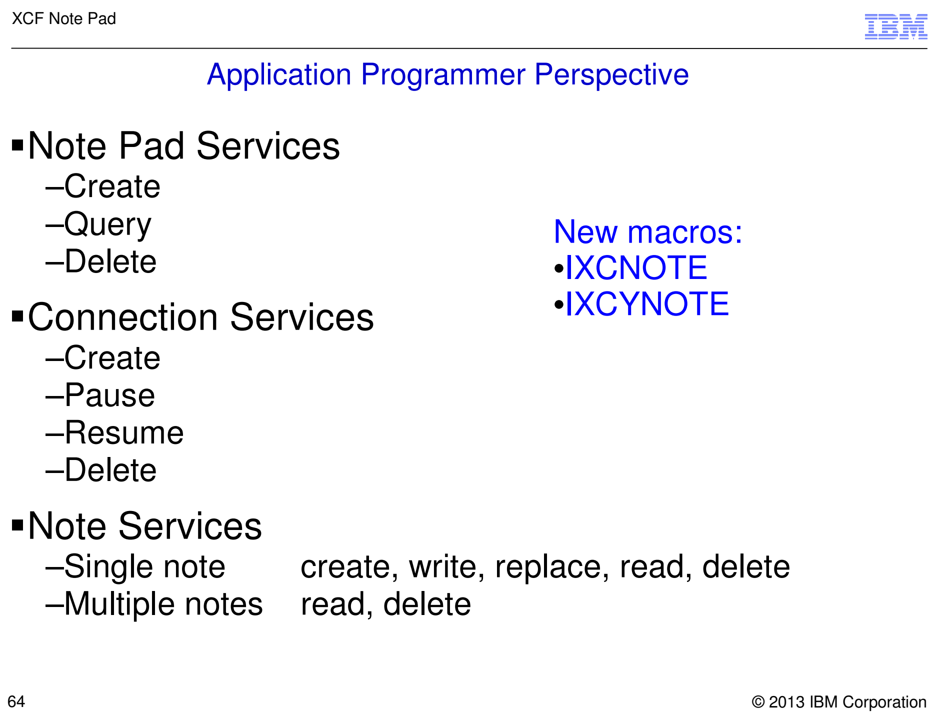
An example to illustrate use of the TRACE CT command to have XCF Component Trace entries be filtered by note pad name. In this case, the traces associated with a particular note pad would only be cut if the name of the note pad name matched the indicated pattern.

CTRACE Messages

- Error messages related to CTRACE OPTIONS syntax checking

- IXC441I – bad note pad name filter specification
- IXC375I – add NOTEPAD and NPNAME insert to list of expected keywords

Existing message IXC375I, which is issued when the TRACE CT command appears to have invalid specifications for XCF Component Tracing, was updated to include the new keywords for note pad support. The message provides a list of expected keywords.
Message IXC441I is a new message issued to complain about invalid note pad name/pattern input.



Application Programmer Perspective

- Note Pad Services
  - –Create
  - –Query
  - –Delete
- Connection Services
  - –Create
  - –Pause
  - –Resume
  - –Delete
- Note Services

–Single note	create, write, replace, read, delete
–Multiple notes	read, delete

New macros:
- •IXCNOTE
- •IXCYNOTE

The XCF note pad macro (IXCNOTE) encapsulates the interfaces that allow an application to interact with the XCF Note Pad services. Macro IXCYNOTE has mappings for various data areas related use of the XCF Note Pad Services. With these interfaces, an application can create or delete a "Note Pad". A Note Pad contains zero or more notes. A "note" contains up to 1024 bytes of application data and is identified by an application provided "name". An application that is "connected" to a Note Pad can create, read, modify, or delete notes in the Note Pad.

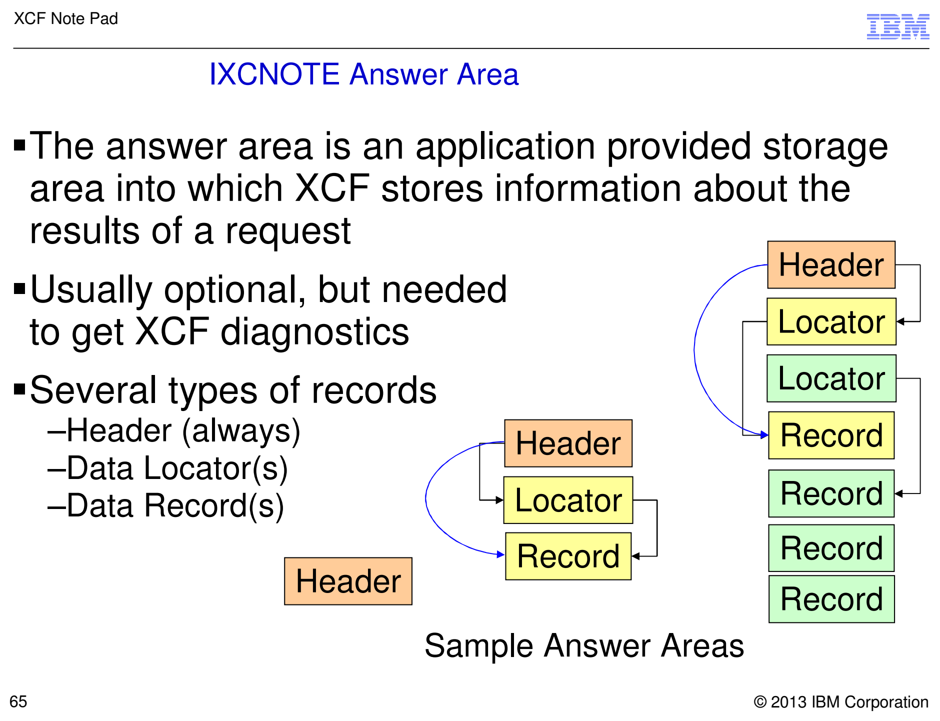
We can group the services into three categories: those that are directed against the note pad as a whole, those related to note pad connections, and those that are used to manipulate notes in a note pad.

Note Pad Services are used to create a new note pad, get information about a note pad, or to delete an existing note pad.

Connection Services are used to create a connection to an existing note pad, or to delete a connection. If a note pad becomes inaccessible, the note pad connection is quiesced. It will not be possible to manipulate notes in the note pad until the note pad is once again accessible. A connector uses the pause service to discover when the note pad becomes accessible. A connector uses the resume service to interrupt the pause service prematurely.

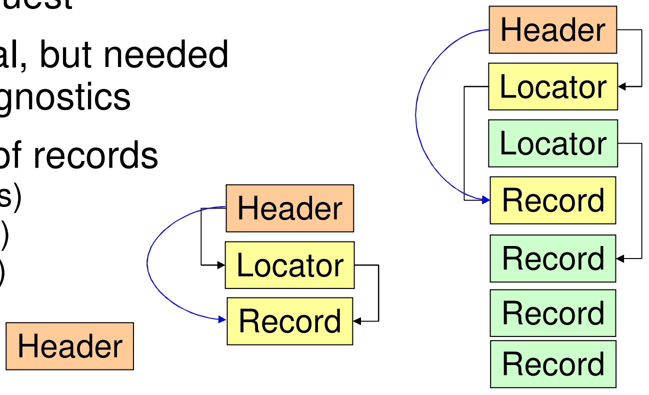
Note services are used to manipulate notes in the note pad. An individual note can be created or deleted. The content of an existing note can be replaced or read. A write request will create a note if it does note currently exist, and replace its content if it does exist.

We discuss these services in subsequent slides. However, we discuss them in the order that an application might use them. So we will create the note pad, create a connection, process notes, delete the connection, and delete the note pad.



IXCNOTE Answer Area

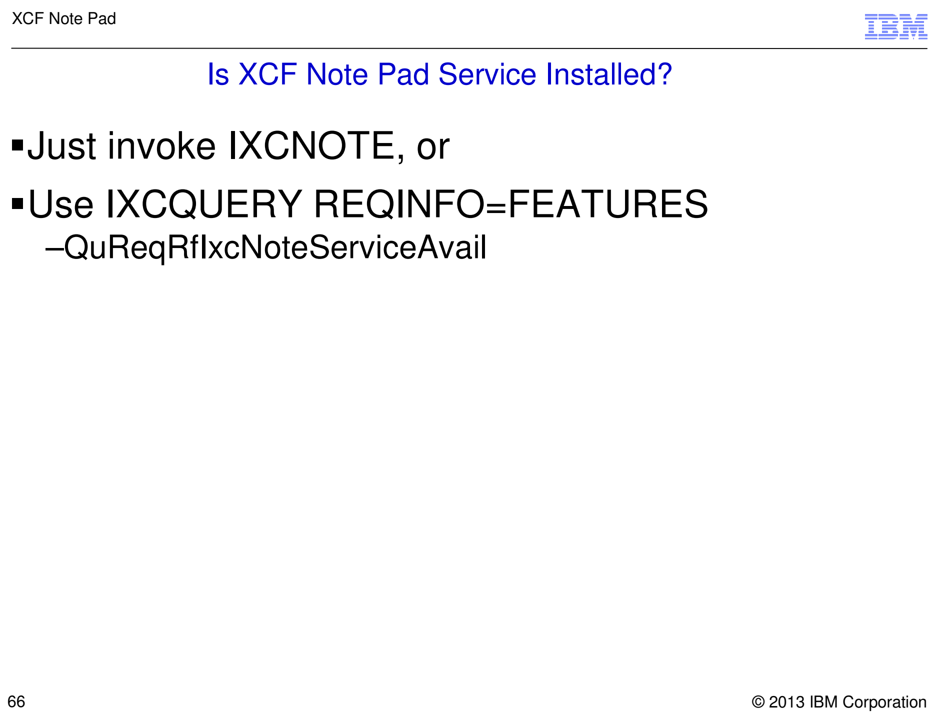
- The answer area is an application provided storage area into which XCF stores information about the results of a request
- Usually optional, but needed to get XCF diagnostics
- Several types of records
  - –Header (always)
  - –Data Locator(s)
  - –Data Record(s)



Sample Answer Areas

The storage for the answer area is provided by your program. For most requests, the answer area is optional. If provided, the answer area always contains a header record. Sometimes the header record is the only information stored in the answer area. If additional information is stored, the answer area header indicates where an array of data locator records can be found within the answer area. Depending on the request and the size of the answer area, zero or more data locator records are provided. Each entry in the data locator array identifies the location of an array of data records. These data records contain the functional output of the request. Both the data locators and the data records themselves are stored in the answer area. If the request is rejected, diagnostic data might be stored in the answer area header to provide detailed information about the failure. An answer area is optional for most requests, but desirable given the potential need for this detailed diagnostic data.

In the slide we depict three sample answer areas. One consists of the header alone. The second shows an answer area header followed by one data locator and one data record. In the third we see the header record followed by two data locators, followed by several data records of which there are two types (depicted as yellow and green). The header indicates how many data locators were stored in the answer area and the offset from the start of the header at which the array of locators is found. Each data locator provides information about one particular type of data record. The locator indicates the type of record, the size of one record, the number of records, and the offset at which the first of the records was stored. When the answer has data records, one of the records is deemed to be the “expected” or “primary” record associated with the request results. As depicted in the slide by the blue arrow, the header also indicates the offset (from the beginning of the header) at which this primary data record can be found in the answer area. This offset is intended as a convenience so that applications that might be solely interested in the primary data record need not have to decode the data locator to find the record.



XCF Note Pad

Is XCF Note Pad Service Installed?

- Just invoke IXCNOTE, or
- Use IXCQUERY REQINFO=FEATURES
  - QuReqRfIxcNoteServiceAvail

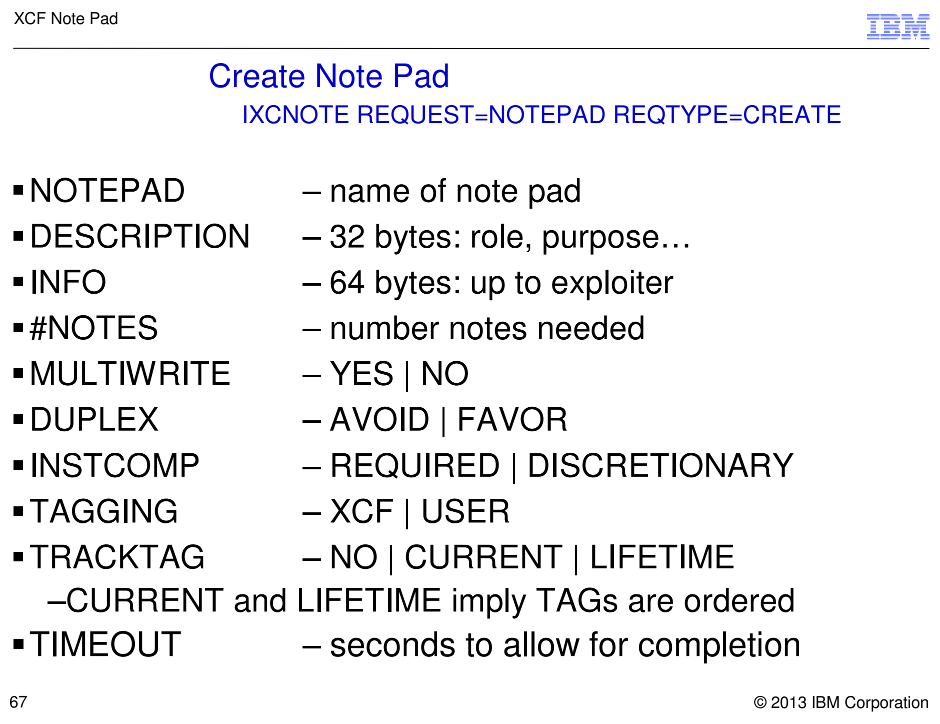
66

© 2013 IBM Corporation

Applications that issue IXCNOTE requests on systems that do not have the necessary support will get a failing return/reason code. Systems without this support will not understand an IXCNOTE request and reject it with whatever invalid condition is encountered. The IXCNOTE documents what values might be returned for such cases. There are only a few possibilities.

However, there is an official interface that is intended for this purpose. IXCQUERY REQINFO=FEATURES can be used to determine whether IXCNOTE is supported on a given system. In the output, the QuReqRfIxcNoteServiceAvail flag will be non-zero if XCF Note Pad Services are installed on the system. Mapping macro IXCYQUAA defines the “feature string” that contains this flag.

IXCYQUAA - QuReqRfIxcNoteServiceAvail



Create Note Pad

IXCNOTE REQUEST=NOTEPAD REQTYPE=CREATE

▪ NOTEPAD	– name of note pad
▪ DESCRIPTION	– 32 bytes: role, purpose…
▪ INFO	– 64 bytes: up to exploiter
▪ #NOTES	– number notes needed
▪ MULTIWRITE	– YES | NO
▪ DUPLEX	– AVOID | FAVOR
▪ INSTCOMP	– REQUIRED | DISCRETIONARY
▪ TAGGING	– XCF | USER
▪ TRACKTAG	– NO | CURRENT | LIFETIME

–CURRENT and LIFETIME imply TAGs are ordered

▪ TIMEOUT	– seconds to allow for completion

Before a note pad can be used, it must be created. A create note pad request is rejected if the note pad is already defined. This slide depicts the various keywords that can be specified when creating a note pad.

NOTEPAD – name of the note pad to be created. The name consists of four 8 byte sections referred to as "owner", "application", "function" and "qualifier". "owner" and "application" are required. The others can be blank. Although we describe a note pad name with a dot qualified format of owner.application.function.qualifier, the note pad name supplied via this keyword never includes the dots. If a section has fewer than 8 characters, it must be left justified within the section and appended on the right with EBCDIC blanks. For example: `'THIS    ISA     NOTEPADSNAME    '` where `owner='THIS    '`, `application='ISA     '`, `function='NOTEPADS'` and `qualifier='NAME    '`. The application must document the note pad name so the system programmer can configure the sysplex to support the note pad.

DESCRIPTION – A short description of the note pad, its function, or exploiting application. The idea is to provide information that allows the installation and service personnel to understand who is using the note pad and perhaps why.

INFO – arbitrary data up to you. The intended purpose is to allow you to associate your own control information with the note pad. This data is included with the output returned by a query note pad request. Your connections might want to obtain this data to determine what application specific protocols are to be used. This data is static for the life of the note pad.

#NOTES – the maximum number of notes that the note pad will need to support at any one time. XCF will put the note pad in a structure that has enough free space to support this number of notes. If your application tries to create more than the requested number of notes, the create note request will be rejected with a "note pad is full" reason. Your application should also allow for the possibility that a create note request is rejected with a "note pad constrained" reason, which occurs when the host note pad structure is out of notes but your note pad is still below its allotment.

MULTIWRITE – indicates whether XCF should permit more than one connection with update access to the note pad. If NO, at most one connection can create, write, replace, or delete notes from the note pad. Other connections can be created, but they are restricted to read access. If YES, more than one connection can have update access to the note pad.

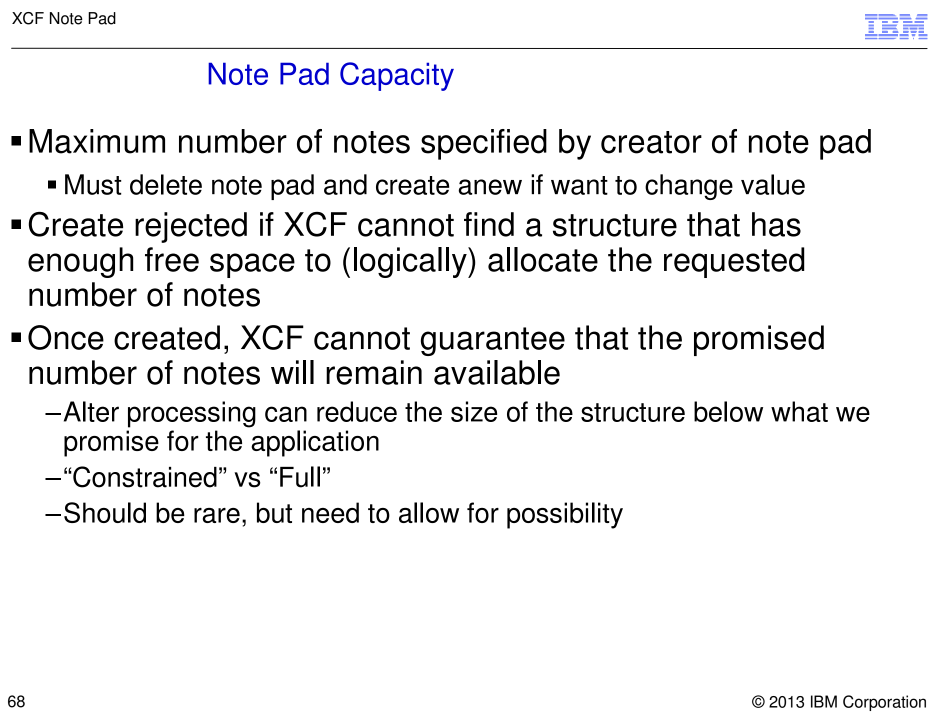
DUPLEX – indicates whether the application prefers that the note pad be hosted by a duplexed note pad structure or not. XCF can either try to avoid duplexed structures, or it can try to favor them. There is no guarantee as to whether or not XCF can accommodate the preference. Even if XCF can initially satisfy the preference, we are unable to guarantee that it will persist. There is no notification if the state changes. We do not provide any information as to whether the preference was satisfied. Authorized programs can discover this information at a point in time. I don't know of a way for unauthorized programs to do so.

INSTCOMP – indicates whether the users of the note pad are required to perform instance number comparisons when updating or deleting an existing note. If such comparisons are required, any replace note or delete note request that does not perform an instance number comparison will be rejected. A write note request will also be rejected if the subject note already exists. The INSTCOMP specification is not relevant to multi-note delete requests since they do not support instance number comparisons.

TAGGING – indicates who is responsible for setting the tag value for the notes in the note pad, XCF or the application. The tag value is 16 bytes of metadata associated with the note. Later when the application invokes IXCNOTE to process a single note request, the TAGGING specification on that invocation must match the choice made here by the note pad creator.

TRACKTAG – indicates whether XCF needs to retain a maximum tag value (MAXTAG) for the note pad: not at all, only for the notes that actually exist in the note pad at the time of the query, or for any note that ever existed for the life of the note pad. Issue a query the note pad request to get the maximum tag value. If MAXTAG is to be tracked, the user assigned tag values for a note must be non-decreasing for the life of a given note.

TIMEOUT – create note pad requests are potentially long running, though this will depend on the dynamics of the sysplex. You can indicate how long your program is willing to wait for the note pad to be created.



Note Pad Capacity

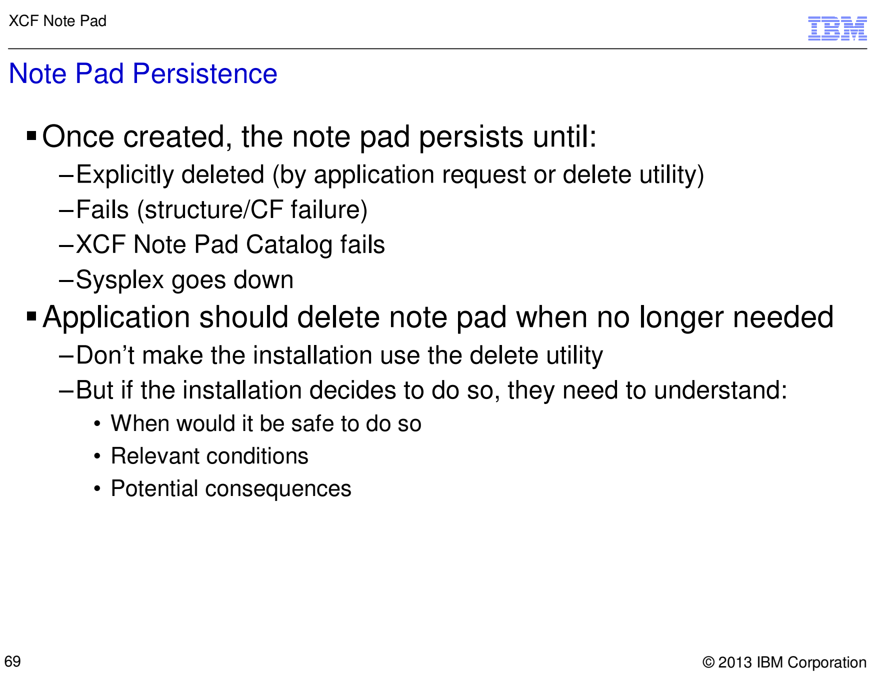
- Maximum number of notes specified by creator of note pad
  - Must delete note pad and create anew if want to change value
- Create rejected if XCF cannot find a structure that has enough free space to (logically) allocate the requested number of notes
- Once created, XCF cannot guarantee that the promised number of notes will remain available
  - Alter processing can reduce the size of the structure below what we promise for the application
  - "Constrained" vs "Full"
  - Should be rare, but need to allow for possibility

When a note pad is successfully created, XCF will have selected a host structure that has enough room for the application to be able to create the requested number of notes. If a note pad structure hosts more than one note pad, XCF makes sure that the structure has enough space to accommodate all of the notes requested by all the note pads in that structure. So if nothing changed, it would be possible for every note pad to create the requested maximum number of notes simultaneously.

However, structures can be altered by the system. In particular, alter processing can reduce the size of the structure. If alter processing reduces the size of the note pad structure below the sum of all the notes requested by all the note pads in the structure, XCF might not be able to create a new note in a given note pad – even though the note pad has not yet created its maximum number of notes. Thus there are two ways in which a create note request can be rejected with respect to space issues. XCF will indicate that the note pad itself is full if the requested number of notes already exist in the note pad. XCF will indicate that the note pad is "constrained" if the host structure is full (out of list entries) but the note pad itself is not currently using all the notes that it requested. That is, a note pad is constrained if a note cannot be created even though the note pad is not yet full.

From a practical standpoint, I think this is unlikely to be a real issue. For the initial exploitation, we expect that note pads will be sparsely populated. Even if usage were to become more dense, it will still likely be the case that not all note pads will have all their notes in use at any one time. So even if the size of the structure falls below the committed note capacity, the fact that not all note pads are consuming all their notes implies that there will likely still be capacity.

However, your application needs to be prepared to handle both sorts of "no note" conditions. You might treat both conditions alike, or you might choose to react differently to each.



XCF Note Pad

Note Pad Persistence

- Once created, the note pad persists until:
  - Explicitly deleted (by application request or delete utility)
  - Fails (structure/CF failure)
  - XCF Note Pad Catalog fails
  - Sysplex goes down
- Application should delete note pad when no longer needed
  - Don't make the installation use the delete utility
  - But if the installation decides to do so, they need to understand:
    - When would it be safe to do so
    - Relevant conditions
    - Potential consequences

69

© 2013 IBM Corporation

Once a note pad is created, it exists until it is deleted or fails.

A note pad can (and should) be explicitly deleted by the application when it is no longer needed. XCF provides a utility program that the installation can use to delete a note pad. But please don't make the installation use it. The utility should only be needed on an exception basis. They have enough real work to do without you making them clean up after your mess.

XCF will also delete a note pad that is deemed to have failed. A note pad is deemed to have failed if the host note pad structure "goes away", or the XCF note pad catalog "goes away", or the sysplex is re-IPLed. A structure "goes away" if it gets deleted, or if it fails, or if the coupling facility (CF) that contains the structure fails. There are also cases where the Coupling Facility Resource Manager (CFRM) will logically delete a structure and create a new physical instance of the structure. From an XCF perspective, any note pads that lived in the logically deleted structure instance are deemed to have failed. There are also scenarios where the sysplex might lose connectivity to a CF that contains a given structure and then gain connectivity to a different CF that also contains an instance of that same structure. XCF Note Pad Services detects that this structure instance does not match, and fails the affected note pads. In other words, it should not be possible for the note pad application to ever observe stale note data from an old instance of a note pad.

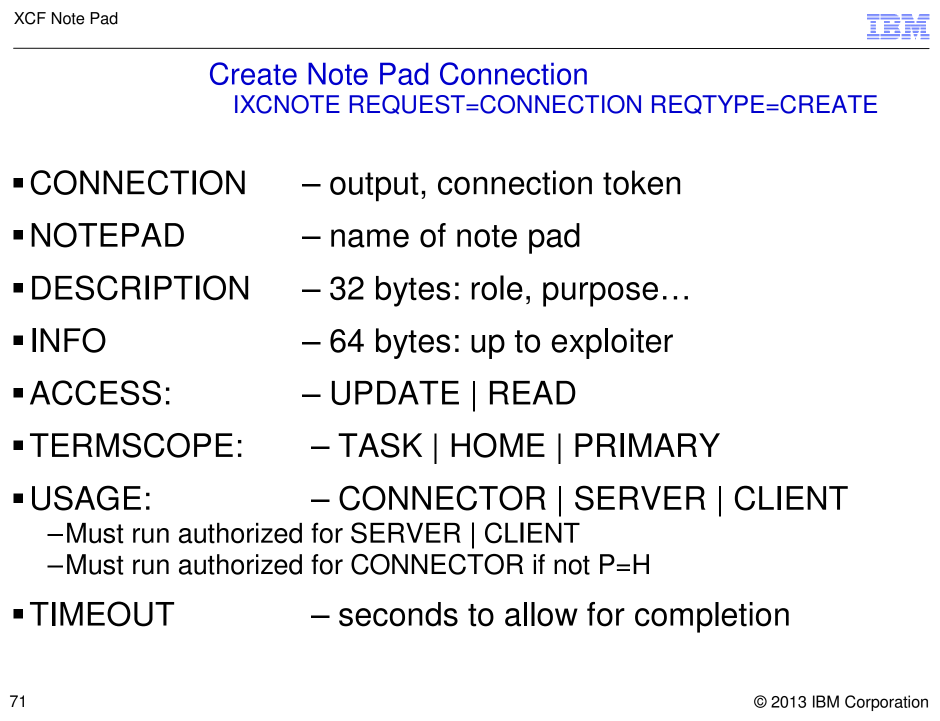
There may be times when the installation might decide to delete your note pad. Your product documentation should describe the potential ramifications of doing so. For example, what are the consequences of deleting the note pad if it contains notes or still has active connections. Ideally you would describe an appropriate procedure that would allow the installation to safely delete the note pad. Alternatively, perhaps in addition, you would also describe the recovery procedure to use to restore the application to a healthy state after the note pad has been deleted.

Query Note Pad
IXCNOTE REQTYPE=QUERY

▪NOTEPAD	- name of note pad
▪TIMEOUT	- seconds to allow for completion

Issue a query note pad request to get information about the note pad. If no answer area is provided via the ANSAREA keyword, the request is in essence testing for the existence of the note pad. If an answer area is provided, information about the note pad is returned. For the most part, the query echoes the parameters specified by the creator of the note pad. But you can also get information about which systems are connected to the note pad and the name of the host structure. However, the query does not return any information about the actual note pad connections.

The request is rejected if your program is not authorized for read access to the specified note pad.



XCF Note Pad

Create Note Pad Connection

IXCNOTE REQUEST=CONNECTION REQTYPE=CREATE

▪ CONNECTION	– output, connection token
▪ NOTEPAD	– name of note pad
▪ DESCRIPTION	– 32 bytes: role, purpose…
▪ INFO	– 64 bytes: up to exploiter
▪ ACCESS:	– UPDATE | READ
▪ TERMSCOPE:	– TASK | HOME | PRIMARY
▪ USAGE:	– CONNECTOR | SERVER | CLIENT

- –Must run authorized for SERVER | CLIENT
- –Must run authorized for CONNECTOR if not P=H

▪ TIMEOUT	– seconds to allow for completion

71 © 2013 IBM Corporation

To use a Note Pad, one must “connect” to the Note Pad. Generally, each address space that wants to use the note pad will need to establish a connection. There is a limit on the maximum number of connections that can be created from a given address space (currently 128). The limit applies across all note pads. So for example, within any given address space, programs could create 128 connections to one note pad, or one connection to each of 128 note pads, or any combination in between. After a connection is created, the connector can manipulate notes in the Note Pad.

Do not confuse a connection to a note pad with a XES connection to a CF structure. Under the covers, XCF does the necessary IXLCONN to connect to the CF structure that contains the note pad. There will be at most one such XES connection per system. The note pad exploiters are unaware of the XES connection.

This slide depicts the various keywords that can be specified when creating a connection to a note pad.

CONNECTION is an output variable into which XCF will store a connection token to represent the connection. This token must be passed as input on subsequent IXCNOTE requests that are issued to manipulate notes in the note pad and/or to manipulate the connection itself. You will need to understand when the connection token is valid for use.

NOTEPAD is the name of the note pad to which a connection is to be established. The note pad must already exist.

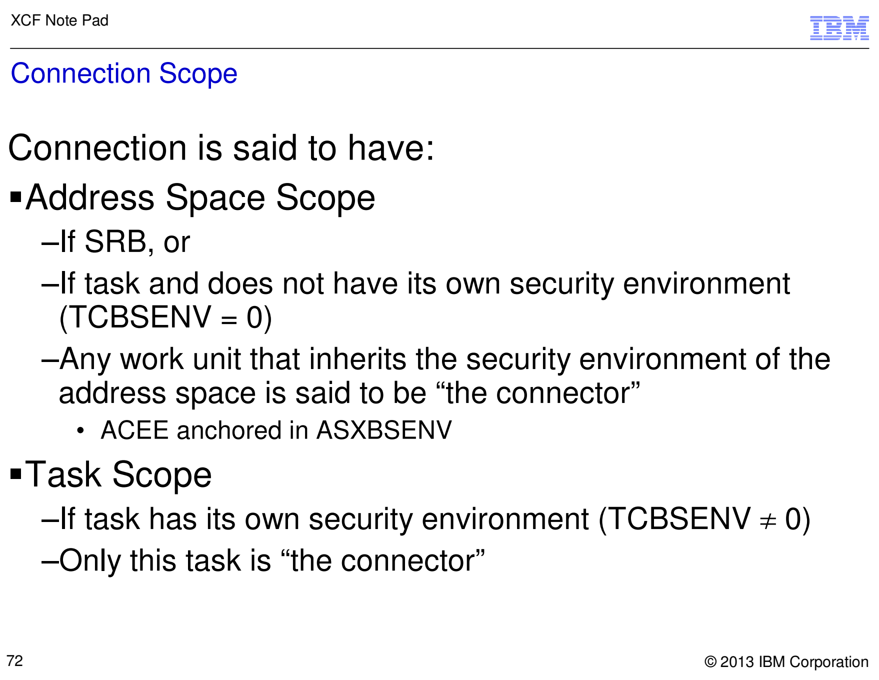
DESCRIPTION is a short description of the connection, or the application or function that is using the connection. The idea is to provide some human readable information that will allow the installation and service personnel to understand who is using the connection and perhaps why.

INFO is arbitrary data up to you. The intended purpose is to allow you to associate your own control information with the connection. This data is static for the life of the note pad. Unfortunately, XCF does not currently provide an interface through which this information can be obtained. If we did have such an interface, your peer connections might want to obtain this data to determine what application specific protocols can be used by the various connections.

ACCESS indicates what type of note pad access is to be granted to the connection. A connection with update access can create, write, replace, read, or delete notes. A connection with read access is only allowed to read notes. For update access, the connector must have SAF authority that permits UPDATE access. For read access, the connector must have SAF authority that permits READ access.

TERMSCOPE identifies the entity to which the connection is to be bound for termination purposes. The connection can be bound to a specific task or to an address space. When the indicated entity terminates, XCF deletes the connection.

USAGE indicates the context in which the connection is to be used. There are three models: connector, server, or client. Connector is likely the most typical, and the only type of connection that can be created by an unauthorized program. With **CONNECTOR**, the connection is created by and for use of the connector. With **SERVER**, the connection is created by some server, and can be used by the server while running in the server address space. The program must be running authorized to create and use a SERVER connection.



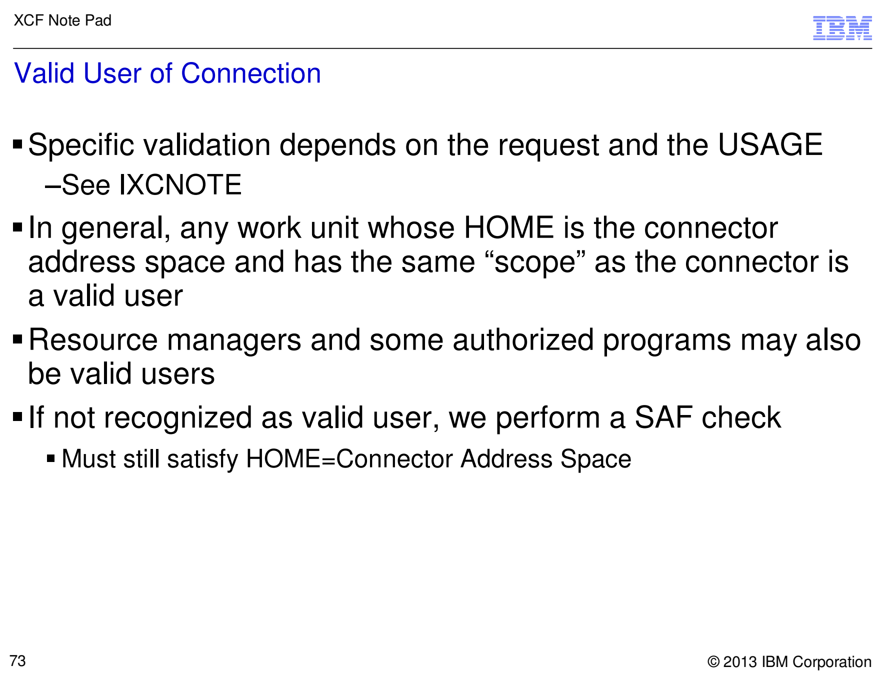
Connection Scope

Connection is said to have:

- Address Space Scope
  - –If SRB, or
  - –If task and does not have its own security environment (TCBSENV = 0)
  - –Any work unit that inherits the security environment of the address space is said to be "the connector"
    - ACEE anchored in ASXBSENV
- Task Scope
  - –If task has its own security environment (TCBSENV ≠ 0)
  - –Only this task is "the connector"

Note that all SAF checking is performed against the security environment of the work unit. For a task that has a task specific security environment, the Access Control Environment Element (ACEE) anchored at offset TCBSENV in the TCB (macro IKJTCB) is used. For an SRB, or a task for which TCBSENV is zero, the ACEE anchored at offset ASXBSENV in the ASXB (macro IHAASXB) for the home address space is used.

As a general principle, if a given work unit has the same scope as the connection, we bypass the SAF check as we would expect to get the same result as for the connection. However, this does imply that we will not observe changes to the connector's authorization. The connection would need to be deleted and created anew for XCF to see such changes (by issuing a new SAF check). This fact might be of interest to the installation in cases where the authorization needs to be changed dynamically (perhaps they need to correct a mistake) after a connection is created. The installation would need to have a mechanism by which to get your application to delete and then re-create its connection. Not clear that you need to provide such support. But it might be reasonable in general for you to provide product documentation describing how the installation should go about shutting down your use of a note pad.



XCF Note Pad

Valid User of Connection

- Specific validation depends on the request and the USAGE
  - –See IXCNOTE
- In general, any work unit whose HOME is the connector address space and has the same "scope" as the connector is a valid user
- Resource managers and some authorized programs may also be valid users
- If not recognized as valid user, we perform a SAF check
  - Must still satisfy HOME=Connector Address Space

73 © 2013 IBM Corporation

A **valid user** is a work unit that is allowed to use a connection to a note pad. Each IXCNOTE request that requires a connection token as input indicates the conditions under which the requester is deemed to be a valid user. The creator of the connection specifies the USAGE keyword to indicate the criteria that are to be used to determine a valid user when accessing notes in the note pad (REQUEST=NOTE or REQUEST=NOTES).

For example, when processing notes, a work unit is deemed to be a valid user of the connection if it can satisfy any of the following conditions:

•Requester is the connector (home=connector, work unit has same "connection scope" as the connector)

•Home is connector space and user has SAF authorization appropriate for the REQTYPE

•Supervisor state or PKM allowing key 0 to 7, and running as an address space resource manager

For USAGE=SERVER we add the following condition:

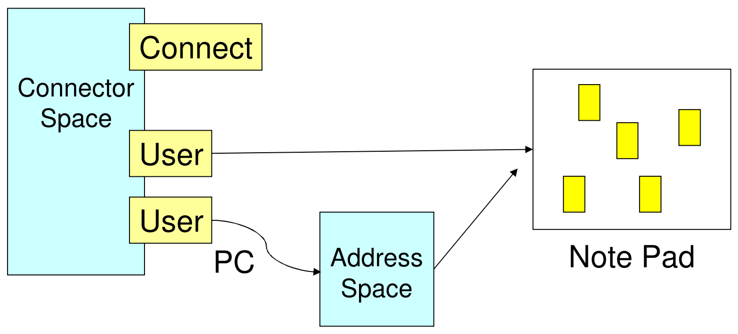
•Primary is connector space, and requester is supervisor state or PKM allowing key 0 to 7.

For USAGE=CLIENT we add the following condition:

•Requester is supervisor state or PKM allowing key 0 to 7.

For USAGE=SERVER and USAGE=CLIENT, the authorized program is responsible for ensuring that access is permitted. In general it is because it is the server making the access (and it passed SAF check when the connection was created). But if the server were to permit the client to access the notes, the server would then be responsible for ensuring that the client had the necessary SAF authorization for such access.

Usage=Connector



Connection created if SAF permits
Any work unit with home=connector can use connection

One of the models for using a connection.

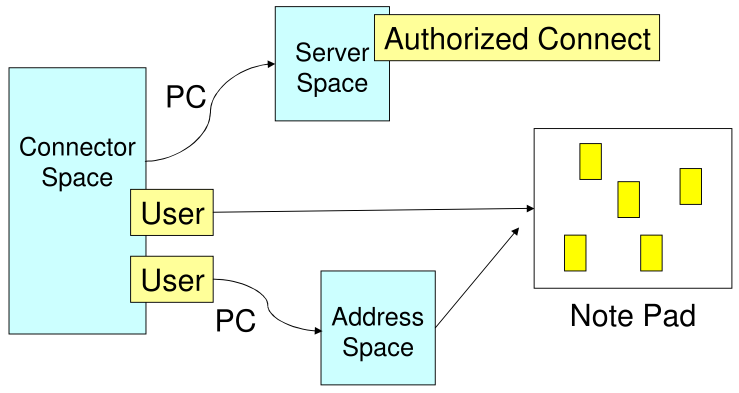
Usage=connector: A work unit running in the connector address space creates the connection. Work units that originate in the connector address space can use the connection. This is the normal means of creating and accessing Note Pads by unauthorized programs.

The connection is terminated when the connector address terminates, or when a designated task in the connector space terminates.

When processing notes, a work unit is deemed to be a valid user of the connection if it can satisfy any of the following conditions:

- Requester is the connector
- Home is connector space and user has SAF authorization appropriate for the REQTYPE
- Supervisor state or PKM allowing key 0 to 7, and running as an address space resource manager

Usage=Connector (authorized creator)



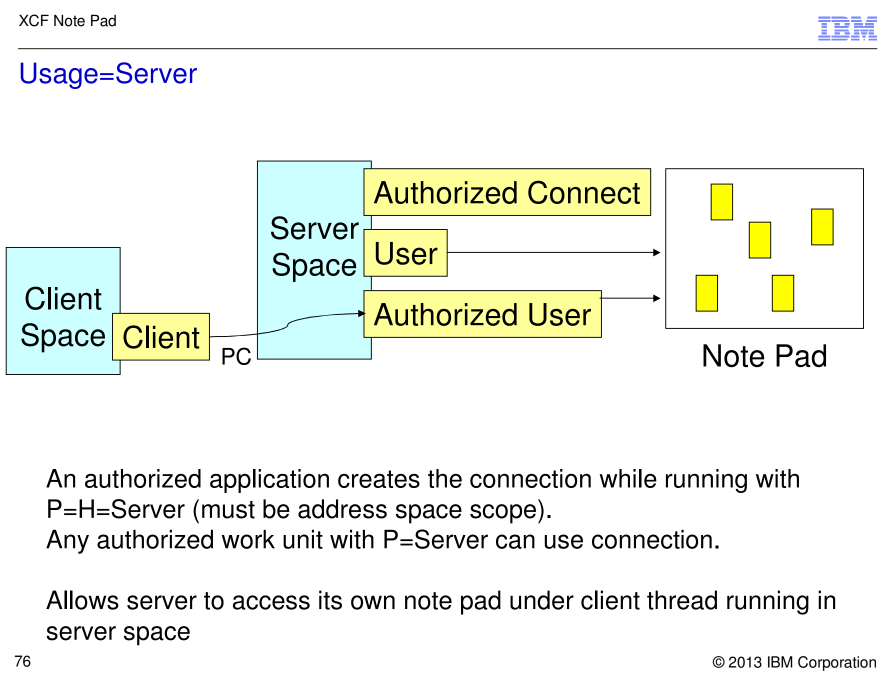
Connection created if SAF permits (connector work unit)
Any work unit with home=connector can use connection
A server can create a connection on behalf of a client, for use by clients

One of the models for using a connection.

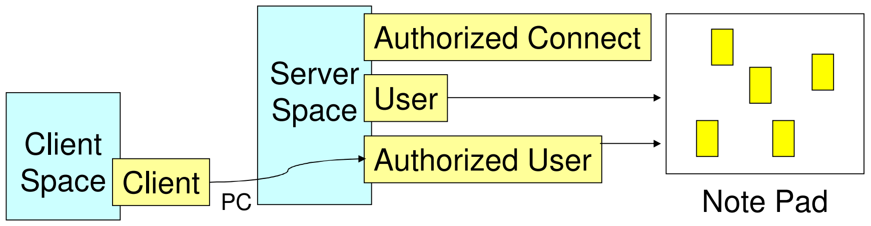
Usage=connector (XMEM creator): Connection can be created while running in cross memory environment, provided the requester is running authorized. The connection is associated with the home space and can be used by any work unit that originates in the home space (whether authorized or not). Thus a server can create a connection on behalf of, and for use by, its clients.

The connection is terminated when the connector (home) address terminates. Optionally, the server that creates the connection can indicate that the connection is to also be terminated when the server address space terminates, or when a designated task in the server address terminates.

Note that a work unit that originates in the server address space would not be permitted to use the connection. It was the client work unit originating from the connector address space that had SAF authority to use the note pad. Work units originating in the server address space are not recognized as being the connector.



Usage=Server



An authorized application creates the connection while running with P=H=Server (must be address space scope).
Any authorized work unit with P=Server can use connection.

Allows server to access its own note pad under client thread running in server space

One of the models for using a connection.

Usage=Server: A server creates a connection and is also the user of the connection. The server provides services to clients, and in so doing, needs to access Note Pads for its own internal purposes. Clients PC to the server space and the access to the Note Pad is made by the server while running under the client thread. The client does not need SAF authorization to the Note Pad. However the server needs to be running authorized when accessing the Note Pad.

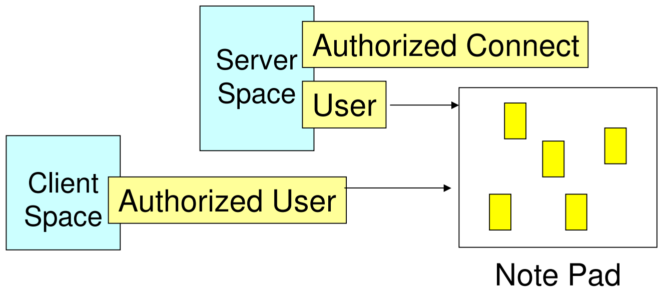
The connection is terminated when the server address space terminates, or when a designated task in the server address space terminates.

In the server model, the connection is created so that the server can use a notepad in conjunction with providing services to its clients. The client invokes a service, and the server accesses the notes in the notepad. It is the server's note pad and it is the server that is authorized to access the note pad. In the SERVER model, the server accesses its own notepad while running under a client work unit, but the primary address space at the time of the access must be the server address space. Note that the client is unlikely to have SAF authority for the server note pad. And in fact, XCF does not check to see whether the client work unit has SAF authorization. Thus both the creator and user of the connection must be running authorized. If the server chooses to give the client access to the note pad, it would be the responsibility of the server to perform the necessary SAF checking.

Note that the connection can also be used by the server itself in any context where USAGE=CONNECTOR type accesses would be permitted. That is, had the connection had been created with USAGE=CONNECTOR, work units originating from the server address space would have been permitted to use the connection. Any work unit that would have qualified for such usage qualifies as a valid user of the USAGE=SERVER connection. The work unit does not need to run authorized.

Note that your program could provide equivalent functionality to your clients with a USAGE=CONNECTOR connection as well. However, to do so, when the client PC's into your server address space, you would then need to queue the work off to some other work unit in the address space. With that implementation, you would not need to run authorized. With a USAGE=SERVER connection, you do not need to make the context switch since you are allowed to access the server note pad while running under the client thread. However, you must be running authorized to do so.

Usage=Client



An authorized application creates the connection
Any authorized application can use the connection

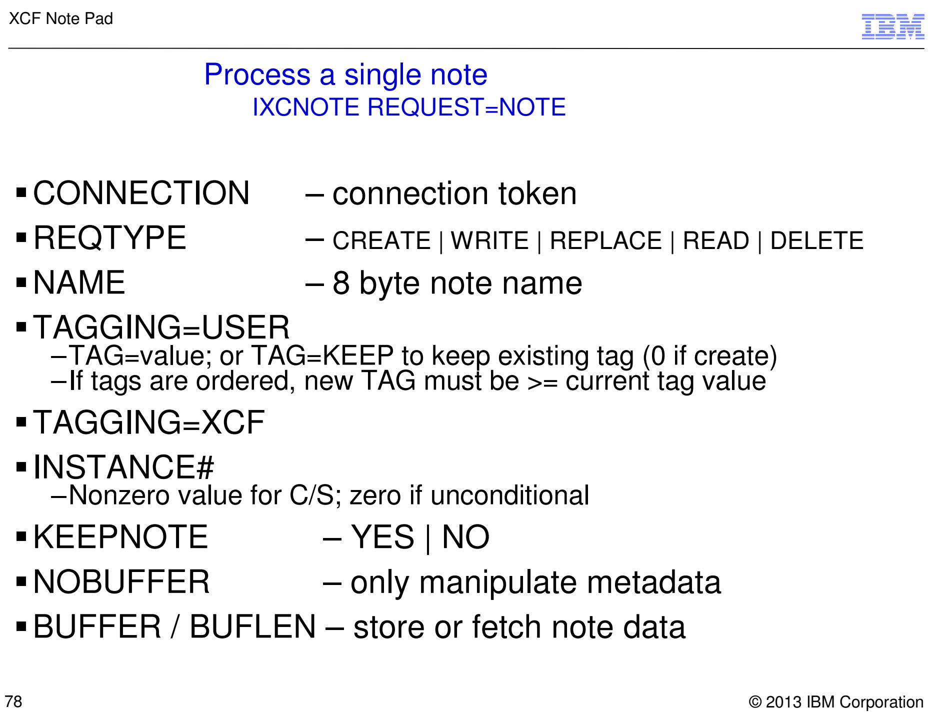
Allows server to access note pad while running in client space

One of the models for using a connection.
Usage=Client: A server creates a connection out of its own space, but wants to be able to process notes while running in client address spaces. The connection can be used from any address space by any work unit that runs authorized. The creator of the connection must also be running authorized.
The connection is terminated when the server address terminates, or when a designated task in the server space terminates.

In the client model, the connection is created so that the server can use a notepad in conjunction with providing services to its clients. The client invokes a service, and the server accesses the notes in the note pad. It is the server's note pad and it is the server that is authorized to access the note pad. In the CLIENT model, the server accesses its own notepad while running under a client work unit. Note that the client is unlikely to have SAF authority for the server notepad. And in fact, XCF does not check to see whether the client work unit has SAF authorization. Thus both the creator and user of the connection must be running authorized. If the server chooses to give the client access to the note pad, it would be the responsibility of the server to perform the necessary SAF checking.

Note that the connection can also be used by the server itself in any context where USAGE=CONNECTOR type accesses would be permitted. That is, had the connection had been created with USAGE=CONNECTOR, work units originating from the server address space would have been permitted to use the connection. Any work unit that would have qualified for such usage qualifies as a valid user of the USAGE=CLIENT connection. The work unit need not be running authorized.

Note that your program could provide equivalent functionality to your clients with a USAGE=CONNECTOR connection as well. However, to do so, the client would PC into your server address space, you would then need to queue the work off to some other work unit in the server address space. With that implementation, you would not need to run authorized. With a USAGE=CLIENT connection, you do not need to make the context switch since you are allowed to access the server note pad while running under the client thread in the client address space (or where ever the work unit takes you). However, you must be running authorized to do so.



Process a single note
IXCNOTE REQUEST=NOTE

- CONNECTION – connection token
- REQTYPE – CREATE | WRITE | REPLACE | READ | DELETE
- NAME – 8 byte note name
- TAGGING=USER
  - –TAG=value; or TAG=KEEP to keep existing tag (0 if create)
  - –If tags are ordered, new TAG must be >= current tag value
- TAGGING=XCF
- INSTANCE#
  - –Nonzero value for C/S; zero if unconditional
- KEEPNOTE – YES | NO
- NOBUFFER – only manipulate metadata
- BUFFER / BUFLEN – store or fetch note data

After a connection is created, the application can manipulate notes in the note pad. Every note is associated with some one connection. Whichever connection most recently updated the note is deemed to be the **associated connection**. Initially the connector that creates the note is the associated connection. After that, any connection that updates (write or replace) the note is deemed to be the associated connection. Deleting a note does not change the associated connection (the note is gone).

This slide depicts the various keywords that can be specified when manipulating one particular note in a note pad.

CONNECTION is the connection token returned by a successful create note pad connection request.

REQTYPE indicates what operation is to be applied to the note. You can create, write, replace, read, or delete the note. A create request creates a new note. The request is rejected if the note already exists. A write request will create the note if it does not currently exist, and replace the note if it does exist. A replace request updates an existing note. The data content and tag value can be changed. The associated connection is implicitly changed by XCF as the result of a successful note update.

NAME is the name of the note to be manipulated. Note names within a note pad must be unique. At most one note with a given name can exist at any one time.

The **TAGGING** specification must match the TAGGING specification used by the creator of the note pad. For **TAGGING=USER**, the application is responsible for setting the tag value of the note. The tag value is 16 bytes of metadata associated with the note. Use the **TAG** keyword to specify the new tag value. The tag value of an existing note is not changed if TAG=KEEP is specified. Otherwise the tag value of the note is set to the indicated value when creating, updating, or deleting a note. Yes, a new tag value can be set when deleting a note (but this is only meaningful if the creator of the note pad specified TRACKTAG=LIFETIME). If the creator of the note pad specified TRACKTAG=CURRENT or TRACKTAG=LIFETIME, the new tag value must be greater than or equal to the current tag value of the note (if the note exists). For **TAGGING=XCF**, a new tag value is set to an XCF determined value when creating or updating a note. The tag value assigned by XCF is an ever increasing sequence number. The sequence number is global to the note pad. Every time a note is created or updated, the tag value is set to the current value of the sequence number and then the sequence number is incremented. Thus the tag values for any given note are ever increasing, but they may not be successive.

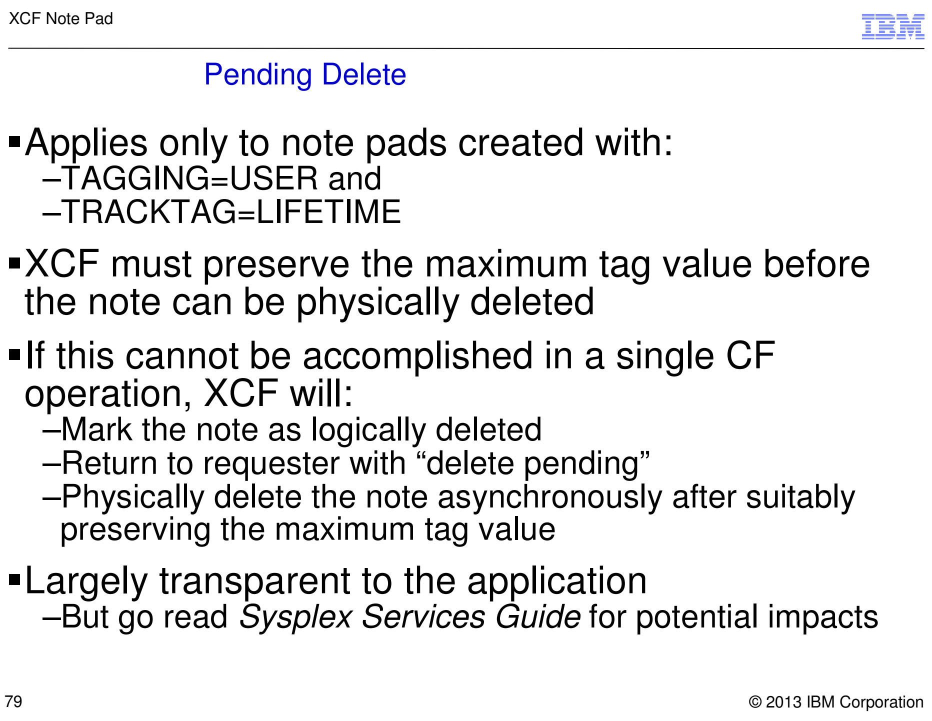
The **INSTANCE#** specification determines whether XCF will perform instance number comparisons when reading, updating, or deleting a note. Every note has an instance number. XCF sets the instance number when the note is first created. The instance number is changed whenever the note is updated (note that I said "changed" not "incremented" – don't make any assumptions about the relationship between two "successive" instance numbers for a given note). If INSTANCE# is zero (or not specified), XCF will not perform instance number comparisons. If INSTANCE# is non-zero, XCF will perform an instance number comparison. If the creator of the note pad specified INSTCOMP=REQUIRED, you must ask XCF to perform an instance number comparison. When performing an instance number comparison, the request will be rejected if the specified instance number does not equal the current instance number value for the note.

KEEPNOTE indicates whether the note is to be kept in the note pad after the associated connection is deleted. This specification applies when a note is being created or updated. Recall that the connection that most recently creates or updates a given note becomes associated with that note. When a connection is deleted, XCF finds the notes in the note pad that are associated with that connection and deletes any for which KEEPNOTE=NO was specified. Otherwise the note will be kept in the note pad even thought the associated connection no longer exists. The KEEPNOTE specification provides a way for you to have XCF do cleanup for your connection. Note that the KEEPNOTE specification is updated every time the note is updated. So if you want the note to survive the connection, you must specify KEEPNOTE=YES every time.

Use **NOBUFFER** to create a null note, to update a note without changing its content, or to read the metadata without reading the note content.

Use **BUFFER** and **BUFLEN** to identify a storage area for the note content (data). When creating or updating a note, the buffer contains the data to be written in the note. If BUFLEN is zero, a null note is written. When reading or deleting a note, XCF stores the note data in the buffer. If BUFLEN is zero, the note must be a null note.

.



Pending Delete

- Applies only to note pads created with:
  - –TAGGING=USER and
  - –TRACKTAG=LIFETIME
- XCF must preserve the maximum tag value before the note can be physically deleted
- If this cannot be accomplished in a single CF operation, XCF will:
  - –Mark the note as logically deleted
  - –Return to requester with “delete pending”
  - –Physically delete the note asynchronously after suitably preserving the maximum tag value
- Largely transparent to the application
  - –But go read *Sysplex Services Guide* for potential impacts

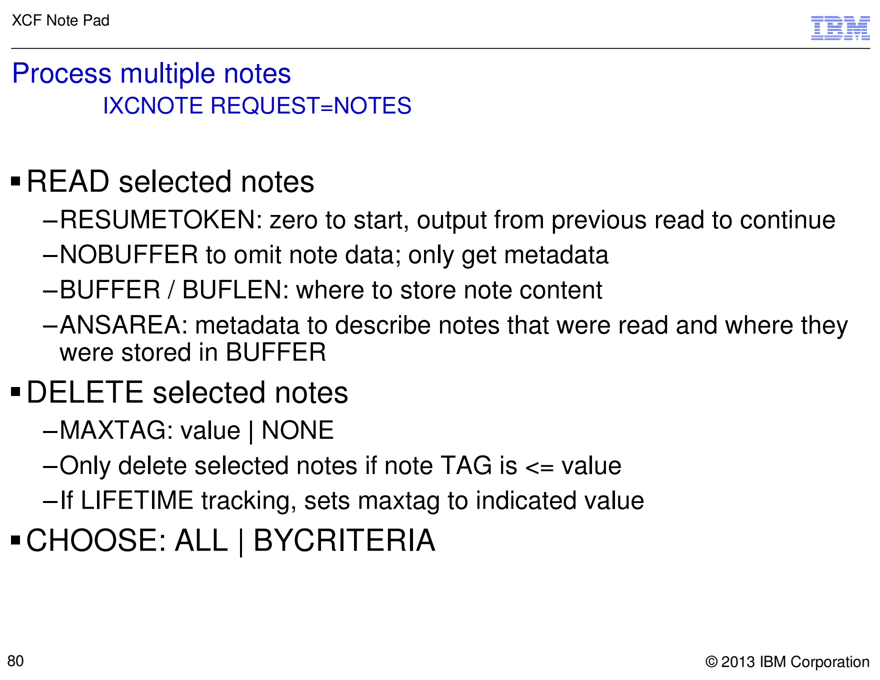
If the creator of the note pad specified TAGGING=USER and TRACKTAG=LIFETIME, XCF is required to preserve the maximum tag value ever assigned to any note that ever existed in the note pad. So when a note is deleted, XCF may need to preserve its tag value so as to retain the maximum tag value. In general, XCF tries to execute at most one coupling facility (CF) operation per (single note) request. When deleting a note that might have the maximum tag value, XCF may need to perform multiple several CF operations in order to preserve the maximum tag value. On those occasions, XCF will issue a CF operation to mark the note as “logically deleted” and return control to the application. Asynchronously to the delete request, XCF will arrange for the maximum tag value to be preserved and the logically deleted note to be physically deleted from the note pad. We may say that the subject note is **pending delete** or that XCF is doing a **deferred delete**.

For the most part, the fact that a note is pending delete will be largely transparent to the application. It is quite likely that XCF will be able to preserve the maximum tag value and physically delete the note before the application can notice. However, until the note is in fact physically deleted, we have the following potential impacts:

(1) When querying a note pad, the number of notes in use will include the logically deleted note.

(2) When creating a new note, the note pad may appear to be full. The create note request that encounters a full condition will incur additional overhead (more than one CF operation) because XCF will finish physically deleting the logically deleted note to make room for the new note.

(3) A subsequent request to manipulate the note may incur additional overhead. XCF ensures that the request proceeds as if the note had been physically deleted. For example, a request to replace the (logically deleted) note will be rejected as if the note did not exist (even though it physically did exist when the replace request was processed by the CF). When a subsequent request encounters a logically deleted note, XCF will complete the deferred delete at that time. Our gamble that the deferred delete would be done before anyone would notice did not pan out. So the next request incurs the additional overhead.



Process multiple notes

IXCNOTE REQUEST=NOTES

- READ selected notes
  - –RESUMETOKEN: zero to start, output from previous read to continue
  - –NOBUFFER to omit note data; only get metadata
  - –BUFFER / BUFLEN: where to store note content
  - –ANSAREA: metadata to describe notes that were read and where they were stored in BUFFER
- DELETE selected notes
  - –MAXTAG: value | NONE
  - –Only delete selected notes if note TAG is <= value
  - –If LIFETIME tracking, sets maxtag to indicated value
- CHOOSE: ALL | BYCRITERIA

This slide depicts the various keywords that can be specified when manipulating a set of notes in a note pad. While processing a multi-note request, XCF might issue one or more coupling facility (CF) operations.

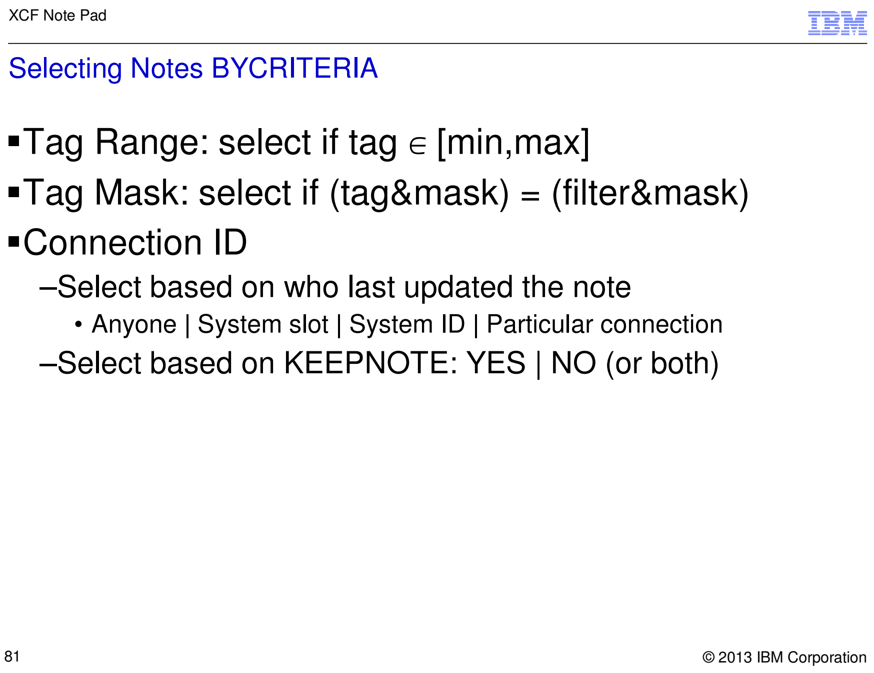
CONNECTION is the connection token returned by a successful create note pad connection request.

REQTYPE indicates what operation is to be applied to the set of notes. You can either read them or delete them.

Specify **REQTYPE=READ** to read a collection of notes. You must provide an **answer area** where XCF is to store information about the notes that were read. For each note that gets read, XCF stores its metadata (note name, tag value, and instance number) in the answer area. If you also provide a **buffer area**, XCF will store the note content (data) in the buffer and include information in the answer to indicate where the note content was stored. If you specify **NOBUFFER**, the note content will not be read. XCF will continue issuing CF operations to read the notes until all the selected notes are read, or until the designated answer area is full, or the designated buffer area is full, whichever comes first. Upon return, XCF will store a **resume token** to indicate where the read request left off. Pass this resume token to a subsequent read notes request to have XCF continue reading the remaining notes that would not fit in the specified storage areas. If all the notes were in fact read, you can still use the resume token for a subsequent read notes request. The read request will return any notes that got created after the previous read notes request had finished looking at all the notes that existed at that time.

Specify **REQTYPE=DELETE** to delete a collection of notes. As needed, XCF will continue issuing CF operations to delete the notes until all the selected notes are deleted. XCF will not return until all the requested notes have been deleted (or a failure occurs). If a tag value is provided via the MAXTAG keyword, only notes whose tag value is less than or equal to the specified tag value will be deleted. Thus MAXTAG might be used in place of other selection criteria, or it might be used in conjunction with other selection criteria to further restrict the set of notes to be deleted. If the creator of the note pad specified TRACKTAG=LIFETIME, specifying MAXTAG=value has an additional side effect. After all the notes are deleted, XCF will set the maximum tag value for the note pad to the specified value if that value is greater than the current maximum tag value.

The **CHOOSE** keyword indicates how the notes are to be selected. You can either choose **ALL** the notes in the note pad, or you can choose **BYCRITERIA** to specify various conditions that a note must satisfy to be selected. Selection criteria are discussed on the next slide.



XCF Note Pad

Selecting Notes BYCRITERIA

- Tag Range: select if tag ∈ [min,max]
- Tag Mask: select if (tag&mask) = (filter&mask)
- Connection ID
  - Select based on who last updated the note
    - Anyone | System slot | System ID | Particular connection
  - Select based on KEEPNOTE: YES | NO (or both)

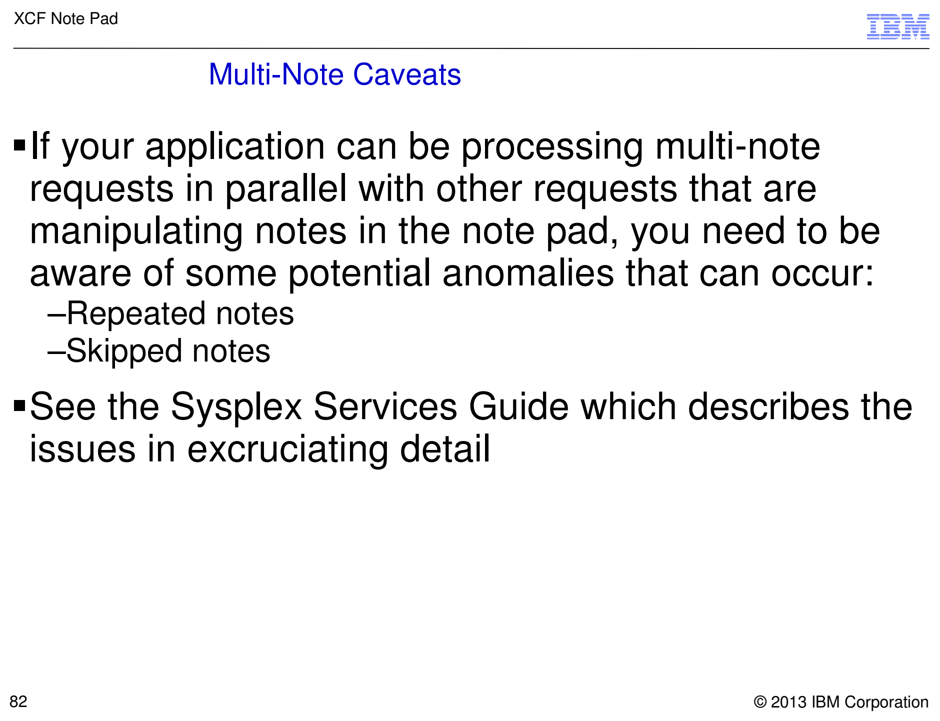
81 © 2013 IBM Corporation

When choosing notes “by criteria”, you code the CRITERIA keyword to identify a storage area containing the criteria that XCF should use to determine whether a note is to be selected for the multi-note request. The IXCYNOTE macro defines the mappings to be used for the data in this storage area. In essence you indicate what type of test is to be performed and the parameters to be used for that test. The following selection criteria are supported:

Tag Range: select the note if its current tag value is between the indicated min and max tag value, inclusive.

Tag Mask: Allows you to select notes for which certain fields within the tag have certain values. For example, your tag value might contain a field indicating the type of data in the note. You could use the tag mask criteria to choose all the notes that were of type 27 (for example). You specify a mask value to indicate which bits in the tag are to be inspected. You specify a filter value to indicate what values the inspected bits must contain. Taking our example, the mask value would tell XCF where to find the type field and the filter value would indicate what value the type field must contain. To be precise: if a particular mask bit is on, the corresponding bit in the current tag value of the note is compared to the corresponding bit in the filter tag value. The note is selected if every such compared bit has the same value in both the note tag and the filter tag.

Connection ID: select notes associated either with a given note pad connection or with a given system. A note is initially associated with the connection that creates the note. After that, the note is associated with the connection that most recently updated the note. A note is associated with a given system if its associated connection was created on that system. Every note pad connector has a connection ID (not to be confused with the connection token). These criteria can be used to select all notes in the note pad, notes associated with any instance of a given system, note associated with a given instance of a system, or notes associated with a given connection. You can choose to include notes for which KEEPNOTE=YES was specified, or for which KEEPNOTE=NO was specified, or both.



Multi-Note Caveats

- If your application can be processing multi-note requests in parallel with other requests that are manipulating notes in the note pad, you need to be aware of some potential anomalies that can occur:
  - –Repeated notes
  - –Skipped notes
- See the Sysplex Services Guide which describes the issues in excruciating detail

When processing a multi-note request, the XCF Note Pad Services inspect the notes in the note pad to determine whether they meet the prescribed selection criteria. If your program creates, updates, or deletes notes while a multi-note request is being processed, you might need to account for some anomalies that could occur with regard to note selection. For example, the multi-note request might fail to include all the notes that meet the selection criteria, or it might include a given note more than once. If your program needs to reissue a multi-note request one or more times to process all the notes in the note pad, these anomalies can occur if your program creates, updates, or deletes notes in between the reissued multi-note requests. To understand how these anomalies might arise, you need to understand how the XCF Note Pad Services scan the note pad when selecting notes.

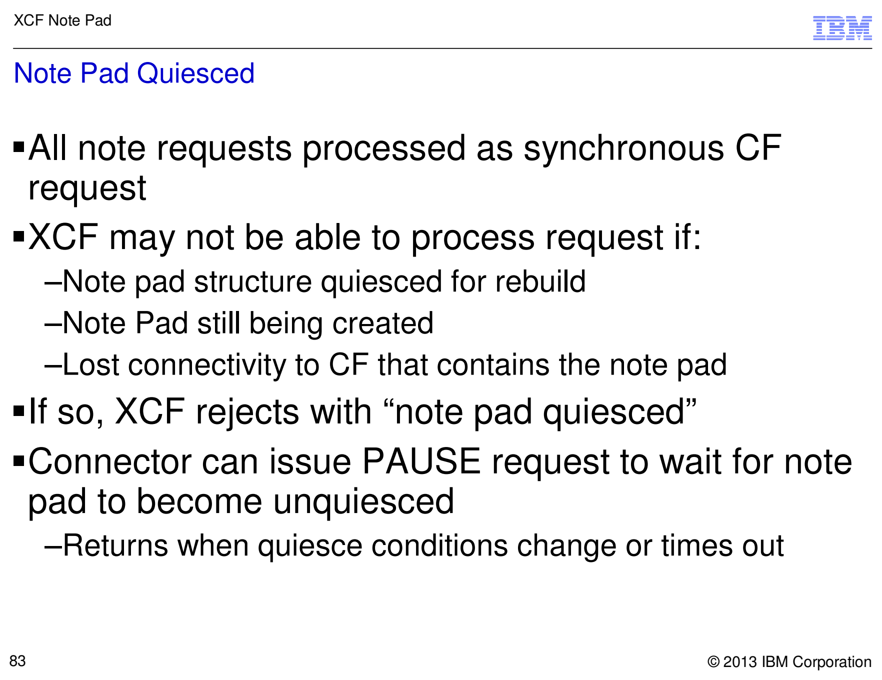
When processing a multi-note request, the notes in the note pad are scanned in the order that they were created. More precisely, the notes in the note pad are maintained in a sorted sequence based on a timestamp taken by the XCF Note Pad Services just before the operation that causes the note to be created is sent to the coupling facility. Thus there is a window between the setting of the created timestamp and the physical creation of the note in the note pad. This window could be observed by a multi-note request. This timestamp remains constant for the life of the note. In particular, the timestamp is not changed when the content of an existing note is replaced. However, if a note is deleted and created anew, the new instance of the note is assigned a new timestamp.

Thus you might need to account for scenarios like the following:

•A given note might be processed more than once if notes are being deleted and recreated while the multi-note request is being processed. Each time a note is newly created, a new created timestamp is set. So as the multi-note request scans the notes in created timestamp order, the timing could be such that it will encounter each newly created instance of the note.

•If an existing note is replaced one or more times while a read notes request is being processed, at most one of the instances will be returned by the request.

•The multi-note request can only observe notes that actually exist in the note pad. But the created timestamps are set by the z/OS system before the note is physically created in the note pad. Thus there could be a delay between when the timestamp is set and when the note is created in the note pad. Thus it is possible for notes to be physically created in a different order than one might expect based on the created timestamps. When this occurs, a multi-note request might not include all the expected notes.



XCF Note Pad

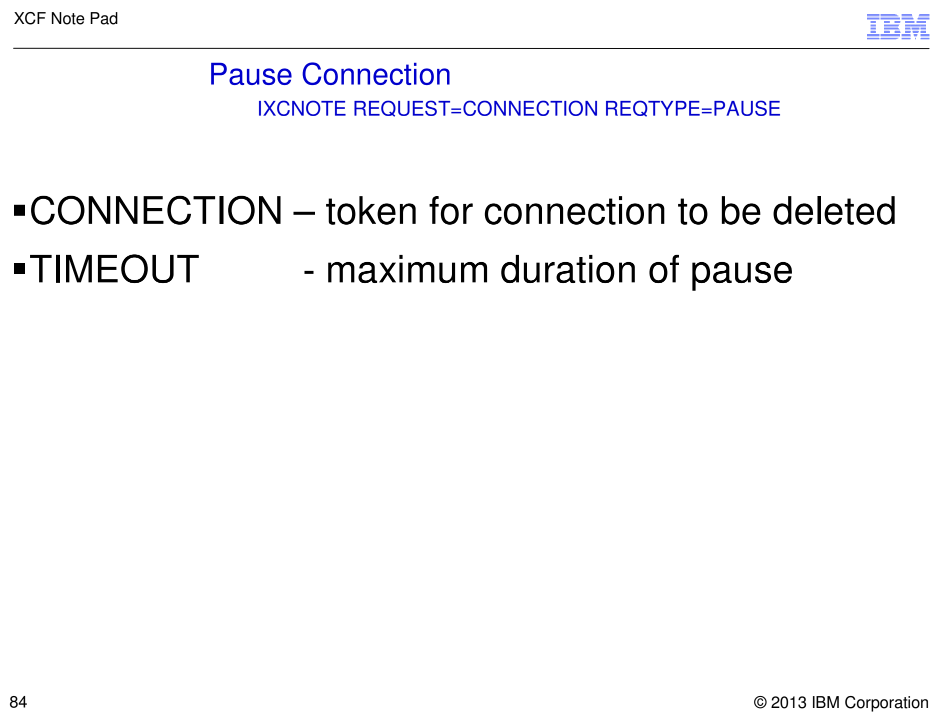
Note Pad Quiesced

- All note requests processed as synchronous CF request
- XCF may not be able to process request if:
  - Note pad structure quiesced for rebuild
  - Note Pad still being created
  - Lost connectivity to CF that contains the note pad
- If so, XCF rejects with “note pad quiesced”
- Connector can issue PAUSE request to wait for note pad to become unquiesced
  - Returns when quiesce conditions change or times out

83 © 2013 IBM Corporation

In general, all note pad requests are processed as synchronous coupling facility (CF) operations. However, a request could encounter conditions that might lead to significant delays in processing the request. Since the note request might be issued on a performance sensitive flow, XCF rejects the request so that the work unit would not remain suspended for an extended period of time. In these cases, the note pad is said to be quiesced. When a note pad is quiesced, XCF will not accept any requests to process notes in the note pad. A note pad will be quiesced, for example, when the host structure is undergoing structure rebuild or when the local system loses connectivity to the coupling facility (CF) that contains the note pad. Creating a note pad is a multi-step process. As soon as XCF creates the note pad definition (so the note pad is logically defined), the application can start to create connections. However, these connection cannot process any notes until XCF finishes instantiating the note pad in a host note pad structure (so the note pad is physically defined). The note pad will also be quiesced in the window where it is logically defined but not yet physically defined.

If requests are being rejected because the note pad is quiesced, the question naturally arises: “How will I know when the note pad becomes unquiesced to that I can resume normal activity?” One possibility is to continue issuing note requests. They will either eventually work or eventually fail for some reason other than a quiescing condition. However, continually issuing note requests to detect the end of the quiescing condition is probably not the best use of system resources. Instead, you can issue an IXCNOTE PAUSE request to wait for the quiescing condition to clear. At a high level, your program issues the pause request and XCF will not return until the note pad is no longer quiesced, at which point normal note activity can resume.



Pause Connection

IXCNOTE REQUEST=CONNECTION REQTYPE=PAUSE

- CONNECTION – token for connection to be deleted
- TIMEOUT - maximum duration of pause

In general, a pause connection request is issued when the application needs to wait for a particular event. For example, note requests are rejected when a quiescing condition is encountered (such as a rebuild of the structure hosting the note pad). In such cases, the application might prefer to stop issuing note requests until the note pad is once again accessible. If so, it would issue a pause connection request to wait for that event. Alternatively, if a connection is pending delete, the application might need to wait for the delete to finish before it proceeds. It could then issue a pause connection request to wait for that event.

For a given connection, at most one work unit can be paused. If the XCF Note Pad Services already have a work unit paused for the connection, a new pause connection request is rejected.

The pause connection request completes successfully when there no quiescing conditions. If the paused connection is resumed before the quiescing conditions clear, the return and reason code indicating that it was resumed. In these cases, the details stored in the answer area provide information about the current quiescing conditions. They also indicate why the pause request was resumed.

When a work unit is suspended by a pause connection request that was accepted by the XCF Note Pad Services, the service routine might return control to the program for any one of the following events:

•The quiescing conditions are eliminated

•The quiescing conditions are changed

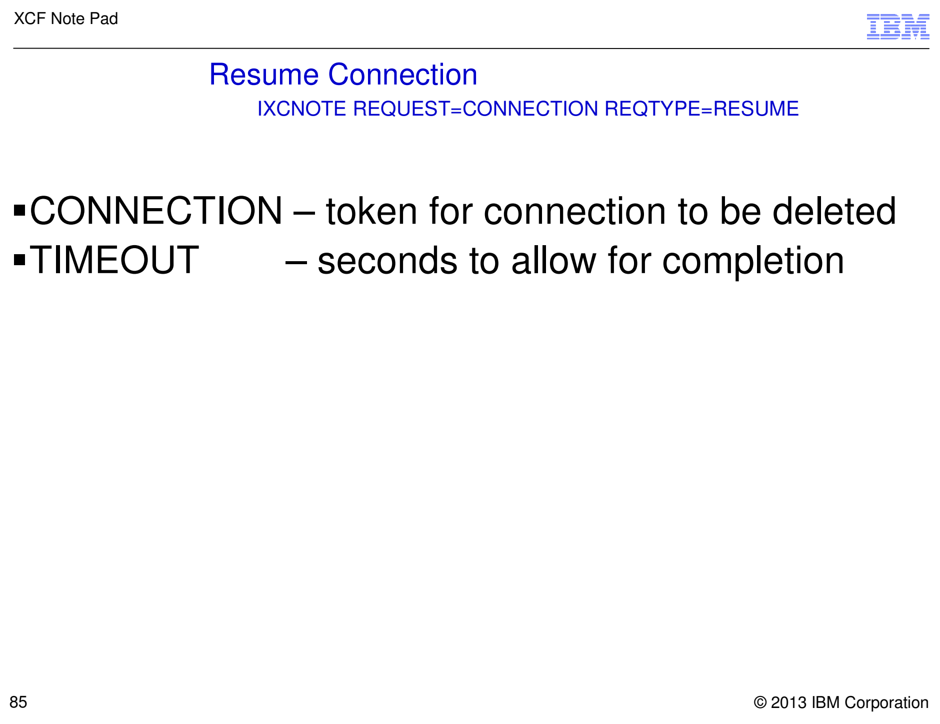
•The connection is deleted

•The note pad is deleted

•The timeout value expires

•An IXCNOTE resume connection request is issued

In particular, the fact that the pause connection request receives control back from the XCF Note Pad Services does not necessarily imply that the note pad is no longer quiesced. In general, it might be appropriate to examine the information returned in the answer area to determine whether the note pad is still quiesced. If so, it might be appropriate to reissue the pause connection request.



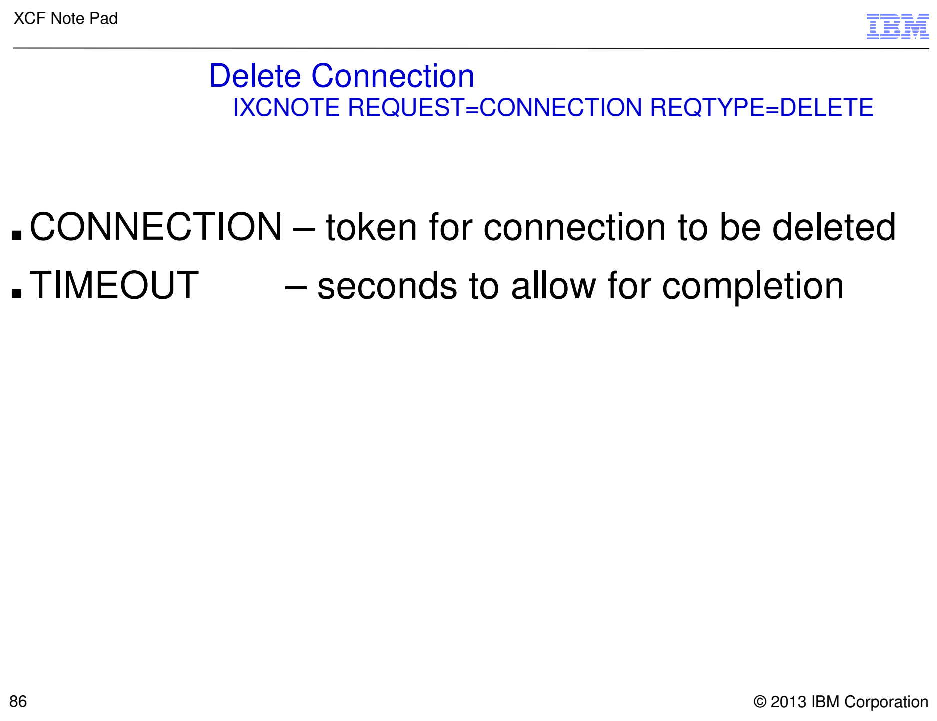
Resume Connection

IXCNOTE REQUEST=CONNECTION REQTYPE=RESUME

- CONNECTION – token for connection to be deleted
- TIMEOUT – seconds to allow for completion

I suspect that few applications will need to make use of the resume connection service. As needed, you issue a resume request to have XCF resume the work unit that issued the pause connection request. Normally XCF does not return to the work unit that issued the pause connection request until some event of interest occurs. The resume connection service gives you a way to force XCF to immediately return control to the work unit that issued the pause connection request.

XCF returns control to the invoker of the pause connection request when the connection is deleted. The fact that the connection is paused likely implies that the note pad is not accessible from the connector’s system. So there is not likely to be any need to resume the connection as part of a shut down procedure. However, there might be occasions where your application needs the paused work unit to wake up to perform some function. The resume connection request is intended for such occasions. At the risk of stating the obvious, the resume connection service must be invoked from some work unit other than the one that is paused.


XCF Note Pad

Delete Connection
IXCNOTE REQUEST=CONNECTION REQTYPE=DELETE

- CONNECTION – token for connection to be deleted
- TIMEOUT – seconds to allow for completion

86

© 2013 IBM Corporation

Delete the connection when it is no longer needed. Use the CONNECTION keyword to pass the connection token that identifies the connection to be deleted.

In general, several tasks must be accomplished to delete the connection. For example, XCF might need to fence the note pad in the host structure to prevent any further note processing by the connector, delete nonpersistent notes (KEEPNOTE=NO) associated with the connector, and update the note pad catalog. Usually the XCF Note Pad Services successfully perform these operations as needed. The connection is deleted and the service routine returns success.

However, XCF might not be able to perform some or all of these tasks if the host structure or the XCF catalog is not accessible. In such cases, the delete request completes with a return and reason code indicating that the delete of the connection is pending. XCF will automatically finish deleting the connection when it becomes possible to do so. Some applications can interpret this as being equivalent to a successful delete, and proceed. Some applications might not be able to proceed until the connection is known to have been fully deleted. For example, if this connector was the only connection permitted to have update access to the note pad (because the creator of the note pad specified MULTIWRITE=NO), a subsequent attempt to create a new connection with update access might be rejected until the pending delete is resolved. Alternatively, the application might have dependencies such that it cannot safely proceed until all its nonpersistent notes are known to have been deleted from the note pad.

If an application cannot safely proceed until the pending delete is resolved, consider issuing a pause connection request to wait for the connection to finish being deleted. The paused connection will be resumed when the delete is completed.

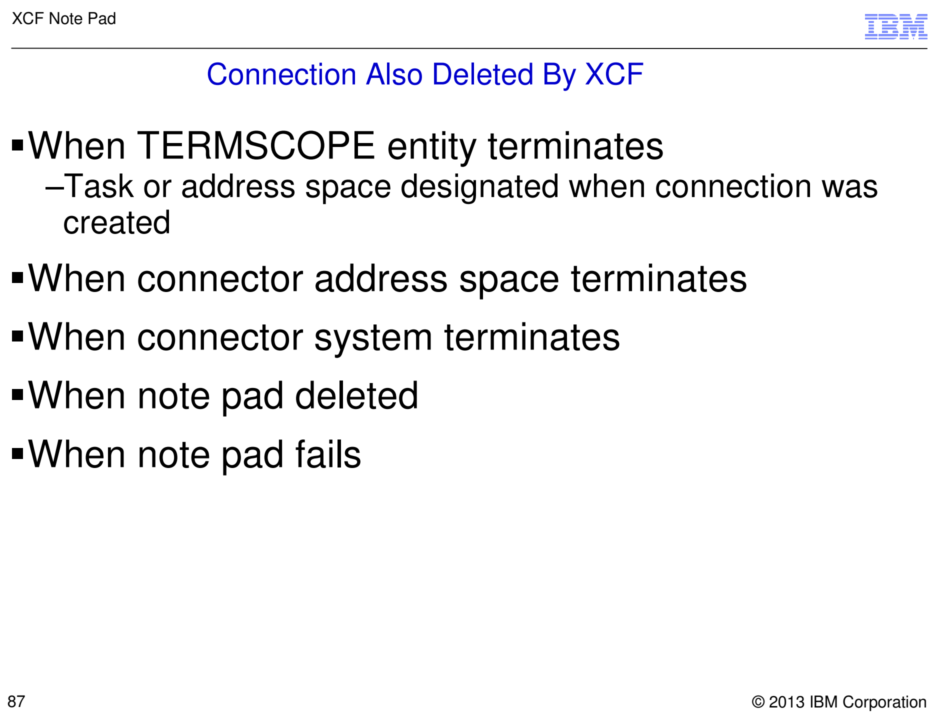
The delete request must be issued from a work unit that XCF deems to be a “valid user”. A work unit that can satisfy any one of the following qualifies:

o Requester is the connector

o Home is connector address space and the requester has SAF authorization appropriate for the ACCESS specified by the creator of the connection

o Running in supervisor state or with a PKM allowing key 0 to 7 and either:

- The creator of the connection specified USAGE=SERVER and primary is the connector address space, or
- The creator of the connection specified USAGE=CLIENT



Connection Also Deleted By XCF

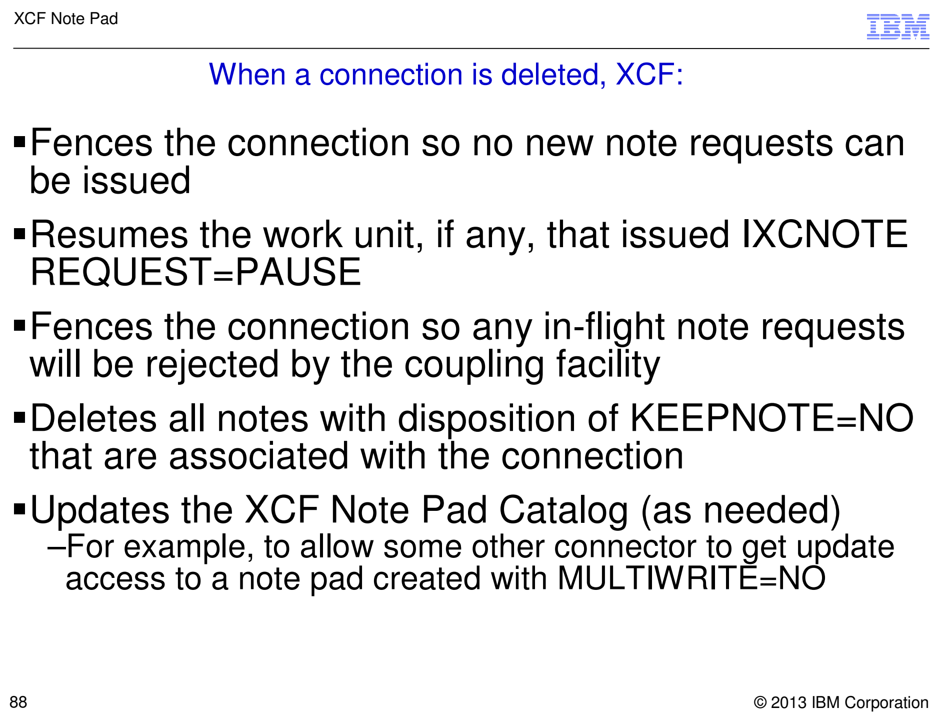
- When TERMSCOPE entity terminates
  - –Task or address space designated when connection was created
- When connector address space terminates
- When connector system terminates
- When note pad deleted
- When note pad fails

When your application invokes the IXCNOTE macro to create a connection, the TERMSCOPE keyword indicates an entity to which the connection is to be bound for termination purposes. When that entity terminates, XCF automatically deletes the connection. When a connection is deleted, its connection token can no longer be used to manipulate notes in the note pad. Depending on the TERMSCOPE specification and the USAGE specification, it could be the case that an in-flight note request in one address space could be rejected due to the termination of a task in some other address space.

When an address space terminates, XCF automatically deletes any connections associated with that address space. Similarly, if a system is removed from the sysplex, XCF automatically deletes any connections that were created by that system.

If a note pad fails, XCF deletes the note pad. When a note pad is deleted, XCF automatically deletes the connections to that note pad.

Note that XCF does not permit a resource manager to delete a connection. Thus you cannot delete your connection while running under the resource manager that you established for your TERMSCOPE entity. In general, this is not an issue because XCF deletes the connection for you.



When a connection is deleted, XCF:

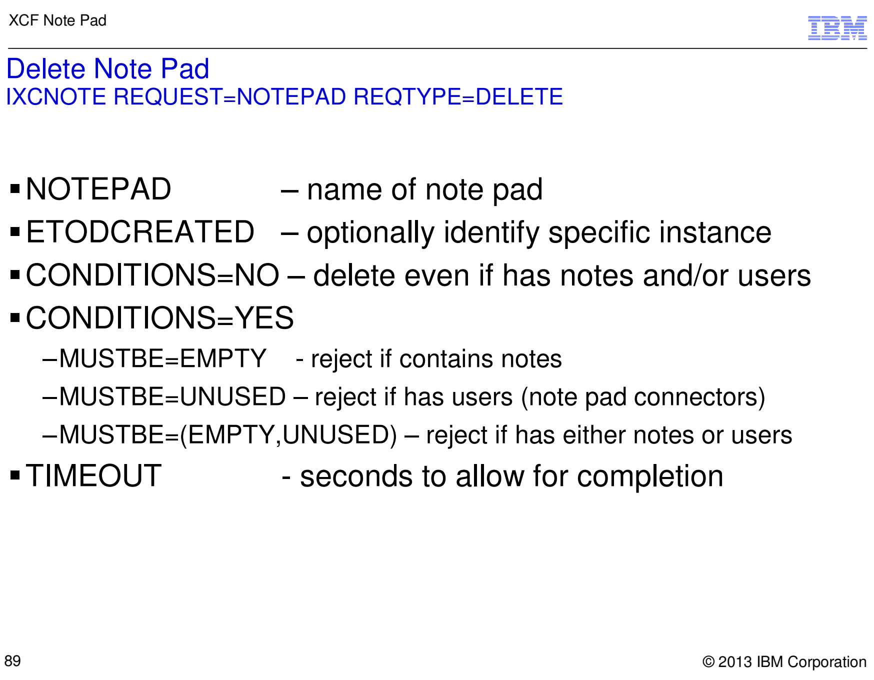
- Fences the connection so no new note requests can be issued
- Resumes the work unit, if any, that issued IXCNOTE REQUEST=PAUSE
- Fences the connection so any in-flight note requests will be rejected by the coupling facility
- Deletes all notes with disposition of KEEPNOTE=NO that are associated with the connection
- Updates the XCF Note Pad Catalog (as needed)
  - For example, to allow some other connector to get update access to a note pad created with MULTIWRITE=NO

Deleting a connection is a multi-step process. Some of these actions require access to the note pad structure and some require access to the XCF note pad catalog. If XCF is unable to access the structures, control will return to the request with an indication that the delete request is pending. XCF will automatically finish deleting the connection when the structures are once again accessible. In the meantime, the connection is marked for deletion and cannot be used to process notes. The pending delete could be visible in a number of ways until such time as XCF can finish deleting the connection:

•Notes written with KEEPNOTE=NO might persist in the note pad

•If the connection had update access to a note pad created with MULTIWRITE=NO, an attempt to create a new connection with update access might be rejected.

If the delete of the connection is pending, but your program needs to know when the delete is complete, you can issue a pause connection request to wait for XCF to finish up. The work unit that issues the pause request will be resumed after XCF finishes deleting the connection.



XCF Note Pad

Delete Note Pad
IXCNOTE REQUEST=NOTEPAD REQTYPE=DELETE

- NOTEPAD – name of note pad
- ETODCREATED – optionally identify specific instance
- CONDITIONS=NO – delete even if has notes and/or users
- CONDITIONS=YES
  - –MUSTBE=EMPTY - reject if contains notes
  - –MUSTBE=UNUSED – reject if has users (note pad connectors)
  - –MUSTBE=(EMPTY,UNUSED) – reject if has either notes or users
- TIMEOUT - seconds to allow for completion

89 © 2013 IBM Corporation

When a note pad is no longer needed, it should be deleted by the application.

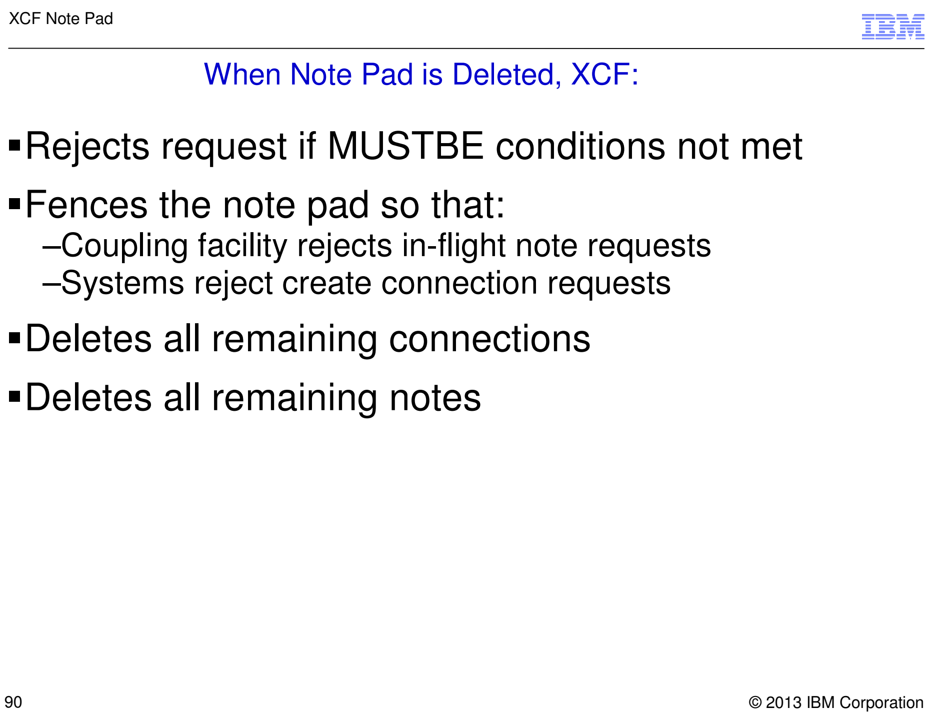
This slide depicts the various keywords that can be specified when deleting a note pad.

NOTEPAD is the name of the note pad to be deleted.

ETODCREATED can be used to identify the specific instance of the note pad to be deleted. It is an optional parameter containing the extended time of day (ETOD) time stamp when XCF created the note pad. If an answer area is provided, this ETOD is returned by the following requests: create note pad, query note pad, and create connection. If you might have competing threads creating and deleting a given note pad, you might want to specify ETODCREATED when deleting a note pad to ensure that the correct instance of the note pad is deleted.

Specify **CONDITIONS=NO** to have the note pad deleted unconditionally. The note pad will be deleted without regard to whether it contains notes or has connections. Both the notes and the connections will be deleted along with the note pad.

Specify **CONDITIONS=YES** to have the note pad deleted only if certain conditions can be satisfied. If the conditions cannot be satisfied, XCF rejects the request. The **MUSTBE** keyword indicates what conditions have to be satisfied. If the note pad must be empty, it will be deleted if it does not have any notes (and without regard to whether it has any connections). If the note pad must be unused, it will be deleted if it does not have any connections (and without regard to whether it contains any notes). If the note pad must be empty and unused, it will be deleted if it does not have any notes and does not have any connections.



XCF Note Pad

When Note Pad is Deleted, XCF:

- Rejects request if MUSTBE conditions not met
- Fences the note pad so that:
  - Coupling facility rejects in-flight note requests
  - Systems reject create connection requests
- Deletes all remaining connections
- Deletes all remaining notes

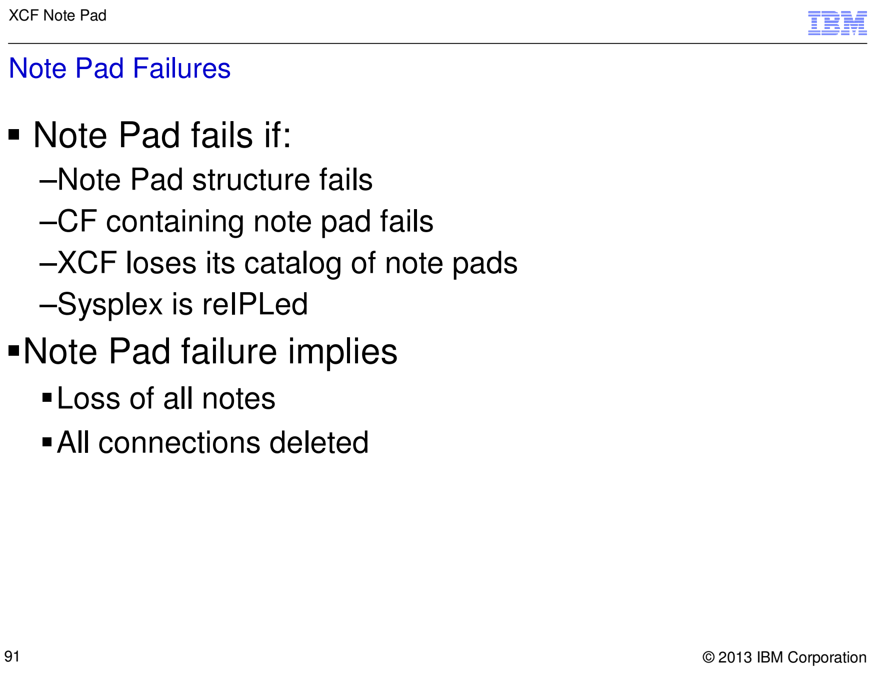
90

Deleting a note pad is a multi-step process. Some of these actions require access to the note pad structure and some require access to the XCF note pad catalog. If XCF is unable to access the structures, the delete request might be rejected outright. For example, XCF must access the note pad structure to determine whether a MUSTBE=EMPTY condition can be satisfied. Without access to the structure, XCF cannot determine whether the condition is satisfied, and so cannot proceed with the delete. In other cases, XCF might be able to mark the note pad for deletion but not be able to finish the delete due to lack of access to the structures. If so, control will return to the requester with an indication that the delete of the note pad is pending. XCF will automatically finish deleting the note pad when the structures are once again accessible. The pending delete could be visible in a number of ways until such time as XCF can finish deleting the note pad:

•The create of a new note pad might fail due to lack of space in the catalog structure

•The create of a new note pad might fail due to lack of space in the note pad structures

Note that the existence of a note pad in the pending delete state does not inherently prevent a new note pad from being created, not even one by the same name.



XCF Note Pad

Note Pad Failures

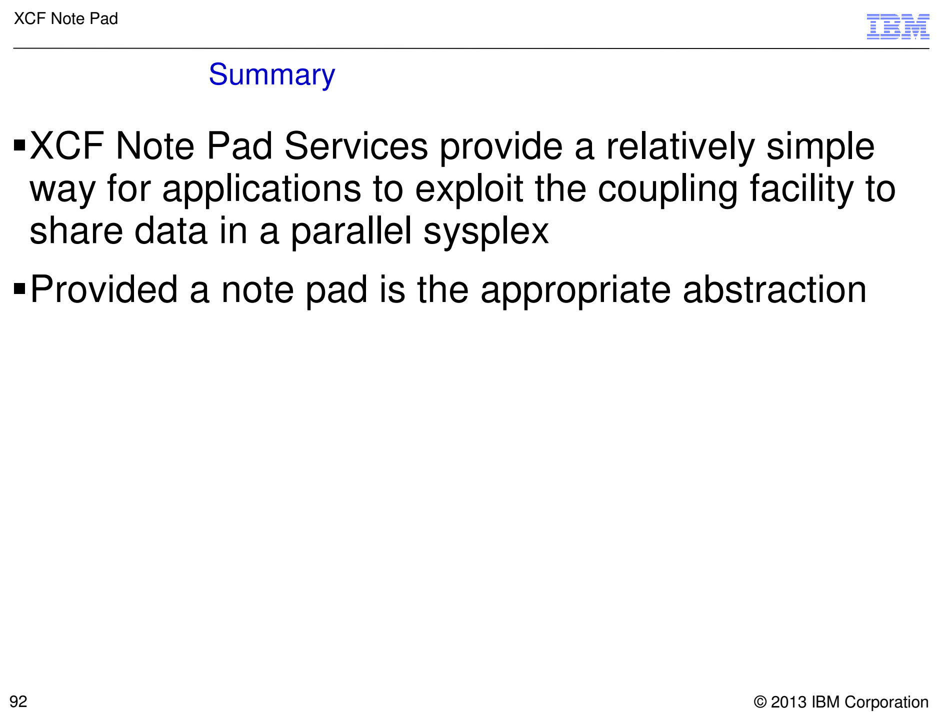
- Note Pad fails if:
  - –Note Pad structure fails
  - –CF containing note pad fails
  - –XCF loses its catalog of note pads
  - –Sysplex is reIPLed
- Note Pad failure implies
  - Loss of all notes
  - All connections deleted

91 © 2013 IBM Corporation

A note pad is deemed to have failed if the host structure or the coupling facility (CF) containing that structure fails (and there is no duplexed copy). All the note pads in the sysplex are deemed to have failed if XCF loses its catalog of note pads or if the sysplex is re-IPLed. When a note pad fails, all of its notes are lost and all of its connections are deleted.

If a system loses connectivity to the CF that contains the note pad, the note pad becomes inaccessible but it does not fail. If XCF loses access to the note pad catalog, it will disconnect (IXLDISC) from the note pad structures. In so doing, the note pad becomes inaccessible but it does not fail.

In general, the outright failure of a note pad should be rare. Similarly, loss of access to a note pad should not be frequent. However, such things do occur. As such, you must consider the potential ramifications to your application.



Summary

- XCF Note Pad Services provide a relatively simple way for applications to exploit the coupling facility to share data in a parallel sysplex
- Provided a note pad is the appropriate abstraction

Try it, you'll like it.

Summary …

- SAP exploits note pads
- The installation must
  - Configure the XCF Note Pad Catalog Structure
  - Configure the XCF Note Pad Structure(s)
  - Configure appropriate SAF protections
- The application must
  - Create and delete the note pad
  - Create and delete connections to the note pad
  - Create and delete notes in the note pad
  - Deal with failures
  - Provide documentation to the installation

An XCF note pad is shared storage that can be accessed by programs throughout the sysplex. Each note pad is hosted in a list structure in a coupling facility, and an additional structure is used to host the note pad catalog. To use the XCF Note Pad Services, the installation needs to define these structures in the Coupling Facility Resource Management (CFRM) policy.

The XCF Note Pad interface supports both authorized and unauthorized callers. With this interface, applications can create notes in a note pad that resides in a coupling facility structure. As such, the application has a relatively simple way to make information commonly available to all the systems in the sysplex.

The security administrator might have to define System Authorization Facility (SAF) profiles for certain note pads to grant access to the unauthorized callers. For authorized callers, XCF will honor the SAF profile if one is defined. If SAF is not installed, or the installation has not defined a SAF profile for the note pad, XCF rejects the request made by an unauthorized caller and permits the request to go forward for an authorized caller.

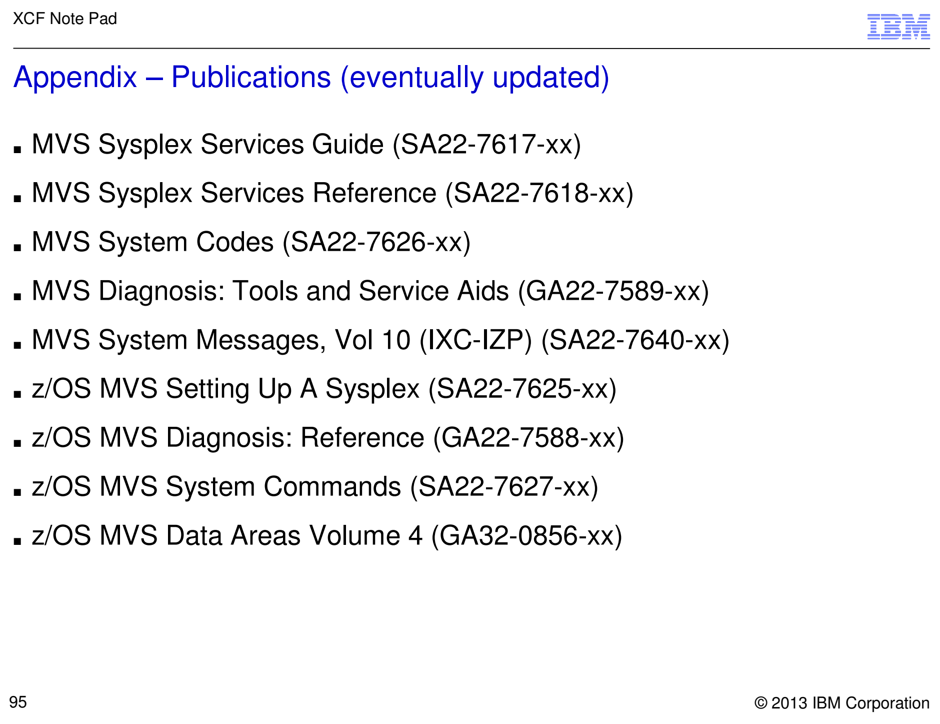
In addition, the installation might consider defining SAF profiles for the XCF catalog and note pad structures to prevent anyone but XCF from establishing XES connections to these structures. The data stored in the XCF catalog and note pad structures is managed solely by XCF, and allowing other programs to access the data directly through established XES connections could cause data integrity issues.

For More Information

- Documentation available on the web at:

publibz.boulder.ibm.com/zoslib/pdf/OA38450.pdf

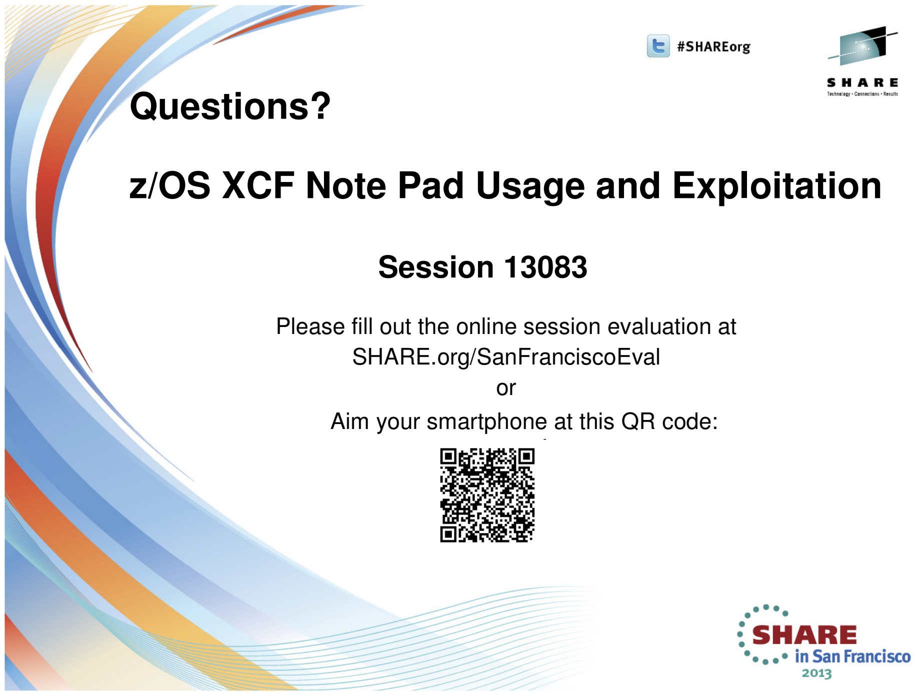
The indicated web site has the documentation for APAR OA38450, which provides support for XCF Note Pad Services on z/OS V1R13 and up. The information in this document will eventually be propagated to the various official z/OS publications.



Appendix – Publications (eventually updated)

- MVS Sysplex Services Guide (SA22-7617-xx)
- MVS Sysplex Services Reference (SA22-7618-xx)
- MVS System Codes (SA22-7626-xx)
- MVS Diagnosis: Tools and Service Aids (GA22-7589-xx)
- MVS System Messages, Vol 10 (IXC-IZP) (SA22-7640-xx)
- z/OS MVS Setting Up A Sysplex (SA22-7625-xx)
- z/OS MVS Diagnosis: Reference (GA22-7588-xx)
- z/OS MVS System Commands (SA22-7627-xx)
- z/OS MVS Data Areas Volume 4 (GA32-0856-xx)

The APAR documentation available at http://publibz.boulder.ibm.com/zoslib/pdf/OA38450.pdf will eventually be incorporated into the indicated publications. The indicated PDF file summarizes the updates to be made to each book.



#SHAREorg

Questions?

z/OS XCF Note Pad Usage and Exploitation

Session 13083

Please fill out the online session evaluation at
SHARE.org/SanFranciscoEval
or
Aim your smartphone at this QR code:

SHARE in San Francisco 2013

Thank you for your attention.

Feel free to email me at mabrook@us.ibm.com with questions, comments, or suggestions. I'd love to hear about applications that you build on top of these services.