Modes of Operation

Table of Contents

1 SNC Management Modes of Operation

In the context of the Interface, the target OS can manage the SNCs using different rules that best suit the particular application or architecture of the target OS. The target OS' behaviour regarding SNCs is called an "SNC management mode of operation".

Four different SNC management modes of operation can be used by an target OS. Each target OS operates in any one (but only one) of these four modes. They are mainly defined based on two "options" that the target OS may or may not support: each of the four combinations of these two options leads to a different mode of operation, as summarized in the following table.

SNC Management Mode Of Operation	Supports Pending State	**Supports Sharing Of Cross-Connects**
No Pending No Sharing	No	No
No Pending With Sharing	No	Yes
Pending No Sharing	Yes	No
Pending With Sharing	Yes	Yes

The two options "supports pending state" and "supports sharing of cross-connects" are described in details in the following subsections. Common rules that apply to all 4 modes are also described in the subsequent subsections. Each mode of operation is illustrated in Section 2, and Section 3 discusses the common applications for each mode.

1.1 "Supports Pending State" Option

The target OS is allowed to support or not the pending state for SNCs. The behaviour for each case is described below.

1.1.1 Pending State Not Supported

If the pending state is not supported, the SNC states are defined as follows:

- SNCS_PENDING: not used.
- SNCS_ACTIVE: a route has been assigned to the SNC and all cross-connects for the SNC are active in the network.
- SNCS_PARTIAL: either a route has not been assigned to the SNC, or at least 1 but not all of the cross-connects of the SNC are active in the network.
- SNCS_NONEXISTENT: this is not an SNC state per se, as it applies to “non-existent SNCs”. It is used in the interface to report SNCs that have been deleted.

Any SNC with an assigned route that has no non-shared active cross-connection in the network is considered non-existent (see Section 1.2 for a discussion on shared cross-connects). Consequently, when the last non-shared cross-connect of an SNC is deactivated in the network, the SNC is deleted by the target OS. In the case of network routed SNCs, it is possible that an SNC temporarily not be assigned a route, in which case it is considered in SNCS_PARTIAL state even though it has no non-shared active cross-connects.

SNCs can be created directly in active or partial state. Conversely, they can be deleted from either state. For an SNC to be deleted however, it must not have any non-shared active cross-connect.

The target OS does not support the createSNC, deactivateSNC, and deleteSNC operations because they all require the pending state. The activateSNC operation must still be supported for SNCs in partial state.

The target OS ensures that the requesting OS can not create and activate SNCs that conflict with other existing SNCs. Any attempt to do so will result in the reject of the create and activate request. An target OS that supports non-singleton networks can not however enforce the non-conflict rule from the network side. This is because a cross-connect that conflicts with an inactive cross-connect of a partial SNC can be created by means other than the TMF814 interface directly in the network (for example, if a currently inactive cross-connect between CTPs a and b belongs to SNC1, a craftsperson can create a cross-connect between CTPs a and c, which conflicts with SNC1 but which does not belong to it). In that particular case, the result is two conflicting SNCs. Such conflicts can be resolved by deactivating and deleting all but one of the conflicting SNCs.

1.1.2 Pending State Supported

If the pending state is supported, the SNC states are defined as follows:

- SNCS_PENDING: the SNC has been created by an requesting OS and has not been activated by any requesting OS; or the SNC has been successfully deactivated by an requesting OS. That state has no relationship with the network state of the cross-connects of the SNC.
- SNCS_ACTIVE: the SNC is not in pending state, a route has been assigned to the SNC, and all cross-connects for the SNC are active in the network.
- SNCS_PARTIAL: the SNC is not in pending state, and either a route has not been assigned to the SNC or not all of the cross-connects of the SNC are active in the network. This includes SNCs that have been activated by an requesting OS, but for which there are currently no active cross-connects in the network.
- SNCS_NONEXISTENT: this is not an SNC state per se, as it applies to “non-existent SNCs”. It is used in the interface to report SNCs that have been deleted.

All SNCs created by an requesting OS are created in pending state, and can from there transition to other states. SNCs created from the network are created directly in active or partial state.

In all cases, only SNCs in pending state can be deleted.

The target OS must support all 6 SNC operations. The createAndActivateSNC operation acts as a createSNC followed by an activateSNC. The deactivateAndDeleteSNC operation acts as a deactivateSNC followed by a deleteSNC.

The target OS allows the requesting OS to create SNCs in pending state that conflict with other existing SNCs. However, the target OS will not allow the target OS to activate pending SNCs that conflict with other active or partial SNCs: any attempt to do so will result in the reject of the activate request. An target OS that supports non-singleton networks can not however enforce the non-conflict rule from the network side. This is because a cross-connect that conflicts with an inactive cross-connect of a partial SNC can be created by a craftsperson in the network (for example, if a currently inactive cross-connect between CTPs a and b belongs to SNC1, a craftsperson can create a cross-connect between CTPs a and c, which conflicts with SNC1 but which does not belong to it). In that particular case, the result is two conflicting partial/active SNCs. Such conflicts can be resolved by deactivating all but one of the conflicting SNCs.

1.2 "Supports Sharing Of Cross-Connects" Option

With some restrictions, the target OS is allowed to support or not sharing of cross-connects amongst SNCs. The behaviour for each case is described below.

1.2.1 Sharing Of Cross-Connects Not Supported

If sharing of cross-connects is not supported, the target OS ensures that cross-connects are not shared amongst partial and active SNCs. Each cross-connect, active or not, can only belong to at most one partial or active SNC. An attempt to violate this rule will result in the reject of the activate operation.

Sharing of cross-connects (as well as conflicting cross-connects) are allowed for SNCs in pending state. However, activation of a pending SNC that shares a CC with an already active or partial SNC will be rejected.

In the case of non-singleton subnetworks, an exception to the non-sharing rule is made for SNCs of a broadcast system, in which case the cross-connects near the source of the broadcast may be shared. "Near the source" means all cross-connects along the broadcast paths until the last fork on a path (from the last fork to the sink CTP, CCs belong to one and only one SNC). This is an exception due to the related nature of these SNCs. SNCs that are part of a broadcast system can easily be identified because (1) they are all unidirectional, (2) the source CTP of every SNC is used by at least one other SNC in the system, and (3) they all have a different sink CTP.

1.2.2 Sharing Of Cross-Connects Supported

In this mode, the target OS allows unrestricted sharing of cross-connects amongst SNCs. A cross-connect can belong to any number of SNCs. SNC creation or activation will not be rejected if the SNC to be created or activated shares one or more CCs with other existing SNCs.

1.3 Deactivating SNCs

Deactivating an SNC that shares cross-connects with other SNCs does not necessarily mean that all its cross-connects must be deactivated. Deactivating an SNC is defined as deactivating only the cross-connects of the SNC that are not shared with other active or partial SNCs; shared ones must not be deactivated. Note that CCs shared with pending SNCs are not considered, so if a CC is part of two SNCs, a pending SNC and an active SNC, deactivating the active SNC will deactivate the CC.

This definition of SNC deactivation applies to all modes of operation.

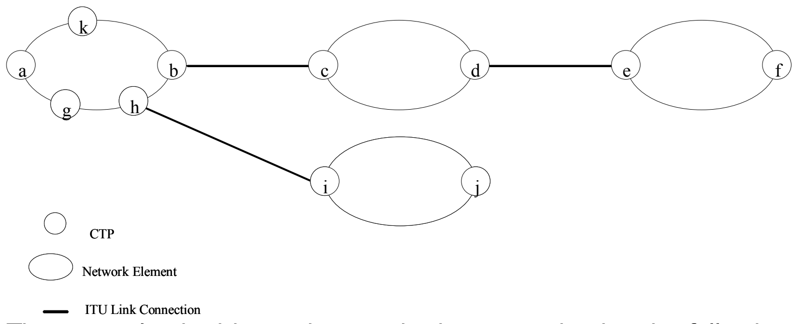
1.4 Cross-Connect Representations

In order for the target OS to provide the current network configuration, every active cross-connect in the network must belong to at least one active or partial SNC. That is, if a CC that does not already belong to any active or partial SNC is created/detected in the network, the target OS will create an active or partial SNC to represent that cross-connect or will include that CC in an already existing active or partial SNC.

If a CC that only belongs to pending SNCs is created/detected in the network, the target OS will create or modify a different SNC, which is in active or partial state, to represent that CC. This is required to be consistent with the pending state definition, while still allowing the requesting OS to be informed of changes in the network.

The rule of this section applies to all modes of operation.

2 Examples



The examples in this section can be interpreted using the following network drawing.

2.1 "No Pending No Sharing" Mode

The goals of this mode are for the target OS to represent only the current network configuration, to limit as much as possible sharing of resources amongst SNCs, and to attempt to have a one-to-one correspondence between the network configurations and the SNC configurations.

The following table shows examples of how an target OS in the "No Pending No Sharing" mode of operation would react to different requests. "SNCn" refers to an SNC, while "x-y" refers to a simple cross-connect between CTPs x and y. All SNCs and CCs are bidirectional, so there are no broadcast SNCs in this example.

Action	Generated SNC Notifications	SNCs		
		PENDING	PARTIAL	ACTIVE
Create&Activate SNC1 (a-b). CC a-b is activated.	OC(SNC1)			**SNC1**
Create&Activate SNC2 (a-b, c-d). Rejected because of shared CC a-b.				SNC1
Create&Activate SNC3 (e-f). Activation of e-f fails.				SNC1
Create&Activate SNC4 (g-h, i-j). CC g-h is activated, but activation of i-j fails.	OC(SNC4)		**SNC4**	SNC1
Create&Activate SNC5 (a-b). Rejected because of shared CC a-b.			SNC4	SNC1
Create&Activate SNC6 (a-k). Rejected because of conflicting SNC1.			SNC4	SNC1
CC i-j is activated outside the interface.	SC(SNC4)			SNC1 **SNC4**
CC a-b is deactivated outside the interface.	OD(SNC1)			SNC4
CC g-h is deactivated outside the interface.	SC(SNC4)		**SNC4**	
g-k is activated outside the interface. Since this cross-connect is not part of any SNC, a new SNC is created by the target OS.	OC(SNC7)		SNC4	**SNC7**
Deactivate&Delete SNC7. CC g-k is deactivated.	OD(SNC7)		SNC4	

2.2 "No Pending With Sharing" Mode

The goals of this mode are for the target OS to represent only the current network configuration. However, contrary to the "No Pending No Sharing" mode, it does not limit sharing of CCs amongst SNCs.

The following table shows examples of how an target OS in the "no pending with sharing" mode of operation would react to different requests. "SNCn" refers to an SNC, while "x-y" refers to a simple cross-connect between CTPs x and y. All SNCs and CCs are bidirectional, so there are no broadcast SNCs in this example.

Action	Generated SNC Notifications	SNCs		
		PENDING	PARTIAL	ACTIVE
Create&Activate SNC1 (a-b). CC a-b is activated.	OC(SNC1)			**SNC1**
Create&Activate SNC2 (a-b, c-d). CC c-d is activated (CC a-b already is).	OC(SNC2)			SNC1 **SNC2**
Create&Activate SNC3 (e-f). Activation of e-f fails.				SNC1 SNC2
Create&Activate SNC4 (g-h, i-j). CC g-h is activated, but activation of i-j fails.	OC(SNC4)		**SNC4**	SNC1 SNC2
Create&Activate SNC5 (a-b). CC a-b is already active.	OC(SNC5)		SNC4	SNC1 SNC2 **SNC5**
Create&Activate SNC6 (a-k). Rejected because of conflicting SNC1, SNC2, and SNC5.			SNC4	SNC1 SNC2 SNC5
Deactivate&delete SNC5. Since CC a-b is shared with SNC1 and SNC2, it is not deactivated.	OD(SNC5)		SNC4	SNC1 SNC2
CC i-j is activated outside the interface.	SC(SNC4)			SNC1 SNC2 **SNC4**

CC a-b is deactivated outside the interface.	OD(SNC1)
	SC(SNC2)
			SNC2	SNC4

Action	Generated SNC Notifications	SNCs		
		PENDING	PARTIAL	ACTIVE
	OD(SNC5)			
CC g-h is deactivated outside the interface.	SC(SNC4)		SNC2 **SNC4**	
CC g-k is activated outside the interface. Since this cross-connect is not part of any SNC, a new SNC is created by the target OS.	OC(SNC7)		SNC2 SNC4	**SNC7**
Deactivate&Delete SNC7. CC g-k is deactivated.	OD(SNC7)		SNC2 SNC4	

2.3 "Pending No Sharing" Mode

The goals of this mode are for the target OS to represent the current network configuration as well as potential "future" SNCs that have been prepared by the requesting OS, but not yet activated.

The following table shows examples of how an target OS in the "Pending No Sharing" mode of operation would react to different requests. "SNCn" refers to an SNC, while "x-y" refers to a simple cross-connect between CTPs x and y. All SNCs and CCs are bidirectional, so there are no broadcast SNCs in this example.

Action	Generated Notifications	SNCs		
		PENDING	PARTIAL	ACTIVE
Create SNC1 (a-b).	OC(SNC1)	**SNC1**		
Activate SNC1. CC a-b is activated.	SC(SNC1)			**SNC1**
Create SNC2 (a-b, c-d).	OC(SNC2)	**SNC2**		SNC1
Activate SNC2. Rejected because of shared CC a-b.		SNC2		SNC1
Create SNC3 (e-f).	OC(SNC3)	**SNC3** SNC2		SNC1
Activate SNC3. Activation of e-f fails.	SC(SNC3)	SNC2	**SNC3**	SNC1

Action	Generated Notifications	SNCs		
		PENDING	PARTIAL	ACTIVE

Create&Activate SNC4 (g-h, i-j). CC g-h is activated, but activation of i-j fails.	OC(SNC4)
	SC(SNC4)
		SNC2	SNC3
			SNC4
				SNC1

Create SNC5 (a-b).	OC(SNC5)	SNC2 **SNC5**	SNC3 SNC4	SNC1
Activate SNC5. Rejected because of shared CC a-b.		SNC2 SNC5	SNC3 SNC4	SNC1
Create SNC6 (a-k).	OC(SNC6)	SNC2 SNC5 SNC6	SNC3 SNC4	SNC1
Activate SNC6. Activation is rejected because it conflicts with SNC1.		SNC2 SNC5 SNC6	SNC3 SNC4	SNC1
Delete SNC5.	OD(SNC5)	SNC2 SNC6	SNC3 SNC4	SNC1
CC i-j is activated outside the interface.	SC(SNC4)	SNC2 SNC6	SNC3	SNC1 **SNC4**
CC a-b is deactivated outside the interface.	SC(SNC1)	SNC2 SNC6	**SNC1** SNC3	SNC4
CC g-h is deactivated outside the interface.	SC(SNC4)	SNC2 SNC6	SNC1 SNC3 **SNC4**	
CC g-k is activated outside the interface. Since this cross-connect is not part of any SNC, a new SNC is created by the target OS.	OC(SNC7)	SNC2 SNC6	SNC1 SNC3 SNC4	**SNC7**

Deactivate&Delete SNC7.	SC(SNC7)
	OD(SNC7)
		SNC2
		SNC6
			SNC1
			SNC3
			SNC4	

2.4 "Pending With Sharing" Mode

The goals of this mode are for the target OS to represent the current network configuration as well as potential "future" SNCs that have been prepared by the requesting OS, but not yet activated. It also allows sharing of CCs amongst SNCs.

The following table shows examples of how an target OS in the "pending with sharing" mode of operation would react to different requests. "SNCn" refers to an SNC, while "x-y" refers to a simple cross-connect between CTPs x and y. All SNCs and CCs are bidirectional, so there are no broadcast SNCs in this example.

Action	**Generated Notifications**	SNCs		
		PENDING	**PARTIAL**	**ACTIVE**
Create SNC1 (a-b).	OC(SNC1)	**SNC1**		
Activate SNC1. CC a-b is activated.	SC(SNC1)			**SNC1**
Create SNC2 (a-b, c-d).	OC(SNC2)	**SNC2**		SNC1
Activate SNC2. CC c-d is activated (CC a-b already is).	SC(SNC2)			SNC1 **SNC2**
Create SNC3 (e-f).	OC(SNC3)	**SNC3**		SNC1 SNC2
Activate SNC3. Activation of e-f fails.	SC(SNC3)		**SNC3**	SNC1 SNC2

Create&Activate SNC4 (g-h, i-j). CC g-h is activated, but activation of i-j fails.	OC(SNC4)
	SC(SNC4)
			SNC3
			SNC4
				SNC1
				SNC2

Create SNC5 (a-b).	OC(SNC5)	**SNC5**	SNC3 SNC4	SNC1 SNC2
Activate SNC5. CC a-b is already active.	SC(SNC5)		SNC3 SNC4	SNC1 SNC2 **SNC5**
Create SNC6 (a-k).	OC(SNC6)	SNC6	SNC3 SNC4	SNC1 SNC2 SNC5
Activate SNC6 (a-k). Activation is rejected because it conflicts with SNC1, SNC2, and SNC5.		SNC6	SNC3 SNC4	SNC1 SNC2 SNC5

Action	Generated Notifications	SNCs		
		PENDING	PARTIAL	ACTIVE
Deactivate SNC5. Since CC a-b is shared with SNC1 and SNC2, it is not deactivated.	SC(SNC5)	**SNC5** SNC6	SNC3 SNC4	SNC1 SNC2
Delete SNC5.	OD(SNC5)	SNC6	SNC3 SNC4	SNC1 SNC2
CC i-j is activated outside the interface.	SC(SNC4)	SNC6	SNC3	SNC1 SNC2 **SNC4**

CC a-b is deactivated outside the interface.	SC(SNC1)
	SC(SNC2)
		SNC6	**SNC1**
			SNC2
			SNC3
				SNC4

CC g-h is deactivated outside the interface.	SC(SNC4)	SNC6	SNC1 SNC2 SNC3 **SNC4**	
CC g-k is activated outside the interface. Since this cross-connect is not part of any SNC, a new SNC is created by the target OS.	OC(SNC7)	SNC6	SNC1 SNC2 SNC3 SNC4	**SNC7**

Deactivate&Delete SNC7.	SC(SNC7)
	OD(SNC7)
		SNC6	SNC1
			SNC2
			SNC3
			SNC4	

3 Applications

All four modes, as defined in the preceding section, can apply to any target OS. However, because of their specific characteristics, some modes are better suited to particular applications.

The “No Pending No Sharing” mode is best suited to target OSes that only support singleton subnetworks and that do not keep a database of pending SNCs. In that case, this mode fully enforces non-sharing of CCs and non-conflicting SNCs (as explained earlier, it is not possible to fully enforce them in non-singleton subnetworks). It also provides a perfect one-to-one correspondence between network CCs and SNCs, where each CC is represented by exactly one SNC.

The “no pending mode with sharing” is best suited to target OSes that support non-singleton subnetworks and that do not keep a database of pending SNCs. In that case, this mode may be preferable to the “No Pending No Sharing” mode, because it eliminates exceptions in the non-sharing rule (required in the other mode because of SNCs of broadcast systems in non-singleton subnetworks), and because it allows SNC reorganizations without traffic interruption (only useful in non-singleton subnetworks). For example, if the target OS currently has two “consecutive” SNCs that the requesting OS wants to merge into one “larger” SNC, this can be done without interrupting traffic by creating and activating the larger SNC (which shares all its CCs with the two consecutive SNCs), then deactivating and deleting the two consecutive SNCs.

The “Pending No Sharing” mode is best suited to target OSes that wish to keep a database of pending SNCs and that only support singleton subnetworks. This mode is similar to the “No Pending No Sharing” mode, except for the fact that it provides pending SNCs.

Finally, the “pending with sharing” mode is best suited to target OSes that wish to keep a database of pending SNCs and that support non-singleton subnetworks. This mode is similar to the “no pending with sharing” mode, except for the fact that it provides pending SNCs.

4 Behavior of Operations on Subnetwork Connections (SNC)

These bullets explain the use of SNCs and SNC names in the table below:

- The letter in quotes represents the unique name of the SNC.
- SNC A is an SNC that exists in the target OS in the active state
- SNC B is an SNC that exists in the target OS in the partial state
- SNC C does not exist in the target OS, i.e., there is no SNC named “C” in the target OS
- SNC D is an SNC that exists in the target OS in the pending state
- SNC E is used to indicate a new SNC with a name different from any other SNC in the target OS

In the table below, when an SNC is used as input to Create or CreateAndActivate, it means that the create data is such that the referenced SNC respects all the conditions of the create data (e.g., aEnd TPs, zEnd TPs, SNCType). When an SNC is used as input to Activate, Deactivate, Delete, or DeactivateAndDelete, it means that the SNC name is used as input to the operation. Thus SNC A as input to one of these operations means the name “A” is the name input to the operation.

In all cases, an errorReason is returned if the operation is not entirely successful.

Mode of Operation	IDL Operation	Return for input SNC “A” (Active)	Return for input SNC “B” (Partial)	Return for input SNC “C” (New)	Return for input SNC “D” (Pending)
No pending, No sharing	DeactivateAndDelete	SNC A Nonexistent SNC A Partial[1] SNC A Active[1]	SNC B Nonexistent SNC B Partial[1]	Exception	
	Activate	SNC A Active[2]	SNC B Active SNC B Partial[3]	Exception	
	CreateAnd Activate	SNC A Active[4] Exception[5,8]	SNC B Active[4] SNC B Partial[4,3] Exception[5,8]	SNC C Active SNC C Partial[3] Exception[5,6,8]	
No pending, Sharing	DeactivateAndDelete	SNC A Nonexistent SNC A Partial[1] SNC A Active[1]	SNC B Nonexistent SNC B Partial[1]	Exception	
	Activate	SNC A Active[2]	SNC B Active SNC B Partial[3]	Exception	

	CreateAndActivate	SNC A Active[4] SNC E Active[7] Exception[8]	SNC B Active[4] SNC B Partial[4,3] SNC E Active[7] SNC E Partial[7,3] Exception[8]	SNC C Active SNC C Partial[3] Exception[5,6,8]	
Pending, No sharing	Create	SNC A Active[4] SNC E Pending Exception[8]	SNC B Partial[4] SNC E Pending Exception[8]	SNC C Pending Exception[8]	SNC D Pending[4] SNC E Pending Exception[8]
	Activate	SNC A Active[2]	SNC B Active SNC B Partial[3]	Exception	SNC D Active SNC D Partial[3] Exception[11]
	CreateAndActivate	SNC A Active[4] SNC E Pending[10] Exception[8]	SNC B Active[4] SNC B Partial[4,3] SNC E Pending[10] Exception[8]	SNC C Active SNC C Partial[3] SNC C Pending[11] Exception[8]	SNC D Active[4] SNC D Partial[4,3] Exception[11] SNC E Pending[11] Exception[8]
	Deactivate	SNC A Pending SNC A Partial[1] SNC A Active[1]	SNC B Pending SNC B Partial[1]	Exception	SNC D Pending[12]
	Delete	Exception[13]	Exception[13]	Exception	SNC D deleted
	DeactivateAndDelete	SNC A Nonexistent SNC A Partial[1] SNC A Active[1]	SNC B Nonexistent SNC B Partial[1]	Exception	SNC D Nonexistent
Pending, Sharing	Create	SNC A Active[4] SNC E Pending Exception[8]	SNC B Partial[4] SNC E Pending Exception[8]	SNC C Pending Exception[8]	SNC D Pending[4] SNC E Pending Exception[8]
	Activate	SNC A Active[2]	SNC B Active SNC B Partial[3]	Exception	SNC D Active SNC D Partial[3] Exception[11]

	CreateAndActivate	SNC A Active[4] SNC E Active Exception[8]	SNC B Active[4] SNC B Partial[4,3] SNC E Active SNC E Partial[3] Exception[8]	SNC C Active SNC C Partial[3] SNC C Pending[11] Exception[8]	SNC D Active[4] SNC D Partial[4,3] Exception[11] SNC E Active SNC E Partial[3] SNC E Pending[11] Exception[8]
	Deactivate	SNC A Pending SNC A Partial[1] SNC A Active[1]	SNC B Pending SNC B Partial[1]	Exception	SNC D Pending[12]
	Delete	Exception[13]	Exception[13]	Exception	SNC D deleted
	DeactivateAndDelete	SNC A Nonexistent SNC A Partial[1] SNC A Active[1]	SNC B Nonexistent SNC B Partial[1]	Exception	SNC D Nonexistent

1 Deactivation fails or is not fully successful. An errorReason is supplied.

2 SNC remains active. Transmission parameters may have changed.

3 Activation is not fully successful. An errorReason is supplied.

4 Create and CreateAndActivate are allowed to reuse an existing SNC. An errorReason indicating reuse may be supplied.

5 An exception to indicate a conflict with an existing SNC may be thrown.

6 An exception is thrown if no cross-connect can be activated.

7 A new SNC may be returned since sharing of cross-connects is allowed. The new SNC will be in active state if the activation is successful; otherwise, it will be in partial state. This may not occur if a unique userLabel is specified and it is the same userLabel as the existing SNC.

8 An exception may be thrown if the forceUniqueness flag is set and the specified userLabel is the same as that of an existing SNC (an existing SNC that has the same userLabel as that specified may be reused and returned, in which case this exception is not thrown).

10 CreateAndActivate can create a new SNC, but it would not be allowed to activate it since sharing is not allowed. The SNC would remain in pending state and an errorReason is supplied.

11 In activating a pending SNC, if the SNC is in conflict with another active or partial SNC, an exception is thrown (except in the case of CreateAndActivate which returns the pending SNC instead).

12 Deactivation of the pending SNC is successful. An errorReason is supplied.

13 An SNC may not be deleted unless it is in pending state. An attempt to do so will throw an exception.

5 Administrative Appendix

5.1 Document History

Version	Date	Description of Change
3.0	June 2005	Conversion of modesOfOperation into new template

5.2 Acknowledgments

First Name	Last Name	Company

5.3 How to comment on this document

Comments and requests for information must be in written form and addressed to the contact identified below:

Keith	Dorking	CIENA
Phone:	+1 678 867 5007	
Fax:	+1 678 867 5010	
e-mail:	Kdorking@ciena.com	

Please be specific, since your comments will be dealt with by the team evaluating numerous inputs and trying to produce a single text. Thus we appreciate significant specific input. We are looking for more input than wordsmith” items, however editing and structural help are greatly appreciated where better clarity is the result.