



Dialogic® Converged Services Platform SwitchKit® Development Environment

Installation & Maintenance Guide

Copyright and Legal Disclaimer

Copyright © [1998-2008] Dialogic Corporation. All Rights Reserved. You may not reproduce this document in whole or in part without permission in writing from Dialogic Corporation at the address provided below.

All contents of this document are subject to change without notice and do not represent a commitment on the part of Dialogic Corporation or its subsidiaries. Reasonable effort is made to ensure the accuracy of the information contained in the document. However, due to ongoing product improvements and revisions, Dialogic Corporation and its subsidiaries do not warrant the accuracy of this information and cannot accept responsibility for errors or omissions that may be contained in this document.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH DIALOGIC® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS EXPLICITLY SET FORTH BELOW OR AS PROVIDED IN A SIGNED AGREEMENT BETWEEN YOU AND DIALOGIC, DIALOGIC ASSUMES NO LIABILITY WHATSOEVER, AND DIALOGIC DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF DIALOGIC PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY INTELLECTUAL PROPERTY RIGHT OF A THIRD PARTY.

Dialogic products are not intended for use in medical, life saving, life sustaining, critical control or safety systems, or in nuclear facility applications.

It is possible that the use or implementation of any one of the concepts, applications, or ideas described in this document, in marketing collateral produced by or on web pages maintained by Dialogic Corporation or its subsidiaries may infringe one or more patents or other intellectual property rights owned by third parties. Dialogic Corporation or its subsidiaries do not provide any intellectual property licenses with the sale of Dialogic products other than a license to use such product in accordance with intellectual property owned or validly licensed by Dialogic Corporation or its subsidiaries. More detailed information about such intellectual property is available from Dialogic Corporation's legal department at 9800 Cavendish Blvd., 5th Floor, Montreal, Quebec, Canada H4M 2V9. The software referred to in this document is provided under a Software License Agreement. Refer to the Software License Agreement for complete details governing the use of the software.

Dialogic Corporation encourages all users of its products to procure all necessary intellectual property licenses required to implement any concepts or applications and does not condone or encourage any intellectual property infringement and disclaims any responsibility related thereto. These intellectual property licenses may differ from country to country and it is the responsibility of those who develop the concepts or applications to be aware of and comply with different national license requirements.

Dialogic, Dialogic Pro, Brooktrout, Cantata, SnowShore, Eicon, Eicon Networks, Eiconcard, Diva, SIPcontrol, Diva ISDN, TruFax, Realblobs, Realcomm 100, NetAccess, Instant ISDN, TRXStream, Exnet, Exnet Connect, EXS, ExchangePlus VSE, Switchkit, N20, Powering The Service-Ready

Network, Vantage, Connecting People to Information, Connecting to Growth, Making Innovation Thrive, and Shiva, among others as well as related logos, are either registered trademarks or trademarks of Dialogic.

Windows and Windows NT are registered trademarks of Microsoft Corporation in the United States and/or other countries. Other names of actual companies and products mentioned herein are the trademarks of their respective owners.

This document discusses one or more open source products, systems and/or releases. Dialogic is not responsible for your decision to use open source in connection with Dialogic products (including without limitation those referred to herein), nor is Dialogic responsible for any present or future effects such usage might have, including without limitation effects on your products, your business, or your intellectual property rights.

Dialogic Product Line Warranty

Unless otherwise stated in an applicable product purchase agreement between the Customer and Dialogic, Dialogic warrants that during the Warranty Period, products will operate in substantial conformance with Dialogic's standard published documentation accompanying the product. If a product does not operate in accordance therewith during the Warranty Period, the Customer must promptly notify Dialogic. Dialogic, at its option, will either repair or replace the product without charge. The Customer has the right, as their exclusive remedy, to return the product for a refund of purchase price or license fee if Dialogic is unable to repair or replace it.

Warranty Period

In the event that you have no signed agreement setting out a warranty period, the Warranty Period shall be the standard warranty period set out on www.dialogic.com on the date of your purchase of the product.

The Warranty Period begins on the date of shipment of any products or software by Dialogic.

The Warranty Period for repaired, replaced or corrected products and software shall be coterminous to the Warranty Provided for the original products or software purchased.

To report warranty claims, Customer may contact Dialogic via email at techsupport@cantata.com or call (781) 433-9600.

Warranty Provisions

A. During the Warranty Period, Dialogic warrants to Customer only that:

- (i) Products manufactured by Dialogic (including those manufactured for Dialogic by an original equipment manufacturer) will be free from defects in material and workmanship and will substantially conform to specifications for such products;
- (ii) software developed by Dialogic will be free from defects which materially affect performance in accordance with the specifications for such software. With respect to products or software or partial assembly of products furnished by Dialogic but not manufactured by Dialogic, Dialogic hereby assigns to Customer, to the extent permitted, the warranties given to Dialogic by its vendors of such items.

B. If, under normal and proper use, a defect or non conformity appears in warranted products or software during the applicable Warranty Period and Customer promptly notifies Dialogic in writing during the applicable warranty period of such defect or non conformance, and follows Dialogic's instructions regarding return of such defective or non conforming Product or Software, then Dialogic will, at no charge to Customer, either:

- (i) repair, replace or correct the same at its manufacturing or repair facility or
- (ii) if Dialogic determines that it is unable or impractical to repair, replace or correct the product or software, provide a refund or credit not to exceed the original purchase price or license fee.

C. No product or software will be accepted for repair or replacement without the written authorization of and in accordance with instructions from Dialogic. Removal and reinstallation expenses as well as transportation expenses associated with returning such product or software to Dialogic shall be borne by Customer. Dialogic shall pay the costs of transportation of the repaired or replaced product or software to the destination designated in the original Order. If Dialogic determines that any returned product or software is not defective, Customer shall pay Dialogic's costs of handling, inspecting, testing and transportation. In repairing or replacing any product, part of product, or software medium under this warranty, Dialogic may use new, remanufactured, reconditioned, refurbished or functionally equivalent products, parts or software media. Replaced products or parts shall become Dialogic's property.

D. Dialogic makes no warranty with respect to defective conditions or non conformities resulting from any of the following: Customer's modifications, misuse, neglect, accident or abuse; improper wiring, repairing, splicing, alteration, installation, storage or maintenance performed in a manner not in accordance with Dialogic's or its vendor's specifications, or operating instructions; failure of Customer to apply Dialogic's previously applicable modifications or corrections; or items not manufactured by Dialogic or purchased by Dialogic pursuant to its procurement specifications. Dialogic makes no warranty with respect to products which have had their serial numbers removed or altered; with respect to expendable items, including, without limitation, fuses, light bulbs, motor brushes and the like; or with respect to defects related to Customer's data base errors. Improper packaging of product for repair will not be covered under this warranty agreement. No warranty is made that software will run uninterrupted or error free.

E. Warranty does not include:

- a) Dialogic's assistance in diagnostic efforts;
- b) access to Dialogic's Technical Support web sites, databases or tools;
- c) product integration testing;
- d) on-site assistance; or
- e) product documentation updates.

These services are available either during or after warranty at Dialogic's published prices.

F. THE FOREGOING WARRANTIES ARE EXCLUSIVE & ARE GRANTED IN LIEU OF ALL OTHER EXPRESS & IMPLIED WARRANTIES (WHETHER WRITTEN, ORAL, STATUTORY OR OTHERWISE), INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT. CUSTOMER'S SOLE AND EXCLUSIVE REMEDY AND DIALOGIC'S SOLE OBLIGATION HEREUNDER, SHALL BE TO REPAIR, REPLACE, CREDIT OR REFUND AS SET FORTH ABOVE.

G. IN NO EVENT SHALL DIALOGIC, ITS DIRECTORS, OFFICERS, EMPLOYEES, AGENTS OR AFFILIATES, BE LIABLE FOR ANY COSTS OR DAMAGES ARISING DIRECTLY OR INDIRECTLY FROM YOUR USE OF ANY PRODUCT INCLUDING ANY INDIRECT,

INCIDENTAL, SPECIAL, EXEMPLARY, MULTIPLE, PUNITIVE OR CONSEQUENTIAL DAMAGES, INCLUDING LOST PROFITS, WHETHER BASED ON CONTRACT, TORT (INCLUDING NEGLIGENCE), STRICT LIABILITY OR OTHER LEGAL THEORY, EVEN IF DIALOGIC, OR ANY OF ITS DIRECTORS, OFFICERS, EMPLOYEES, AGENTS OR AFFILIATES HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN ANY EVENT, DIALOGIC'S CUMULATIVE LIABILITY TO YOU FOR ANY AND ALL CLAIMS RELATING TO THE USE OF ANY PRODUCT SHALL NOT EXCEED THE TOTAL AMOUNT OF THE PURCHASE PRICE OR LICENSE FEES PAID TO DIALOGIC FOR SUCH PRODUCT.

H. CUSTOMER AND DIALOGIC HEREBY WAIVE THEIR RIGHT TO TRIAL BY JURY TO THE FULLEST EXTENT PERMITTED BY LAW IN CONNECTION WITH ALL CLAIMS ARISING OUT OF OR RELATED TO THIS WARRANTY, THE PRODUCTS COVERED HEREBY OR THE PERFORMANCE OF ANY PARTY HEREUNDER.

I. THIS WARRANTY SHALL BE CONSTRUED UNDER AND GOVERNED BY THE LAWS OF THE COMMONWEALTH OF MASSACHUSETTS WITHOUT GIVING EFFECT TO ANY CHOICE OR CONFLICT OF LAW PROVISION OR RULE (WHETHER OF THE COMMONWEALTH OF MASSACHUSETTS OR ANY OTHER JURISDICTION) THAT WOULD CAUSE THE APPLICATION OF THE LAWS OF ANY JURISDICTION OTHER THAN THE COMMONWEALTH OF MASSACHUSETTS. CUSTOMER SPECIFICALLY AND IRREVOCABLY CONSENTS TO THE PERSONAL AND SUBJECT MATTER JURISDICTION AND VENUE OF THE FEDERAL AND STATE COURTS OF THE COMMONWEALTH OF MASSACHUSETTS AND SUCH COURTS SHALL HAVE EXCLUSIVE JURISDICTION WITH RESPECT TO ALL MATTERS CONCERNING THIS WARRANTY OR THE ENFORCEMENT OF ANY OF THE FOREGOING.

J. THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS. YOU MAY ALSO HAVE OTHER RIGHTS WHICH VARY FROM JURISDICTION TO JURISDICTION.

About this Publication

Purpose

This documentation provides guidelines for using the Dialogic® CSP.

Safety Labels

The following Safety labels may appear in this information product to alert customers to avoidable hazards. The following are in the order of priority:



DANGER

Danger indicates the presence of a hazard that will cause death or severe personal injury if the hazard is not avoided.



WARNING

Warning indicates the presence of a hazard that can cause death or severe personal injury if the hazard is not avoided.



CAUTION

Caution indicates the presence of a hazard that will or can cause minor personal injury or property damage if the hazard is not avoided. Caution can also indicate the possibility of data loss, loss of service, or that an application will fail.

Conventions used

This information product uses the text conventions explained below. In addition, hexadecimal numbers are preceded by a zero and small “x.” For example, the decimal number 15 is represented in hexadecimal as 0x0F.

Convention	Description
...	A horizontal ellipsis in an API message indicates fields of variable length.
:	A vertical ellipsis in an API message indicates that a block of information is repeated or is variable.
<i>n</i>	The letter <i>n</i> is a generic placeholder for a number.
Sans serif mono space	Indicates a command name, option, input, output, non-GUI error, and system messages.
<i>Sans serif monospace italic</i>	Indicates a parameter name in an input message. Example: move *.dot a: c: -s The -s is the parameter.
<i>Serif italic</i>	Indicates the name of a book, chapter, path, file, or API message. Example: <i>UserDirectory/Config.exe</i>
Boldface	Indicates keyboard keys, key combinations, and command buttons Example: Ctrl+Alt+Del
Sans serif boldface	Identifies text that is part of a graphical user interface (GUI). Example: Go to the Configuration menu and select Card->Span Configuration

Contents

Copyright and Legal Disclaimer	2
Dialogic Product Line Warranty	4

1 SwitchKit® Development Environment Introduction

SwitchKit® Product Description	1-2
SwitchKit Features and Components	1-4
LLC - The Low-Level Communicator	1-6
The SwitchManager	1-7
SwitchKit Application Programming Interface	1-8
The Simple Network Management Protocol Agent	1-10

2 Installation

SwitchKit Supported Operating Environments	2-2
Downloading Or Upgrading System Software	2-4
SwitchKit Software Licensing	2-6
Installing SwitchKit on UNIX	2-7
Installing SwitchKit on Windows®	2-10
Running SwitchKit Automatically at Startup on Windows®	2-13
Installing the CSA	2-19
Environment Variables Modify SwitchKit Default Behavior	2-21
Setting Environment Variables on Windows NT® 4.0	2-24
Setting Environment Variables on Windows® XP	2-25
List of Environment Variables	2-27

3	Running SwitchKit Components and Companion Products	
	Running the LLC.....	3-2
	LLC Arguments	3-4
	Running Redundant LLCs.....	3-6
	Basics of SwitchManager	3-8
	Starting SwitchManager	3-9
	Starting SwitchManager with <i>tandem.cfg</i>	3-11
	SwitchManager Arguments	3-13

4	Redundancy Setup	
	SwitchKit Redundancy.....	4-2
	Failure Scenarios Resolved with Redundancy	4-4

5	Building a Configuration File	
	API Messages Used for Configuration.....	5-6
	Configuration Message Syntax.....	5-11

6	Logging and Configuration Updates	
	Log Files.....	6-2
	Log File Management.....	6-6
	Updating Configuration	6-8

7	Device Connectivity	
	Add LLC Node Feature	7-2
	AddLLCNode.....	7-4
	Connecting LLC to an SS7 card	7-7
	AddSS7TCAPCard.....	7-10
	TCAPMessageRegister	7-12
	TCAP Routing Feature.....	7-14
	LLC: Application Load Balancing for TCAP	7-18
	Matrix ConfigureDS Configure : DeviceServerEx	7-22

8	Controlled Line Card Switchover	
	Controlled Line Card Switchover Feature.....	8-2
	SwitchBackFromStandby	8-4
<hr/>		
9	Server Status Change	
	Server Status Change Feature.....	9-2
	ServerStatusChange	9-3
	9-8
<hr/>		
10	Configuration Tutorial	
	Tandem Configuration	10-2
<hr/>		
A	Appendix	
	Frequently Asked Questions	A-2
	Example Configuration File	A-9

1 SwitchKit® Development Environment Introduction

Purpose This chapter is an overview of the SwitchKit® Development Environment and its companion applications.

SwitchKit® Product Description

SwitchKit ® SwitchKit® provides a comprehensive, high-level and open programming environment for the application development and maintenance of the CSP. SwitchKit includes a feature-rich operations, administration and maintenance (OA&M) system and a high-level API suite, freeing developers to concentrate on revenue-generating applications and services. Because it facilitates the development and integration of Dialogic switch-based telephony applications, it delivers important benefits to both system integrators and service providers.

SwitchKit greatly reduces the time and cost in the deploying and maintaining switch-based solutions - providing a quantitative edge in markets that are increasingly competitive, dynamic, global, and deregulated. SwitchKit enables the development and implementation of custom, highly differentiated services, providing a qualitative edge in response to the demands of the telecommunications marketplace.

SwitchKit is implemented in a modular manner. It segregates administration and configuration functions from the application, enabling multiple services to utilize a single, consistent OA&M system. Since an application developed using SwitchKit does not need to concern itself with OA&M overhead, it can be simpler, modular, more efficient, and easily maintained. Furthermore, applications are portable and scalable because they operate independently of the switch utilities. The result is that developers can implement applications and services with better price performance and superior flexibility over a wide range of market needs and sizes.

SwitchKit supports resource sharing and redundancy. It lets multiple independent applications efficiently share a single switch by handling the channel sharing and message routing automatically. Resources such as cards, lines and nodes can be added to the environment while an application is running, and configured automatically. Thus, multiple solutions and facilities can be developed and delivered through a single, open and programmable switch platform — Dialogic's platform — for superior return on investment and reliability in all aspects of the services creation and provisioning process.

SwitchKit has been developed based on the C++ programming language and industry-standard communications protocols, and is available on a variety of platforms. It is designed to be easily scalable

and upgradable, so that modules can be added and or extended, enabling easy growth of existing services and addition of new ones - without redesign.

SwitchKit features include:

- Modules that administer, configure, and automatically manage the operation of switches.
- Redundant low-level communicators that eliminate any single point of failure.
- A C and C++ API suite for the CSP, based on an industry-standard call processing model.
- Converged Services Administrator, a graphical user interface for configuring, monitoring and maintaining the CSP.

SwitchKit has been designed to support the performance of CSP hardware. As a result, applications can make better and more efficient use of Dialogic resources. Running on either Windows® or UNIX platforms, SwitchKit provides a comprehensive suite of tools.

Application Checklist

SwitchKit provides the following core functional areas needed for your application to work effectively with the Multi-Signaling Platform.

- Host Connection
- Communications
- Message Management
- Configuration
- Alarm Manager
- Real-time Logging
- Real-time Management
- User Interface

After all the above functional areas have been addressed in your application, you may now add call processing to your application.

SwitchKit Features and Components

Description SwitchKit is a software package that consists primarily of five modular components that function on an external host computer. SwitchKit acts as an application server, communicating with all applications and the CSP through a Transmission Control Protocol/Internet Protocol (TCP/IP) interface.

SwitchKit also provides a comprehensive, high-level and open programming environment for the application development and maintenance of the CSP. SwitchKit includes a feature-rich OA&M (operations, administration, and maintenance) system and a high-level API suite, freeing developers to concentrate on revenue-generating applications and services. Because it facilitates the development and integration of Dialogic switch-based telephony applications, it delivers important benefits to both system integrators and service providers.

In supporting multi-node CSP systems, SwitchKit connects to each node.

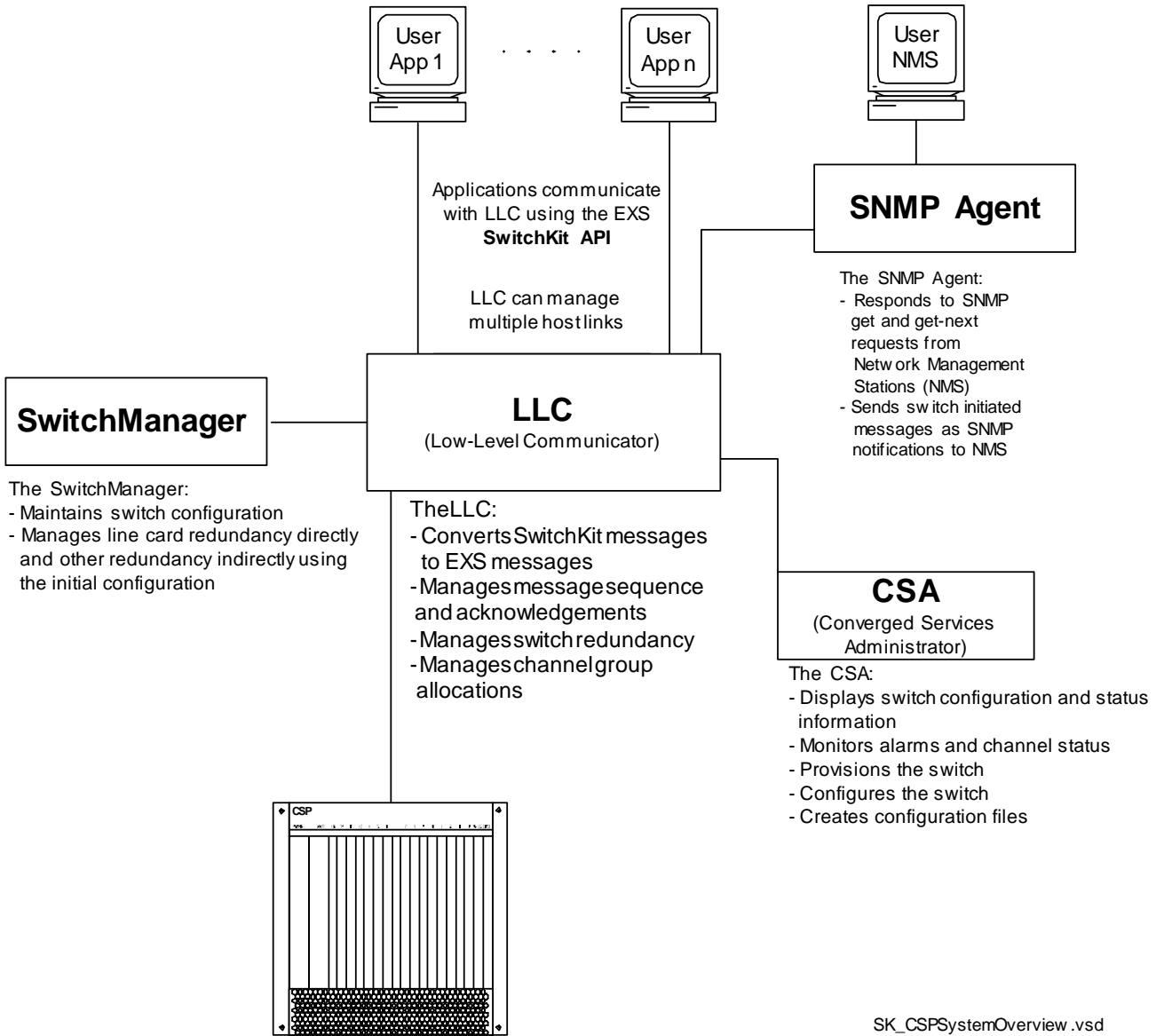
Main Components SwitchKit contains the following five components:

- Low-Level Communicator (LLC)
- SwitchManager
- Application Programming Interface (API)
- Converged Services Administrator (CSA)
- Simple Network Management Protocol (SNMP) Agent

For more information on the LLC and SwitchManager see the *SwitchKit Introduction and Installation Guide* and for SwitchKit API information see the *API Reference*.

SwitchKit Components Diagram

The following diagram provides an overview of the SwitchKit components. All components communicate using TCP/IP.



SK_CSPSystemOverview.vsd

LLC - The Low-Level Communicator

The LLC The Low-Level Communicator (LLC) coordinates and intelligently routes all communication, for example messages, between applications and the switch. It is the only module that communicates directly with the switch over an Ethernet connection, and that can distribute and collect messages to and from all the application modules.

The LLC coordinates many low-level communication tasks that otherwise would be the application's responsibility, such as:

- Message formatting, queuing, sequencing and acknowledgment
- Calculating checksums
- Handling special characters
- Message and error logging
- Intelligent channel allocation
- Socket management
- Connections to multiple switches

Because the LLC serves as a layer above the platform, it can handle multiple application modules running on any machine in the network. Messages can be routed intelligently among all of the application modules. When a message arrives on a channel, LLC routes that message to whatever application is responsible for that channel. By using this mechanism, two independent applications can share a single CSP. Also, you can structure an application system to contain multiple application modules, to provide redundancy, load sharing, and separation of functionality.

The SwitchManager

The SwitchManager The SwitchManager is a process that helps maintain the CSP configuration environments in a state of readiness. It implements switch and switch resource configuration with a user-defined configuration file. It reads the configuration files and sends configuration messages to the switch, and supports automatic query of the switch to determine its current hardware setup and configuration. Configuration files are comprised of easily-used text names and fields. The switch can also be dynamically reconfigured. For example, even while running traffic, SwitchManager can reconfigure the platform automatically if the switch or any of its individual cards lose configuration, such as, when a card is pulled.

You can use the SwitchManager to configure resources — cards, spans, DSPs and channels — by providing default values. You can also define inseize and outseize rules, call progress analysis parameters, hierarchical groups of channels, custom tones, and filters and timers. PPL tables can be specified so that they are downloaded automatically. Voice Recorded Announcement System (VRAS) prompts can be downloaded dynamically. SwitchManager enhances reliability with extensive fault detection and notification features. It monitors alarms, tracks the CSP Matrix Series 3 heartbeat and switchover recovery, and automates N+1 line card redundancy.

SwitchManager works in conjunction with the Converged Services Administrator (CSA) to provide graphical configuration and provision in the CSP.

SwitchKit Application Programming Interface

Description The SwitchKit Application Programming Interface (API) is a high-level interface between the application and the LLC that facilitates rapid switch-to-application integration. The API extracts the EXS API suite in C/C++ Library format. This interface enables the application developer to support EXS messaging as well as administrative messages (between the application and the low level communicator) with all the benefits of a high-level language API.

In short, SwitchKit lets developers address the call processing aspects of their solutions as they see fit — giving them the power to implement new services easily, quickly, and profitably.

For the UNIX operating environments, SwitchKit API is available in two sets of libraries: standard SK API and threadsafe SK API. Both APIs offer the same basic functionality to the application developer, with one major difference. The standard SK API allows single-threaded applications to be developed to connect to the LLC. The threadsafe SK API allows multi-threaded applications to be developed to connect to LLC. For maximum flexibility in application functionality today and in the future, it is recommended that all new development occur on the threadsafe SK API.

CSA - Converged Services Administrator

Description The Converged Services Administrator (CSA) is a real-time switch configuration, monitoring, and administration application. The CSA lets developers monitor the current state of the switch in real-time through an intuitive graphical user interface. You can even monitor the switch remotely over multiple terminals. The state of the switch and all its components and alarms are graphically displayed as they occur. The view of the switch can be easily changed at any time.

You can also monitor trunk groups and trace and view channel status in real-time. You can take channels and trunk groups in and out of service without downtime. You can view alarm and statistic details without decoding log files.

For additional information on CSA, see the *Converged Services Administrator User's Guide*.

CSA offers the ability to configure a CSP off-line, without any SwitchKit components or a Converged Services Platform available. CSA also offers the ability to configure a CSP on-line, from scratch or by modifying the existing application.

The Simple Network Management Protocol Agent

Description The Simple Network Management Protocol (SNMP) Agent allows system traps and alarms to be sent to a remote network management application. This protocol also allows queries to be initiated from a remote application. For information on the SNMP Agent, see the *Simple Network Management Protocol User's Manual*.

2 Installation

Purpose This chapter guides you through the installation process of SwitchKit.

SwitchKit Supported Operating Environments

Purpose This information provides the Operating Environments supported on SwitchKit and its companion products with this release.

SwitchKit Run-time Environment The next table shows the operating systems and versions supported by SwitchKit in a run-time environment.

Table 2-1 Operating System Requirements

Operating System	SwitchKit is supported on...
Solaris	Version 10.0 for the SPARC platform
Linux	Red Hat Enterprise Linux ES, version 3.0
HP-UX	Version 11.0
Windows NT®	Version 4.0, service pack 5 or greater
Windows® XP	Windows/XP service pack 2

SwitchKit Development Environment

The next table shows the compilers supported by SwitchKit in a development environment.

Table 2-2 Compiler Requirements

Operating System	SwitchKit applications must be built using...
Solaris Sparc	GNU gcc 3.4.3 Sun WorkShop 6 Update 2 C++ 5.3 Patch 111685-18 20003/09/24
Solaris x86	GNU gcc 3.4.3
Linux	RedHat ES 3 w/ gcc 3.2.3 20030502
HP-UX	ACC 3.37 C++ compiler Two versions of SwitchKit API are provided, one to support applications using the <code>-AA</code> flag, and one to support applications not using the <code>-AA</code> flag. If you are using the <code>-AA</code> compiler flag, the following patch must be installed: For HP-UX 11.0, install the patch PHSS_26945 1.0 HP aC++ -AA runtime libraries (aCC A.03.37).
Windows NT®	MS Visual Studio C++® Version 6.0 compilers
Windows® XP	MS Visual Studio C++® Version 6.0 compilers
Windows® 2000	

CSA The Converged Services Administrator (CSA) works in conjunction with SwitchKit. The CSA is available on Windows NT® and Windows® XP.

Run CSA from a Windows®-based machine in conjunction with SwitchKit on any of the supported operating environments detailed above by connecting via TCP/IP.

Switch System Software

SwitchKit Release 8.4 CI is compatible with a CSP running Switch System Software Release 8.4 CI.

Downloading Or Upgrading System Software

Purpose This procedure explains the process for downloading system software to the CSP either as an upgrade to existing software or installing the software for the first time. The CSP system software must be installed before you can start the SwitchKit components: Low-Level Communicator (LLC) or SwitchManager.

Before you begin BOOTP and TFTP servers **must** be configured and their services started. If you have redundant hosts you can have BOOTP and TFTP running on both hosts.

Upgrading System Software Download The following step is recommended for upgrading the System Software using SwitchKit.

1 Send Clear System Software and Reset Matrix through CSA to both CSP Matrix Series 3 cards.

2 In CSA select the node you want to clear the System Software on.

3 Go to the menu bar, select **Provisioning**→**Node**→ **Clear System Software**. Power cycle the CSP.

END OF STEPS

System Software Download for a new system The following procedure is recommended for installing the System Software using SwitchKit on a new CSP that has no system software.

1 Power ON the switch. The BOOTP request is sent automatically and finds an IP address for the switch. The BOOTP response must contain the TFTP configuration settings. The matrix then requests the TFTP software loads.

-
- 2** Start SwitchKit. If TFTP is still in progress while the matrix is getting binaries, LLC reports the TFTP download is still in progress. LLC waits for the matrix to become active.

END OF STEPS

Full TFTP download process

- LLC clears the system software on both CSP Matrix Series 3 cards and then waits until download is complete.
- During a TFTP download, LLC displays “TFTP Download in progress” in the LLC maintenance window/log file. This process could take up to several minutes.
- In a redundant scenario, after the CSP Matrix Series 3 card to CSP Matrix Series 3 Card download is complete, the CSP Matrix Series 3 card(s) reset. One CSP Matrix Series 3 card becomes active and the other standby.

SwitchKit Software Licensing

Purpose This procedure explains the SwitchKit software licensing.

Before you begin Locate your license file. The license file is included on a floppy disk that is shipped with the Switchkit CD. The file is likely named: “SK_YOUR_COMPANY_NAME.dat”.

Licenses Licenses are available for single-node and multi-node CSP configurations. Both types of licenses are matched to the chassis IDs of each node in the CSP. SwitchKit applications can connect to any LLC that is running with a valid software license. Independent software licenses are not required for SwitchKit applications.

Steps for Licensing Follow this procedure to license your product.

1 Copy your license file into the base of your installation directory. By default this is for

- Windows: *C:\Program Files\Excel Switching Corporation\SwitchKit*
 - UNIX: */usr/local/switchkit*
-

2 Rename the license file to “sk_license.dat”.

END OF STEPS

Note Demonstration and Beta releases of SwitchKit can be used with an expiring license. Do not use an expiring license on a production system. To determine if a license is an expiring license, view the license file with a text editor. If you see the term “Expires:” in your license file, please call your Dialogic Sales Representative and request a permanent license file.

Installing SwitchKit on UNIX

Purpose This procedure describes the installation of SwitchKit on a UNIX system.

Before you begin If you are installing SwitchKit in a directory other than the default, replace all path names in this procedure with the appropriate path. All file names are case-sensitive and should be in lower case in UNIX. All linkable libraries in SwitchKit are in ELF format.

Installing SwitchKit on UNIX Follow this procedure to install SwitchKit on UNIX systems.

1 Log on to the target machine:

```
login:<user name>  
password:<your password>
```

2 Insert the SwitchKit installation CD.

3 Create the following directory if it does not already exist:

```
mkdir /usr/local
```

4 Create the SwitchKit installation directory by typing:

```
mkdir /usr/local/switchkit
```

5 Copy the SwitchKit installation script and zip file reside on the CD in the directory /SwKit/<UNIX Operating System>. The name of the SwitchKit installation zip file is as follows:

```
Solaris: solsparc.bin.gz
```

```
Linux: linux.bin.gz
```

```
HP/UX: hpux.bin.gz
```

Copy the appropriate SwitchKit installation file and zip file to the installation directory:

```
cp /SwKit/<UNIX Operating System>/*.bin.gz
cp /SwKit/<UNIX Operating System>/install.sh
chmod +x install.sh
```

- 6** Type the following exactly:

```
./install.sh
```

- 7** Follow the instructions on the screen to complete the installation.
-

- 8** This step is optional. Create a directory under */usr/local/switchkit* called “**log**” by typing:

```
mkdir /usr/local/switchkit/log
```

Log files by default are stored in the same directory that LLC and SwitchManager are run from.

- 9** This step is optional. Create a directory under */usr/local/switchkit* called “**cfg**” by typing:

```
mkdir /usr/local/switchkit/cfg
```

You can store config files in any directory that SwitchManager can access. If the configuration file is in a separate directory that SwitchManager cannot access, the config file will need to specify the path (absolute or relative to where SwitchManager is).

- 10** Add entries to the */etc/profile* file

This allows the system to automatically set the SwitchKit environment variables every time you log in. To do this, add the following lines to the */etc/profile* file (somewhere near the top).

```
SK_LIB_DIR=/usr/local/switchkit
SK_LOG_DIR=/usr/local/switchkit/log
```

```
export SK_LIB_DIR SK_LOG_DIR
END OF STEPS
```

Directory Summary The directories listed in the following table are required for operation. They are either created automatically during installation or must be created by the user, as noted.

Directory	Contents and Use	Created
/usr/local/switchkit/bin	SwitchKit executable files	Automatically
/usr/local/switchkit/lib	SwitchKit development libraries	Automatically
/usr/local/switchkit/include	SwitchKit development header files	Automatically
/usr/local/switchkit/samples	Contains sample configuration files	Automatically
/usr/local/switchkit/log	Stores log files	By User
/usr/local/switchkit/cfg	Stores configuration files	By User

Installing SwitchKit on Windows®

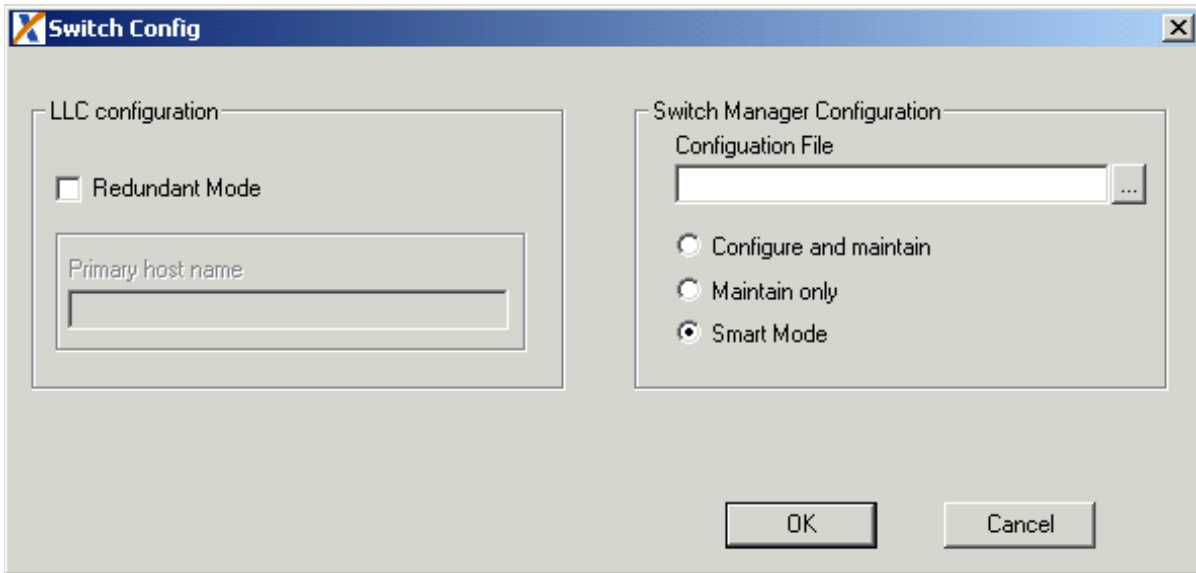
- Purpose** This procedure guides you through the SwitchKit installation.
- Default directory** The SwitchKit default installation directory is *C:\Program Files\Excel Switching Corporation\SwitchKit* and for the CSA to *C:\Program Files\Excel Switching Corporation\CSA*.
- Before you begin** Other installations of SwitchKit components must not be running on the system when reinstalling or installing a new version. LLC and/or SwitchManager must be stopped if they are running as automatic services at startup.
- If you previously saved customized files in your installation directory, move the files to a temporary location of your choice. After you have completed your SwitchKit installation you need to move the files to your installation directory, by default *C:\Program Files\Excel Switching Corporation\SwitchKit*.
- Installing SwitchKit** Follow the steps below to install SwitchKit on Windows® systems. If you are installing SwitchKit in a directory other than the default, replace all path names in this procedure with the appropriate path.
-
- 1** Log on using an administrative account. If you are installing SwitchKit on a system for the first time, start with *Step 3*.

 - 2** Remove your current SwitchKit installation from your system. Use the **Add→Remove Programs** feature under the Windows® Control Panel to remove the program.

 - 3** Insert the SwitchKit Installation CD.

 - 4** Double-click *skit.exe* located in the *SwitchKit/WindowsNT* folder on the CD. This is a self-extracting executable. By default, all associated files are placed in the directory *C:\Program Files\Excel Switching Corporation\SwitchKit*.

-
- 5 While you are installing SwitchKit, the switch configuration dialog box is invoked. See the next screen shot.



-
- 6 This dialog box allows you to set up a redundant LLC and a SwitchManager configuration file when SwitchKit components are to be run automatically as Windows® services. Select **Redundant Mode** under **LLC Configuration** and enter the **Primary host name**. Use the browse button to set up the SwitchManager **Configuration File**. Click **OK**.

If you do not intend to run these as Windows® services, click **Cancel**. For further information, see the *Running SwitchKit Automatically at Startup on Windows® (2-13)*.

NOTE: By default, the LLC and SwitchManager are set up in Windows Services to be started manually.

-
- 7 Follow the instructions on the screen to complete the installation.

-
- 8 Reboot your machine if prompted.

END OF STEPS

SwitchKit (SwitchManager and LLC) is now installed.

LoadLibrary Error while installing SwitchKit

With certain DLLs a RegSvr32 error can occur during installation, stating LoadLibrary (“GridWiz.ocx”) failed. See the following figure.



If after clicking **OK**, the Switch Config window opens, you can ignore this error message. If the **Switch Config** window doesn't open, reboot the computer and reinstall SwitchKit.

If you are not running SwitchKit as a service, ignore this error message.

Running SwitchKit Automatically at Startup on Windows®

Purpose This procedure describes how to make the Low-Level Communicator (LLC) and SwitchManager run automatically as services on a Windows® computer at startup. By default, when you install SwitchKit, the LLC and SwitchManager are installed as services but must be started manually.

To ensure maximum system up-time in production environments, we recommend setting up both the LLC and SwitchManager to run automatically as Windows® Services.

Before you begin You must have SwitchKit installed.



CAUTION

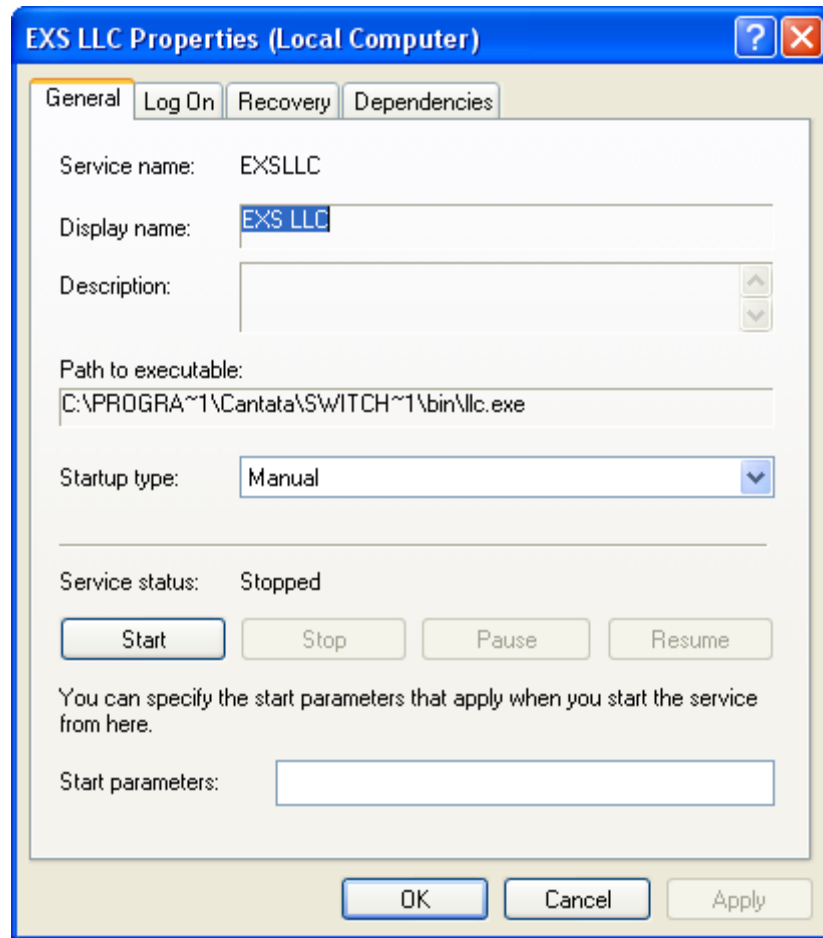
Auto-start is not suitable for a development environment where resources are shared. Configuring LLC and SwitchManager as auto-start services is recommended only if your host computer will have continuous connectivity to the switch. If a host computer does not have full time access to a switch, configuring LLC and SwitchManager as auto-start services may cause problems during initialization. Also, if a host computer shares access to a switch, this could cause thrashing between multiple LLCs and their connections to the switch.

Enabling SwitchKit to Run as a Service

To make LLC and SwitchManager run as automatic Services at startup do the following. If you are using Windows NT® start at *Step 4*:

For Windows XP

- 1 From the **Start** button, go to **Settings→Control Panel→Administrative Tools→Services**.
 - 2 Select the service **EXS LLC** and right-click **Properties** from the menu. The **EXS LLC Properties (Local Computer)** dialog box opens. See the next screen shot.
-



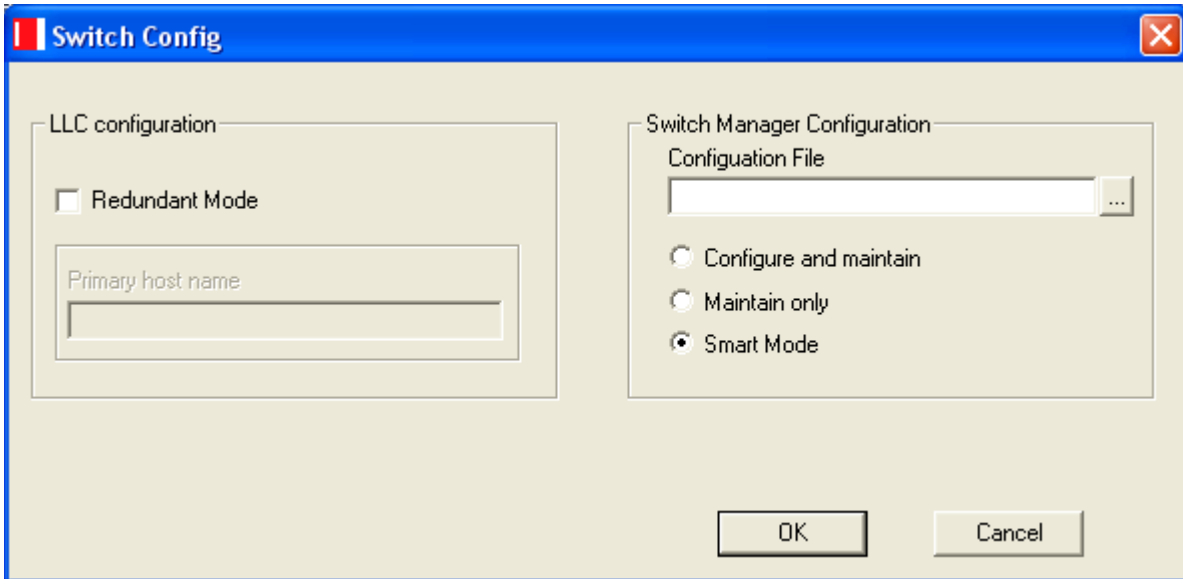
-
- 3 Go to the **General Tab**, **Startup type** field and select **Automatic** from the drop-down list. Click **Apply**, then **OK**. Repeat the previous steps for SwitchManager. Then, go to *Step 7*.

For Windows NT

-
- 4 From the **Start** button, go to **Settings**→**Control Panel**→**Services**.
 - 5 Select the service **EXS LLC** and click the **Startup** button.
 - 6 Select **Automatic** under **Startup type**. Click **OK**. Repeat the previous steps for SwitchManager. Then, go to *Step 7*.

-
- 7 From the **Start** menu, select **Programs**→**SwitchKit**→**SwitchConfig**. The **Switch Config** dialog box opens.

See the next screen shot.



-
- 8 Specify the LLC Configuration
- If the host is going to run as the redundant controller, check the box for **Redundant Mode** to enable redundancy and provide the Primary host name.

-
- 9 Specify the SwitchManager Configuration
- In the **Configuration File** text box, enter the full path to the configuration file.

-
- 10 Select the connection method by selecting one of the following SwitchManager options (For more information about this options see *Starting SwitchManager with tandem.cfg (3-11)*):

- **Configure and maintain**
Each time it connects to the LLC, SwitchManager sends the entire configuration file to the switch. This results in a complete system configuration.

- **Maintain Only**

Each time an initial SwitchManager connects to the LLC, SwitchManager reads the configuration file and does no configuration.

- **Smart Mode**

Each time it connects to the LLC, SwitchManager reads a configuration file and reconfigures anything that is configured differently than specified in the configuration file. SwitchManager determines this by checking the Configuration Tag of all cards in the system. Any card with a Configuration Tag of zero (0) is reconfigured.

Important! This option only applies to the initial startup of the SwitchManager. A system running in a production environment should be run in Smart Mode.

11 Click **OK** to save your settings and exit.

12 Invoke the **Control Panel** by select **Start/Settings→Control Panel**.

13 On Windows NT® systems double-click the **Services** icon.

On Windows® XP systems double-click the **Administrative Tools** icon and then double-click the **Services** icon.

This opens the services dialog box.

14 Select “EXS LLC,” change the StartUp type to Automatic, and click **OK**.

15 Select “EXS SwitchMgr,” change the StartUp type to Automatic, and click **OK**.

16 Apply the changes and close the **Control Panel**.

END OF STEPS

SwitchKit Tray Icons on Windows®

- Description** When running SwitchKit on Windows NT® as a service, tray icons give a visual indication of the LLC and SwitchManager status within the icon tray bar.
- LLC Icons** The icon for the LLC shows an "LLC" at the bottom. The "LLC" icon is complemented with one of the following:
- a green A above the LLC indicates the LLC on this machine is the primary LLC.
 - a yellow S above the LLC indicates the LLC on this machine is the redundant LLC.
 - a red I above the LLC indicates the LLC on this machine is initializing.
- SwitchManager Icons** The icon for the SwitchManager shows an "SM" at the top. The "SM" icon is complemented with one of the following:
- a transparent bar under the SM indicates that SwitchManager is idle.
 - a yellow bar under the SM indicates that SwitchManager on this machine is the redundant SwitchManager.
 - a red bar under the SM indicates that SwitchManager is currently handling a message.
 - a red ! over the SM indicates that SwitchManager is unable to talk to the LLC.
- Viewing SwitchKit log files** You can view log files using the default file viewer (Notepad) or with different viewer. To override the default file viewer, set the environment variable `SK_VIEW_SPEC` or add `VIEW_SPEC` to the defaults file. Use the full path to the viewer. The "%s" MUST be enclosed in quotes("") so that file paths with spaces works.
- LLC Logs** To open the messages.log file double-click the LLC icon. If you right-click the LLC icon, the following options are presented:
- View messages.log
 - View messages.log with Notepad
 - View maintenance.log
 - View maintenance.log with Notepad

SwitchManager Logs To open the alarm.log file double-click on the SM icon. If you right-click the SM icon, the following options are presented:

- View alarm.log
- View alarm.log with Notepad

Installing the CSA

Purpose This procedure guides you through the Converged Services Administrator installation. CSA can be installed before or after you install your CSP system software. You can create configuration files offline using CSA. See the *Converged Services Administrator User's Guide* for more details.

Before you begin You should run the CSA on a separate computer from other SwitchKit modules and call control applications. Dialogic recommends separating critical real-time processes from graphical display environments.

Important! It might be necessary to reboot your machine after the installation.

Installing the CSA Follow the steps below to install the CSA.

- 1 Log on to your target machine with an administrative account.

- 2 If you have the SwitchSight Administrator (SSA) on your system, you should remove it from your system. Use the **Add→Remove Programs** feature in the Windows **Control Panel** to remove the program.

- 3 When the SSA or older version of CSA is successfully uninstalled, revisit your installation directory and remove the SSA folder.

- 4 Insert the *SwitchKit Installation CD*.

- 5 Go to the folder: *SwitchKit/User Interface*.

- 6 Double-click on *CSA.exe* to install the program. This is a self-extracting executable. By default, all associated files will be installed in the directory *C:\Program Files\Excel Switching Corporation\CSA.exe*.

7 Follow the screen instructions.

8 Reboot your machine if prompted.

END OF STEPS

Environment Variables Modify SwitchKit Default Behavior

Overview You can modify SwitchKit component behavior through environment variables or by changing the values in the SwitchKit *Defaults* file. Environment variables take precedence over the variables in the *Defaults* file.

Modifying Default Behavior on Windows®

In a Windows® environment, the SwitchKit system environment variable *SK_LIB_DIR* is automatically set. If you have *SK_LIB_DIR* set in your user environment variables, please remove it.

Important! When re-installing SwitchKit on Windows NT®, *SK_LIB_DIR* gets overwritten.

SwitchKit environment variables can be set in three ways on Windows®:

- Use the **Command Line**. Here is an example of the syntax:
`set SK_ENV_VAR=x`

Important! Environment variables set on the command line only apply to the Switchkit process (LLC, SwitchManager or a user application) running in that command shell.

- Use the **Control Panel** in Windows®. Please refer to the procedures *Setting Environment Variables on Windows NT® 4.0 (2-24)* or *Setting Environment Variables on Windows® XP (2-25)*.
- Use the defaults file. You need to create this in the following directory:
SK_LIB_DIR.

Modifying Default Behavior on UNIX

SwitchKit environment variables can be set in three ways on UNIX:

- Use the **@ Shell Prompt**. Here is an example of the syntax:
`SK_ENV_VAR=x`
`export SK_ENV_VAR`

Important! Environment variables set on the **@ Shell Prompt** only apply to the Switchkit process (LLC, SwitchManager or a user application) running in that command shell.

- Use **.Profile** for user.

- Use the *defaults* file. You need to create this in the SK_LIB_DIR directory.

Variable Rules These are the rules regarding environment variables and defaults files.

- Environment Variable names are case-sensitive and must be entered as all capital letters. For example, SK_VARIABLE.
- The Defaults file name is case-sensitive and must be entered exactly as shown below for each operating environment.

/SK_LIB_DIR

- All *Defaults* file variable names are entered in lower case and do not use the prefix, sk.
- In the *Defaults* file, if the first character in a line is a pound symbol (#), subsequent text is considered a comment.
- Blank lines are allowed.

Format The format for default values in both UNIX and Windows® is as follows:

<fieldname> : value

Example LLC_Host : localhost
LLC_Port : 1312
RLLC_Host : localhost
RLLC_Port : 1312

Logging Environment Variables In maintenance.llc.log, LLC will log all environment variables used, where that variable was set and what the value is. The next example shows details logged by LLC:

```
Jan 11 2005 11:48:45: Variables read from environment:
Jan 11 2005 11:48:45: Environment var SK_APP_DISABLED_TIMEOUT -> "99999"
Jan 11 2005 11:48:45: Environment var SK_FILE_LIFETIME -> "1"
Jan 11 2005 11:48:45: Environment var SK_LIB_DIR -> "C:\Program Files\Excel Switching Corporation\SwitchKit"
Jan 11 2005 11:48:45: Environment var SK_LOG_DIR -> "C:\Program Files\Excel Switching Corporation\Log"
Jan 11 2005 11:48:45:
Jan 11 2005 11:48:45: Variables read from file "C:\Program Files\Excel Switching Corporation\SwitchKit\Defaults":
Jan 11 2005 11:48:45: In defaults file, key of SK_DISABLE_CSM has value "0"
Jan 11 2005 11:48:45: In defaults file, key of SK_LLC_HOST has value "135.119.52.4"
Jan 11 2005 11:48:45: In defaults file, key of SK_LLC_PORT has value "1312"
Jan 11 2005 11:48:45: In defaults file, key of SK_LOG_LEVEL has value "6"
Jan 11 2005 11:48:45:
Jan 11 2005 11:48:45: Variables set at runtime
Jan 11 2005 11:48:45:
Jan 11 2005 11:48:45: Blanked Variables
```

```
Jan 11 2005 11:48:45:  
Jan 11 2005 11:48:45: Variables as seen at runtime  
Jan 11 2005 11:48:45: key of SK_APP_DISABLED_TIMEOUT has value "99999"  
Jan 11 2005 11:48:45: key of SK_DISABLE_CSM has value "0"  
Jan 11 2005 11:48:45: key of SK_FILE_LIFETIME has value "1"  
Jan 11 2005 11:48:45: key of SK_LIB_DIR has value "C:\Program Files\Excel Switching  
Corporation\SwitchKit"  
Jan 11 2005 11:48:45: key of SK_LLC_HOST has value "135.119.52.4"  
Jan 11 2005 11:48:45: key of SK_LLC_PORT has value "1312"  
Jan 11 2005 11:48:45: key of SK_LOG_DIR has value "C:\Program Files\Excel Switching  
Corporation\Log"  
Jan 11 2005 11:48:45: key of SK_LOG_LEVEL has value "6"
```

Setting Environment Variables on Windows NT® 4.0

Purpose This procedure explains how to set an environment variable using the **System Variables** option on Windows NT®. The environment variable, *SK_LOG_DIR*, is used as an example in this procedure.

Before you begin Create the directory to store the log files, *C:\Program Files\Excel Switching Corporation\SwitchKit\Log*. If you are updating or reinstalling SwitchKit this directory is already there. The log files are written into the directory based on the Environment Variable *SK_LOG_DIR*.

Setting SK_LOG_DIR Follow the steps below to set the *SK_LOG_DIR* Environment Variable.



CAUTION

Do not define environment variables as User variables. You need to set environment variables under System variables to run SwitchKit successfully as Windows NT Services.

- 1 Log on using an administrative account.

- 2 To change or add Environment Variables, go to **Control Panel**→**System/Environment**→**System Variables**. If the SwitchKit Environment Variables are defined in the User Variables panel, delete them from this location.

- 3 Go to the **System Variable** field and type *SK_LOG_DIR*.

- 4 Enter into the **Value** field *C:\Program Files\Excel Switching Corporation\SwitchKit\Log*.

- 5 Click the **Set** button and close all open windows by clicking **OK**.

END OF STEPS

Setting Environment Variables on Windows® XP

Purpose This procedure explains how to set an environment variable using the **System Variables** option on Windows® XP. The environment variable, *SK_LOG_DIR*, is used as an example in this procedure.

Before you begin Create the directory to store the log files, *C:\Program Files\Excel Switching Corporation\SwitchKit\Log*. If you are updating or reinstalling SwitchKit this directory is already there. The log files are written into the directory based on the Environment Variable *SK_LOG_DIR*.

Setting SK_LOG_DIR Follow this procedure to set the *SK_LOG_DIR* Environment Variable.

If you are updating or reinstalling your SwitchKit installation, this environment variable should already be set on your system.



CAUTION

Do not define environment variables as User variables. You need to set environment variables in the Control Panel → System variables to run SwitchKit successfully as Windows NT Services.

-
- 1** Log on using an administrative account.

 - 2** Go to **Control Panel** → **System**.

 - 3** Select the tab **Advanced**.

 - 4** Select the button **Environment Variables**.

 - 5** If this is your initial setup, click **New** below the **System variables** field and continue with Step 7. Otherwise, refer to Step 6.

-
- 6 If you want to change the variable settings, click **Edit** below the **System variables** field.

 - 7 In the field **Variable Name** type *SK_LOG_DIR*.

 - 8 In the field Variable Value type *C:\Program Files\Excel Switching Corporation\SwitchKit\Log*.

 - 9 Click **OK** to close the **New System Variable** or **Edit System Variable** dialog box.

 - 10 Click **OK** to close the **Environment Variables** dialog box.

 - 11 Click **OK** to close the **System Properties** dialog box.

 - 12 Then close the **Control Panel**.

END OF STEPS

List of Environment Variables

Overview

This section provides the names of all of the environment variables available with SwitchKit. The environment variables are listed in alphabetical order and each is described indicating what it is used for. The variable name in the *Defaults* file is provided, along with the valid values for each variable.

Important! Environment variables should be set within the value ranges specified for each variable listed. Setting a variable outside this range will result in unexpected behavior.

The environment variables and the *Defaults* file are read at start-up by LLC and SwitchKit applications. To modify settings at run-time, you can use CSA. See the *CSA User's Guide*.

SK_APP_ DISABLED_TIMEOUT

SK_APP_DISABLED_TIMEOUT specifies the number of seconds that an application must be silent and fail to respond to pings before the LLC considers it unavailable. This value applies to all applications connected to the LLC. You can dynamically change this environment variable using the SwitchKit graphical user interface, CSA. See the *Converged Services Administrator User's Guide*, "Changing Environment Variables at Run-Time."

Defaults file variable name

app_disabled_timeout

Valid Values

SK_APP_DISABLE_TIMEOUT = 15

Disconnect applications after 15 seconds (default)

SK_APP_DISABLE_TIMEOUT = 1 to 65535

Valid range for timeout

SK_AUTO_RETURN_ CHANNELS

SK_AUTO_RETURN_CHANNELS specifies whether a ChannelReleased message from the switch should trigger an automatic sk_returnChannel. The use of SK_AUTO_RETURN_CHANNELS is recommended because it simplifies application development. It also prevents a possible race condition, if an *XL_ChannelReleased* message is immediately followed by an *XL_RequestForService*, before the application has been able to return the channel. It defaults to 0 for backward compatibility.

Defaults file variable name

auto_return_channels

Valid Values

SK_AUTO_RETURN_CHANNELS = 0

- Disable (default)

SK_AUTO_RETURN_CHANNELS = 1

- Enable

**SK_BACKUP_WHEN
_CLEARLOG**

SK_BACKUP_WHEN_CLEARLOG controls whether a backup of log files will be saved when the clear log command is sent to the LLC.

Defaults file variable name

backup_when_clearlog

Valid Values

SK_BACKUP_WHEN_CLEARLOG = 0

- Disabled

SK_BACKUP_WHEN_CLEARLOG = 1

- Enabled (default)

**SK_CHANNEL_RECOVERY
_METHOD**

SK_CHANNEL_RECOVERY_METHOD allows LLC control of the channel in case of an abrupt disconnect from an application that is controlling the channel.

Defaults file variable name

channel_recovery_method

Valid Values

SK_CHANNEL_RECOVERY_METHOD = 0

- OFF (default) channels owned by the application are left in their current state

SK_CHANNEL_RECOVERY_METHOD = 1

- Release the non-idle channels owned by the application

SK_CHANNEL_RECOVERY_METHOD = 2

- Take non-idle channels owned by the application out of service

SK_CHANNEL_RECOVERY_METHOD = 3

- Take all channels out-of-service within the watched channel group(s)

Note:

Methods 1 and 2 only affect non-idle channels (channels that are in active calls).

For Method 2 and 3, after you restart the application to watch the channel groups, you must bring the channels back in-service using the *ForceGroupState* message.

Method 3, will affect channels:

- in an entire group being watched using the `sk_watchChannelGroup()` function;
- that are not watched by any other applications;
- that are currently active only after they are released prior to being brought out-of-service.

SK_CSA_INSTALL_DIR

SK_CSA_INSTALL_DIR is set when CSA is installed. This environment variable is used by CSA to store the files it sends to SwitchManager. The default path for the CSA installation is C:\Program Files\Cantata\CSA.

Important! Note: If this environment is not set then CSA will crash.

Defaults file variable name

csa_install_dir

Valid Values

SK_CSA_INSTALL_DIR = YourInstallationFolder

Example

SK_CSA_INSTALL_DIR = C:\Program Files\Cantata\CSA

SK_DEBUG_CHANNEL_MANAGER

Important! SK_DEBUG_CHANNEL_MANAGER should only be used after consulting with a Dialogic Technical Support representative.

Use this environment variable to enable logging of debug messages from the LLC channel manager. These messages will be logged in the messages.log.

SK_DEBUG_CHANNEL_MANAGER needs to be set in conjunction with SK_ENABLE_CHANNEL_OWNERSHIP_LOGGING. For more information on channel state changes, see the description of *SK_ENABLE_CHANNEL_OWNERSHIP_LOGGING (2-32)*.

Defaults file variable name

debug_channel_manager

Valid Values

SK_DEBUG_CHANNEL_MANAGER = 1

- Enables channel manager debug logging

SK_DEBUG_CHANNEL_MANAGER = 0

- Disables channel manager debug logging (default)

SK_DISABLE_CSM

SK_DISABLE_CSM enables or disables LLC from sending the *SK_ConnectionStatusMsg* to all connected applications.

Defaults file variable name

disable_CSM

Valid Values

SK_DISABLE_CSM = 0

- LLC sends ConnectionStatusMsg to all applications (default)

SK_DISABLE_CSM = 1

- LLC does not send ConnectionStatusMsg to any application

SK_DISABLE_FILE_ROLLOVER_LOGGING

SK_DISABLE_FILE_ROLLOVER_LOGGING disables the printing of the switch mid-plane query information banner and the connected SwitchKit application banners within the maintenance_llc.log.

Defaults file variable name

disable_file_rollover_logging

Valid Values

SK_DISABLE_FILE_ROLLOVER_LOGGING = 0

- Enables printing of information about each connected process when the log file rolls over (default)

SK_DISABLE_FILE_ROLLOVER_LOGGING = 1

- Disables printing of information about each connected process when the log file rolls over

SK_DISABLE_PPL_EVENT_LOGGING

Set SK_DISABLE_PPL_EVENT_LOGGING to stop the SwitchManager from printing identified PPL events to an alarm.log file.

Defaults file variable name

disable_ppl_event_logging

Valid Values

SK_DISABLE_PPL_EVENT_LOGGING = 0

- Enables logging of all identified PPL events (default)

SK_DISABLE_PPL_EVENT_LOGGING = 1

- Disables logging of all identified PPL events

**SK_DISABLE_PPL_UNKNO
WN_EVENT_LOGGING**

Set SK_DISABLE_PPL_UNKNOWN_EVENT_LOGGING to stop the SwitchManager from printing unknown PPL events to an alarm.log file.

Defaults file variable name

disable_ppl_unknown_event_logging

Valid Values

SK_DISABLE_PPL_UNKNOWN_EVENT_LOGGING = 0

- Enables logging of all unknown PPL events (default)

SK_DISABLE_PPL_UNKNOWN_EVENT_LOGGING = 1

- Disables logging of all unknown PPL events

**SK_DISPLAY_
CONNECTION_COUNT**

SK_DISPLAY_CONNECTION_COUNT is used to enable the logging of the total number of applications connected to LLC any time an applications connects or disconnects.

Defaults file variable name

display_connection_count

Valid Values

SK_DISPLAY_CONNECTION_COUNT = 0

- No additional information will be logged (default).

SK_DISPLAY_CONNECTION_COUNT = 1

- Text similar to the following will be printed in the maintenance.llc.log each time a connection to an application or node is created or destroyed.

Example

```
Jun 19 2002 16:33:49: New Link - total created: 69 -
active count: 5
```

```
Jun 19 2002 16:33:49: Destroyed Link - total created: 69 -
active count: 4
```

If this environment variable is not defined, the usual text will be displayed when connections are created or destroyed.

SK_ENABLE_CHANNEL_OWNERSHIP_LOGGING

Important! SK_ENABLE_CHANNEL_OWNERSHIP_LOGGING should only be used after consulting with a Dialogic Technical Support representative.

Use this environment variable to enable logging of the channel ownership changes which are then logged in *messages.log*. The logs include the channel state information as well as information about the application gaining/losing ownership. Channel ownership by an application can change for many reasons. Here are some examples:

- Gain of ownership
 - AllocateChannel (and AllocateChannelGroup)
 - RFS via the watchChannelGroup method
 - RequestChannel
 - requestOutseizedChannel
 - requestRouteControlledChannel)
 - TransferChannel (from another application)
 - Redundant Application Pool (RAP) switchover to new primary
 - Outsize or routeControl on an unclaimed channel
- Loss of ownership
 - Transfer channel (to another application)
 - sk_returnChannel
 - ChannelReleased (withdata) message received from switch
 - DS0 of purge or out of service
 - Termination of application
 - Redundant Application Pool (RAP) switchover to new primary

SK_ENABLE_CHANNEL_OWNERSHIP_LOGGING needs to be set in conjunction with *SK_DEBUG_CHANNEL_MANAGER* (2-29).

Defaults file variable name

enable_channel_ownership_logging

Valid Values

- SK_ENABLE_CHANNEL_OWNERSHIP_LOGGING = 1
- Enables logging of channel ownership
- SK_ENABLE_CHANNEL_OWNERSHIP_LOGGING = 0
- Disables logging of channel ownership (default)

SK_FILE_CLOSEOUT_HOUR

SK_FILE_CLOSEOUT_HOUR specifies the hour on which to cycle all managed log files. You may not want to set this variable, if you have already set SK_FILE_LIFETIME. You can dynamically change this environment variable using the SwitchKit graphical user interface, CSA. See the Converged Services Administrator User's Guide, "Changing Environment Variables at Run-Time."

Defaults file variable name`file_closeout_hour`**Valid Values**`SK_FILE_CLOSEOUT_HOUR = 0`

- File close-out midnight, local time (Default)

Valid values are integers 0-23. If `SK_FILE_CLOSEOUT_HOUR = 7`, all managed logs files will cycle at 7:00 AM local time. Be aware, the longer the log files remain open, the larger they grow. Large files can consume all available disk space and cause system errors. It is recommended that log files never remain open longer than 24 hours.

**SK_FILE_EXPIRE_
ALL_AT_ONCE**

`SK_FILE_EXPIRE_ALL_AT_ONCE` will cause all the SwitchKit log files to expire any time one of them expires due to reaching a file size limit. This variable is used in conjunction with `SK_FILE_MAX_SIZE`. This variable is used in conjunction with `SK_FILE_MAX_SIZE`.

Defaults file variable name`file_expire_all_at_once`**Valid Values**`SK_FILE_EXPIRE_ALL_AT_ONCE = 0`

- Do not expire all at once (Default)

`SK_FILE_EXPIRE_ALL_AT_ONCE =1`

- Expire all at once

SK_FILE_LIFETIME

`SK_FILE_LIFETIME` specifies in hours the length of time log files are open. There is no minimum or maximum length of time. You may not want to set this variable, if you have already set `SK_FILE_CLOSEOUT_HOUR`. You can dynamically change this environment variable using the SwitchKit graphical user interface, CSA. See the Converged Services Administrator User's Guide, "Changing Environment Variables at Run-Time."

Defaults file variable name`file_lifetime`**Valid Values**`SK_FILE_LIFETIME = 24`

- Close logs every 24 hours (Default)

The valid range is 1-24

Any whole number of hours can be specified. Be aware, the longer the log files remain open, the larger they grow. Large files can consume all available disk space and cause system errors. It is recommend that log files never remain open longer than 24 hours.

SK_FILE_MAX_SIZE

`SK_FILE_MAX_SIZE` specifies the size, in megabytes, to which the log file is allowed to grow before cycling. When a log cycles because the size is too big, any existing logging timers are not reset. The log will still cycle at the proper time. For example, assuming `SK_FILE_LIFETIME = 1` and `SK_FILE_MAX_SIZE = 1`; if it is now 1:45 p.m. and the log file reaches one megabyte, the log will cycle at 1:45 p.m. The log will cycle again at 2:00 pm.

Defaults file variable name

`file_max_size`

Valid Values

`SK_FILE_MAX_SIZE` can be set to any positive integer value. There is no default value.

Important! Any log will close on a host machine when the `socket.log` on that same machine reaches its size limit. All logs on that machine will roll over. This is done to keep all the log files synchronized.

SK_HANDLER_BEHAVIOR

`SK_HANDLER_BEHAVIOR` allows you to customize the behavior of handlers within an application. The default handler or handler stack acts as a safety net if all other handlers invoked return `SK_NOT_HANDLED`. Currently, the default handler stack is used, if no other handlers are defined.

For example:

- If a channel handler is defined by an application and then a call processing message arrives, the message will be handed to the channel handler(s) or stack of channel handlers.
- If all the channel handlers return `SK_NOT_HANDLED`, then the message will be returned in the original call to `sk_rcvAndDispatch()`.
- If a channel handler is not defined but a default handler is defined, then the message is handed to the default handler or stack of default handlers.

If `SK_HANDLER_BEHAVIOR` is set to `SK_HDLR_BHVR_SAFETY_NET(1)`, a default handler will be used when the appropriate non-default handlers have returned `SK_NOT_HANDLED`.

In effect, the default handler stack acts as a safety net for all messages. In the above example, if a channel handler or handler stack is defined, and all handlers return `SK_NOT_HANDLED`, then the default handler or handler stack will be presented the message if defined and the `SK_HANDLER_BEHAVIOR` is enabled.

Important! The default handler or handler stack cannot act as a safety net to itself.

Defaults file variable name

`handler_behavior`

Valid Values

Use the following to set the bitmask:

`SK_HDLR_BHVR_SAFETY_NET(1)`

SK_LIB_DIR `SK_LIB_DIR` specifies the location of the base directory of the SwitchKit installation. Must be set to the base directory of SwitchKit. If this variable is not set, LLC will terminate. Variable is automatically created for the default installation directory.

Important! When reinstalling SwitchKit on Windows NT®, `SK_LIB_DIR` gets overwritten.

Defaults file variable name

There is no default setting for `SK_LIB_DIR`.

SK_LLC_HOST `SK_LLC_HOST` specifies what machine the LLC is running on. This environment variable should be set similarly on both the primary host and the redundant host. `SK_LLC_HOST` affects all client modules of the LLC. The value may be either a host name or an IP address.

Defaults file variable name

`llc_host`

Valid Values

`SK_LLC_HOST = localhost`

- local host (Default)

SK_LLC_PORT SK_LLC_PORT specifies the port number of the host the LLC resides on. This environment variable should be set similarly on both the primary host and the redundant host. Each LLC and all applications that intend to connect to the LLC must have SK_LLC_PORT set to the same value in their environment.

Defaults file variable name

llc_port

Valid Values

SK_LLC_PORT = 1312

- Port 1312 (Default)

Valid values are 1024-32767

SK_LLC_RETRY_DELAY SK_LLC_RETRY_DELAY specifies the delay between each LLC connection retry. This variable only applies to connections between the LLC and SwitchKit applications, not connections between the LLC and the CSP. For example, this variable can be used for connections between LLC and RLLC, LLC and SwitchManager, and between LLC and SwitchKit user applications.

Defaults file variable name

llc_retry_delay

Valid Value

SK_LLC_RETRY_DELAY = 2
– Retry connection every two seconds (Default)

Valid values include any integer.

Important! Dialogic recommends that this value not be changed from the default.

SK_LOG_LEVEL SK_LOG_LEVEL specifies the EXS API message logging to be done by LLC. You can dynamically change this environment variable using the SwitchKit graphical user interface, CSA. See the Converged Services Administrator User's Guide, "Changing Environment Variables at Run-Time." The possible options are to log to:

- socket.log
- messages.log
- and/or the screen output

Socket.log is the preferred log level.

Defaults file variable name

log_level

Valid Values

Use the following to set the bitmask:

SK_LOG_NONE (0)
– LLC will perform no EXS API message logging
SK_LOG_SOCKET (0x01)
– LLC will log EXS API messages to socket.log (default)
SK_LOG_MESSAGE (0x02)
– LLC will log EXS API messages to message.log
SK_LOG_SCREEN_OFF (0x04)

- LLC screen output is disabled

SK_LOG_DIR SK_LOG_DIR specifies the directory where all SwitchKit log files that are generated on this host computer will be stored. If this variable is not set, log files will be stored in the directory from which the process was started. See *Setting Environment Variables on Windows NT® 4.0 (2-24)* or *Setting Environment Variables on Windows® XP (2-25)*.

Defaults file variable name

log_dir

Valid Values

Default = None

Any network path that is read/write accessible.

SK_DIALOGUE_MONITOR SK_DIALOGUE_MONITOR is used to activate the LLC's dialogue monitor.

Defaults file variable name

dialogue_monitor

Valid Values

SK_DIALOGUE_MONITOR=0

- -disables the LLC's dialogue monitor (default)

SK_DIALOGUE_MONITOR=1

- - enables the LLC's dialogue monitor

See *LLC: Application Load Balancing for TCAP (7-18)* for more information on how to use the LLC's dialogue monitoring capabilities.

SK_NO_DEFAULT_CONNECTION By default, LLC connections are established using the SK_LLC_HOST, SK_LLC_PORT, SK_RLLC_HOST and SK_RLLC_PORT definitions. When a connection is made, these values determine what LLC is connected to if sk_createConnection has never been called. Enabling SK_NO_DEFAULT_CONNECTION stops an application from using the environment settings to determine the LLC location. To establish the LLC connection, the application must call sk_createConnection. After the initial call, any time the LLC connection must be re-established, the data provided in the original sk_createConnection is used.

Defaults file variable name

no_default_connection

Valid Values

SK_NO_DEFAULT_CONNECTION = 0

- Establish connections based on environment definitions.
Default.

SK_NO_DEFAULT_CONNECTION = 1

- Require `sk_createConnection` to establish connection to LLC.

**SK_NUM_RETRIES_TO
_LLC**

SK_NUM_RETRIES_TO_LLC specifies the number of times an application will retry the connection to LLC before NACKing the request. This value applies to all applications, including Redundant LLC. It does not apply to the SwitchMgr.

Defaults file variable name

`num_retries_to_llc`

Valid Value

SK_NUM_RETRIES_TO_LLC = 0

- Retry connection to LLC once (Default)

SK_NUM_RETRIES_TO_LLC = 1

- Channel can only be released by application that initially requested the channel.

This variable may be set to any number of retries, but retries take time. The application will be waiting for a response the entire time. SK API is retrying to send the message.

**SK_RETURN_CHANNEL_B
Y_OWNER_ONLY**

SK_RETURN_CHANNEL_BY_OWNER_ONLY controls the channel release behavior related to the application that initially sent the channel request. When this environment variable is disabled, any application can return any channel.

Defaults file variable name

`return_channel_by_owner_only`

Valid Values

SK_RETURN_CHANNEL_BY_OWNER_ONLY = 0

- Feature disabled. Any application can return any channel.
(Default)

SK_RETURN_CHANNEL_BY_OWNER_ONLY = 1

- Channel can only be released by application that initially requested the channel.

SK_RLLC_HOST SK_RLLC_HOST specifies the IP Address of the host the redundant LLC resides on. This environment variable should be set similarly on both the primary host and the redundant host.

Defaults file variable name

rllc_host

Valid Values

SK_RLLC_HOST = localhost
– local host (Default)

SK_RLLC_PORT SK_RLLC_PORT specifies the port number of the host the redundant LLC resides on. This environment variable should be set similarly on both the primary host and the redundant host. Each LLC and all applications that intend to connect to the RLLC must have SK_RLLC_PORT set to the same value in their environment.

Defaults file variable name

rllc_port

Valid Values

SK_RLLC_PORT = 1312
– Port 1312 (Default)

Valid values are 1024-32767

SK_SIP_UA_APPSELECT_OPTION SK_SIP_UA_APPSELECT_OPTION allows load sharing of SIP user agent (UA) Accept Registration Requests. SIP UA load sharing can be enabled from a user application by sending a sk_pplComponentRegister(0xA7) and then setting the environment variable, with the appropriate option, on the host.

When enabled this load sharing feature will cause only one PPL event indication to be sent to a user application (CSA and Switchmgr will be excluded from the list). Because only a single PPL event indication is expected for each of these SIP transactions, it is not required that the LLC keep track of applications load. It will simply select the next registered application by using a round robin or randomizing algorithm.

Defaults file variable name

sip_ua_appselect_option

Valid Values

Value	Option	Description
0 default	SIP_UA_BROADCAST	No load sharing. Sends to all registered applications.
1	SIP_UA_RANDOMIZE	Sends to one randomly selected application.
2	SIP_UA_ROUNDROBIN	Sends to one application, which is selected in a round robin manner. The first available application is selected; the selection process then proceeds through the applications, always choosing the next one available starting from the application it last selected.

SK_SOCKET_VALIDATE_WAIT

When an application connects to LLC, a handshaking occurs through which critical information regarding the connection is exchanged. **SK_SOCKET_VALIDATE_WAIT** allows the time allocated to validate a client connection to the LLC to be changed.

Defaults file variable name

socket_validate_wait

Valid Values

SK_SOCKET_VALIDATE_WAIT = 3

- Allow three seconds for socket validation (Default)

This variable may be set to any number of seconds.

SK_LLC_SWITCHBACK_MODE

SK_LLC_SWITCHBACK_MODE can be used by the LLC to determine whether to automatically switchback an LLC to its original Primary/Secondary configuration; or remain in its current fail-over configuration until told to do otherwise by the **SK_LLC_CONTROL_START_SYNC_TLV** in *LLCControl*.

Defaults file variable name

llc_switchback_mode

Valid Values

Value	Option	Description
0 default	SB_PRIMARY_PREFERRED_MODE	The LLC designated as Primary (no -r) will always attempt to gain control and become active. In the event that the primary active LLC receives a GO_STANDBY request, a switch-over will occur. Once synchronization is complete between the two LLCs a switchback will occur and the primary will regain control from the secondary.
1	SB_MANUAL_MODE	Whichever LLC is currently active will remain active until told to GO_STANDBY manually by an <i>LLCControl</i> message. In the event that two LLCs are concurrently in the active state, and have been configured by an <i>AddLLCNode</i> message, the one that has been configured the longest will win arbitration. If the configuration times are equal then the Primary (no -r) will win arbitration.

SK_TRANSFERCHAN_OPTION

SK_TRANSFERCHAN_OPTION allows you to decide when the LLC will transfer channel ownership for a particular application. The default, option 0, is to transfer the ownership when the LLC receives a *TransferChanMsg*. When option 1 has been selected, the transfer will not take place until the LLC receives the *TransferChanMsgAck* and has successfully sent it to the requesting application.

Defaults file variable name

transferchan_option

Valid Values

SK_TRANSFERCHAN_OPTION = 0

- transfer application ownership on transferChan request (Default if not defined).

SK_TRANSFERCHAN_OPTION = 1

- transfer application ownership on transferChanAck

SK_WARN_ORPHAN_RFS

SK_WARN_ORPHAN_RFS specifies that when the switch sends a *Request For Service* or *Request for Service with Data* on a channel that is not associated with an application, a warning message is logged in the maintenance.log file. The warning message states: “Message RFS received on channel *n* that was not associated with any app” .

You can dynamically change this environment variable using the SwitchKit graphical user interface, CSA. See the Converged Services Administrator User’s Guide, "Changing Environment Variables at Run-Time."

Defaults file variable name

warn_orphan_rfs

Valid Values

- SK_WARN_ORPHAN_RFS = 0
- No Warning (Default)
- SK_WARN_ORPHAN_RFS = 1
- Log Warning

3 Running SwitchKit Components and Companion Products

Purpose Now that you have installed SwitchKit, it is time to connect to the CSP and perform configuration. This chapter guides you through running each component/companion product, explaining the proper order of execution, and all available startup arguments. Dialogic provides you with a sample configuration file to get you started using SwitchKit.

You can run both the LLC and SwitchManager automatically or manually in both UNIX and Windows NT®. The recommended method depends on the environment in which it is running, as follows:

- **Dedicated** - Single-user development environment with a dedicated CSP. If you are running SwitchKit in a dedicated environment, you may run it automatically or manually.
- **Shared** - Multi-user development environment, or one without a dedicated CSP. If you are running SwitchKit in a shared environment, run it manually to avoid thrashing over CSP resources.
- **Production** - Run-time environment. If you are running SwitchKit in a production environment, run it automatically.

Running the LLC

Purpose This procedure describes the available parameters for running the Low-Level Communicator (LLC) and how to run the LLC with or without any parameter specification.

Before you begin The CSP system software must be installed before you can start the LLC. SwitchKit must also be installed on your system. The LLC parameters must not be already specified through the shortcut settings in the **Start** menu, refer to *Configuring SwitchKit to Run from the Start Menu*.

Steps to start the LLC with CSA Do the following to start the LLC:

-
- 1 Go to the **Start** button, select **Programs**→**SwitchKit**→**llc**.

 - 2 The **llc** window opens, indicates that it is ready to accept connections, and then minimizes.

 - 3 Now you are ready to start SwitchManager.

END OF STEPS

Steps to start the LLC without CSA Do the following to start the LLC:

-
- 1 Open a Command Prompt window.

 - 2 Specify the path to your installation directory. For the default location C:\Program Files\Excel Switching Corporation\ SwitchKit.

-
- 3** Type `llc` and specify the start-up argument, if desired. You can use multiple arguments together. See the *LLC Arguments (3-4)*.

END OF STEPS

LLC start-up behavior

When the LLC connects to the CSP, *Poll* messages from the CSP Matrix Series 3 card indicate the presence of system software on the CSP. If no system software is found, the LLC will wait until the system software is downloaded by the TFTP server. If a TFTP download is already in progress, LLC does not interfere and waits for the download to complete.

After the download is complete, `messages.log` or `socket.log` records all messages (such as polls) arriving from the switch. The `maintenance_llc.pid.log` file records administrative details.

When the LLC is running, all downloading has been completed, and the *****Ready to accept connections** message appears, SwitchManager can be started.

When LLC is running, it automatically sends the host system time to all the CSP Matrix Series 3 cards in the system.

LLC Arguments

Purpose Use the arguments listed in this section when in a Windows® Command Prompt to change the default behavior of LLC at startup.

Argument -p **Change LLC Port**

By default, the LLC accepts TCP/IP connections from SwitchKit applications on Port 1312. If you want the LLC to bind to a different port, use the **-p portNumber** argument. All applications must be informed of any non-default port bindings for them to connect to the LLC.

Syntax

`-p <Port Number>`

Example

```
llc -p 2000
```

Argument -r **Run LLC in Redundant Mode**

To run the LLC in redundant mode as a hot standby, use the **-r** argument. Use this argument when starting the redundant LLC. The primary LLC should not use this argument. To use LLC redundancy, the SK_LLC_HOST, SK_LLC_PORT, SK_RLLC_HOST, SK_RLLC_PORT environment variables must be set.

Syntax

`-r <Internet Address of the Primary LLC>`

Example

```
llc -r 150.10.22.100
```

Argument -v **Query LLC Version**

Displays the version of LLC that is running.

Syntax

`-v <no value>`

Example

```
llc -v
```

Argument -h Help

Displays a list all of the available arguments for the LLC.

Syntax

```
-h <no value>
```

Example

```
llc -h
```

Running Redundant LLCs

Purpose This procedure describes the available parameters for running a redundant Low-Level Communicator (LLC) on Windows® or UNIX.

Before you begin The CSP system software must be installed before you can start an LLC. SwitchKit must also be installed on your system. The LLC parameters must not be already specified through the shortcut settings in the **Start** menu, refer to *Configuring SwitchKit to Run from the Start Menu*.

Steps to start LLCs without CSA Do the following to start redundant LLCs:

1 Open a Command Prompt window.

2 Specify the path to your installation directory. For the default locations:

- **Windows** C:\Program Files\Excel Switching Corporation\SwitchKit
- **UNIX** /usr/local/switchkit/bin

3 On the primary Host (Host 1) set the following system variables:

```
SK_LLC_HOST=localhost
SK_LLC_PORT=1312
SK_RLLC_HOST=<IP Address> (IP Address of Host 2)
SK_RLLC_PORT=1312
```

4 On Secondary Host (Host 2) set the following system variables:

```
SK_LLC_HOST=<IP Address> (IP Address of Host 1)
SK_LLC_PORT=1312
SK_RLLC_HOST= localhost
SK_RLLC_PORT=1312
```

5 Start Primary LLC on Host 1:

- *Windows Syntax* llc
 - *UNIX Syntax* ./llc
-

6 Start redundant LLC on Host 2

Windows Syntax llc -r <IP Address of Host 1>
UNIX Syntax ./llc -r <IP Address of Host 1>

7 Start SwitchManager on either Host 1 or Host 2.

8 See that the primary LLC shares data sending multiple messages to the "new LLC".

"Done sharing data with new llc...."

9 Verify that the LLC on Host 2 is in a standby state. The example below shows the type of messages your would expect to see in your console window:

```

Jul 22 2004 11:38:22: using /usr/local/switchkit for
SK_LIB_DIR
Jul 22 2004 11:38:22: **Running with ExcelLC
Jul 22 2004 11:38:22: Waiting for connections on port
1312
Jul 22 2004 11:38:22: Connecting to 135.119.52.167
(135.119.52.167) on port 1312
Jul 22 2004 11:38:22: Connected to 135.119.52.167 as 2
Jul 22 2004 11:38:22: Found another llc running,
receiving sync info...
Jul 22 2004 11:38:22: Received LogLevel Message and set
log level to 1
Jul 22 2004 11:38:22: This LLC is Synchronized
Jul 22 2004 11:38:22: ** Standing by in redundant
configuration
END OF STEPS

```

Basics of SwitchManager

Description SwitchManager establishes a connection to the LLC, configures the switch as specified in the configuration file and begins to monitor the switch. SwitchManager generates an *alarm.log* file that logs all API messages sent to and from the switch, and a *maintenance_switchmgr.pid.log* file. Use the *alarm.log* file to debug configuration issues and to diagnose run-time problems that may occur.

Important! In some cases, the sequence number of an API message in an *alarm.log* is different from the sequence number shown in a *socket.log*.

You will see many messages displayed in the SwitchManager windows. These API messages are also viewable in the *alarm.log* file. Messages to cards referenced in the sample configuration file that you do not have in your system are rejected by the switch with a negative acknowledgment (NACK). When debugging a configuration, begin by analyzing the first NACK in the *alarm.log* file. Many times, multiple alarms are propagated from one initial problem. By fixing the initial problem, subsequent alarms may be eliminated as well

System Start-up Behavior SwitchManager will connect to LLC. The configuration file specified when starting SwitchManager must include at least one *AddLLCNode* () command. LLC will initiate connections to all nodes specified in these commands and continue to configure according to the configuration file.

Important! When the Switch Manager reconnects to the LLC, *AddLLCNode* messages are sent from the Switch Manager to the LLC. This causes the LLC to open a socket (connection) to the CSP and the *AssignNode* message (which is the switch-level equivalent of *AddLLCNode*) is sent to the CSP.

Running SwitchManager Automatically If you have modified your shortcuts for Windows® to include your configuration file name, select SwitchManager from your **Start**→**Programs** menu. The command line arguments are set in **SwitchConfig**. See *Running SwitchKit Automatically at Startup on Windows® (2-13)*.

Starting SwitchManager

Purpose This procedure explains how to start SwitchManager for the first time when you plan to use the Converged Services Administrator (CSA) to create your CSP configuration. If you are not using CSA, see *Starting SwitchManager with tandem.cfg (3-11)* for information on how to start SwitchManager with an example configuration.

Before you begin The CSP system software must be installed before you can start the SwitchManager. SwitchKit must be installed on your system. You must have the *sk_license.dat* file in your installation directory. For example, using the default directory, you would install the license file in:
\ProgramFiles\Excel Switching Corporation\SwitchKit

SwitchManager arguments must not be already specified through the shortcut settings in the **Start** menu. See *SwitchManager Arguments (3-13)*.

Steps for starting SwitchManager for the first time when using CSA

Do the following to start SwitchManager when you are using it for the first time and using the CSA:

1 Open a **Command Prompt** window.

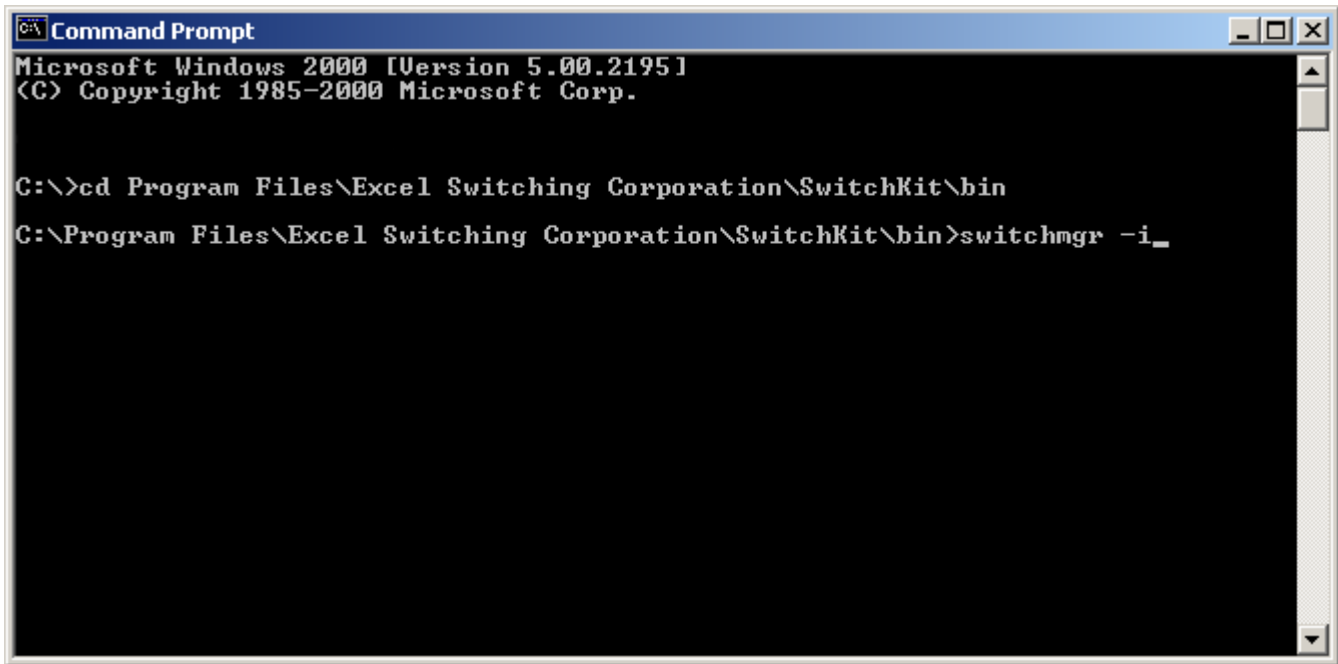
2 Specify the path to your installation directory. The default location for Windows NT® is:

C:\Program Files\Excel Switching Corporation\SwitchKit\bin.

3 Type the following:

```
switchmgr -i
```

See the next screen shot.



```
Command Prompt
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\>cd Program Files\Excel Switching Corporation\SwitchKit\bin
C:\Program Files\Excel Switching Corporation\SwitchKit\bin>switchmgr -i_
```

SwitchManager will start and become active.

- 4 Minimize the **Command Prompt** window when you see the message:

fopen:No such file or directory

Now you can start the CSA to create and send a configuration to the CSP. See the *Converged Services Administrator User's Guide*.

END OF STEPS

Starting SwitchManager with *tandem.cfg*

Purpose This procedure explains how to start SwitchManager using an example configuration file.

Before you begin The CSP system software must be installed before you can start the SwitchManager. SwitchKit must be installed on your system. You must have the *sk_license.dat* file in your installation directory. For example, using the default directory, you would install the license file in: *\ProgramFiles\Excel Switching Corporation\SwitchKit*. You cannot use the *tandem.cfg* file to assign spans unless you modify the file to include your license key.

SwitchManager arguments must not be already specified through the shortcut settings in the **Start** menu. See *SwitchManager Arguments (3-13)*.

Steps for starting SwitchManager with a sample configuration

To get started with SwitchKit more easily, Dialogic provides you a sample configuration file named, *tandem.cfg*. Do the following to use this file:

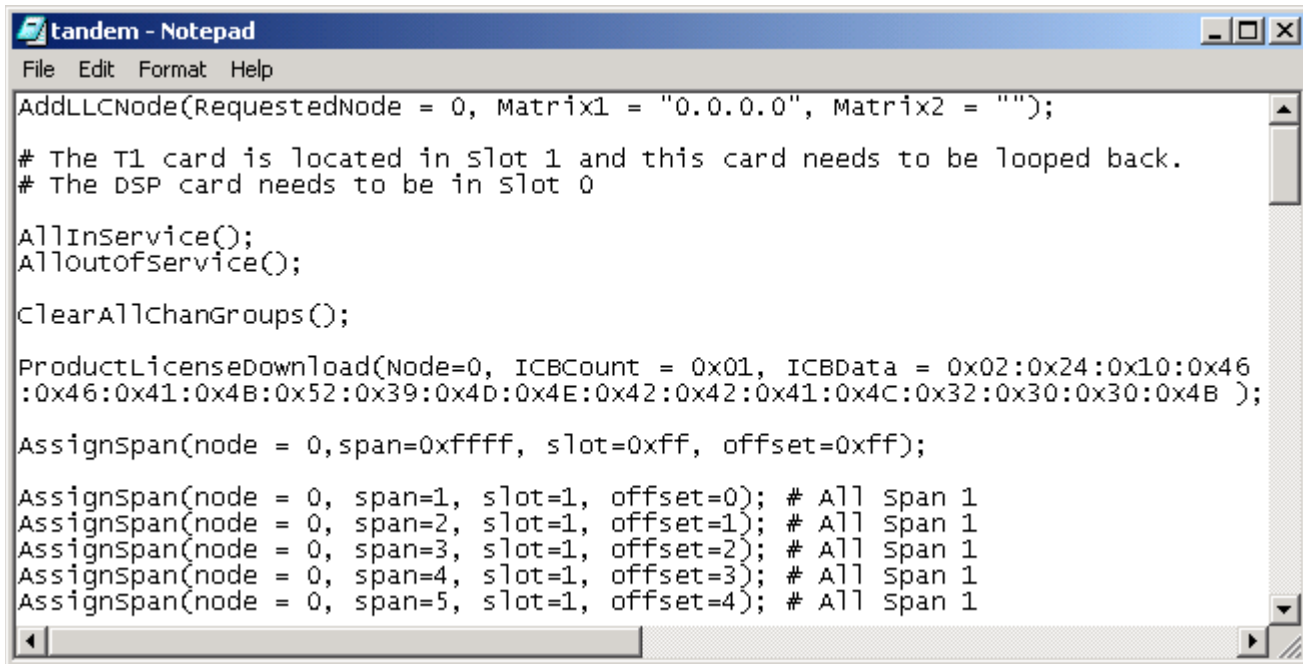
-
- 1 Open the *tandem.cfg* file using a text editor. The file's default location for Windows NT® is:

C:\Program Files\Excel Switching Corporation\SwitchKit\Samples\cfg.

- 2 In the *tandem.cfg* file, change the IP address of `Matrix1` and enter slot information to correspond with your CSP. If you have a redundant CSP Matrix Series 3 card, add the IP address for this as well. Save the file.
-

- 3 If you want to assign spans, you must add your product key which can be entered in either hexadecimal or decimal format. In your text editor add the *ProductLicenseDownload* message to *tandem.cfg*.

See the example below for the correct syntax. Note that the Node in the ProductLicenseDownload message must be the same as the RequestedNode in the AddLLCNode message. The license key is the ICBDData.



```

tandem - Notepad
File Edit Format Help
AddLLCNode(RequestedNode = 0, Matrix1 = "0.0.0.0", Matrix2 = "");
# The T1 card is located in slot 1 and this card needs to be looped back.
# The DSP card needs to be in slot 0
AllInservice();
Alloutofservice();
ClearAllChangGroups();
ProductLicenseDownload(Node=0, ICBCCount = 0x01, ICBDData = 0x02:0x24:0x10:0x46
:0x46:0x41:0x4B:0x52:0x39:0x4D:0x4E:0x42:0x42:0x41:0x4C:0x32:0x30:0x30:0x4B );
AssignSpan(node = 0,span=0xffff, slot=0xff, offset=0xff);
AssignSpan(node = 0, span=1, slot=1, offset=0); # All Span 1
AssignSpan(node = 0, span=2, slot=1, offset=1); # All Span 1
AssignSpan(node = 0, span=3, slot=1, offset=2); # All Span 1
AssignSpan(node = 0, span=4, slot=1, offset=3); # All Span 1
AssignSpan(node = 0, span=5, slot=1, offset=4); # All Span 1

```

4 Save *tandem.cfg* and close it.

5 Open a Command Prompt window.

6 Specify the path to your installation directory. The default location is
C:\Program Files\Excel Switching Corporation\SwitchKit\Samples\cfg.

7 Type the following:
switchmgr tandem.cfg

END OF STEPS

SwitchManager Arguments

Purpose Use the arguments listed in this section when in a Command Prompt to change the default behavior of SwitchManager.

Argument -i This argument allows SwitchManager to be started without any configuration file being specified, usually during initial configuration. It will scan for an existing system.cfg file and, if present, will clear the configuration therein allowing the user to start from a clean slate. This option is used when building a new configuration or loading an existing configuration using an Dialogic graphical user interface. Be advised if loading an existing configuration, the CSP will be reconfigured without regard to the tags. Dialogic recommends that the -i argument should not be used on a live system.

Syntax

-i

Example

```
switchmgr -i
```

Argument -n **Maintain Only**

SwitchManager does not do any configuration on start-up. It does, however, assume that the contents of the configuration file as already been loaded onto the switch. Should any cards fail, it reconfigures them according to the configuration file. This is useful if you know that the switch is already configured correctly, and you want to avoid taking everything out of service.

Important! This option only applies to the initial startup of the first SwitchManager to connect to the LLC. In redundant SwitchManager instances, the redundant LLC will start in Smart Mode since it is actually a continuation of the first instance of SwitchManager. Also, if the active SwitchManager were to disconnect and reconnect to LLC it would behave as in Smart Mode.

Syntax

-n <filename>

Example

```
switchmgr -n tandem.cfg
```

Argument -s Smart Mode

SwitchManager examines the configuration tags of all cards in the system that are taggable. If any cards contain tags different from what is expected, SwitchManager configures that card. If all the tags are the same, SwitchManager does nothing.

Important! Dialogic recommends the -s argument be used when running SwitchManager in a live environment.

Syntax

```
-s <filename>
```

Example

```
switchmgr -s tandem.cfg
```

4 Redundancy Setup

Purpose This chapter explains the redundancy setup for operational systems.

SwitchKit Redundancy

Overview In the basic SwitchKit architecture, all call processing and OAM functions are performed through the LLC. All applications require the LLC for communicating with the switch. If the LLC stops running, no further call processing can take place. To eliminate the LLC as a single point of failure, run a redundant LLC.

Description The Redundant LLC (RLLC) is a hot standby for the Primary LLC (PLLC). The RLLC mirrors the state of the PLLC, and, if the PLLC goes down, RLLC becomes the primary LLC. All applications connect to the RLLC automatically.

Running LLC in Redundant Mode To run the LLC in redundant mode, use the **-r** argument. The **-r** flag tells the RLLC where to find the PLLC. This can consist of just a host name, or optionally a host name followed by a colon followed by a TCP port where the primary LLC is waiting for connections. By default, it assumes the default port of 1312. See *Running Redundant LLCs (3-6)*.

You cannot run two LLCs on the same host computer accepting socket connections on the same port.

Running SwitchManager in Redundant Mode For redundancy, SwitchManager uses the LLC and Application Redundancy. See *Redundancy (10-1)* in the *SwitchKit Programmer's Guide*. When multiple SwitchManagers connect to LLC, the first one connected becomes Primary. This SwitchManager remains primary unless there is an LLC switchover. In this case, the new active LLC selects the first SwitchManager that connects as primary. In some cases, this induces a switchover of SwitchManagers, which is transparent to the system.

Specifying the location of the RLLC for Applications Once you are running an RLLC, all applications that connect to the PLLC, including SwitchManager, must know the location of the RLLC so they can automatically reconnect to the RLLC in case of a PLLC failure. This is done through the following environment variables:

- SK_RLLC_HOST
- SK_RLLC_PORT

These values must be set before the applications are run. They are only examined upon start-up. You cannot dynamically configure SwitchKit Redundancy. Please refer to *Environment Variables Modify SwitchKit Default Behavior (2-21)*

Example

For example, in the default file, you could have:

```
rllc_host: pc2  
rllc_port:1200
```

This would specify to all applications starting up that there is a redundant LLC running on host pc2, on the TCP port of 1200. Environmental variables must be set the same on both the Primary host and the Redundant host.

Failure Scenarios Resolved with Redundancy

Description You can configure LLC Redundancy to avoid disruption of service in a production environment. Avoid as many single points of failure as possible in your configuration.

The following diagrams illustrate a redundant configuration using two host computers.

Host 1 is acting as the primary host and is running the following:

- LLC (PLLC) connected to the CSP Matrix Series 3 cards in the CSP.
- The primary SwitchManager, connected to the PLLC manages the node.
- Any custom application, such as call processing applications and SNMP.

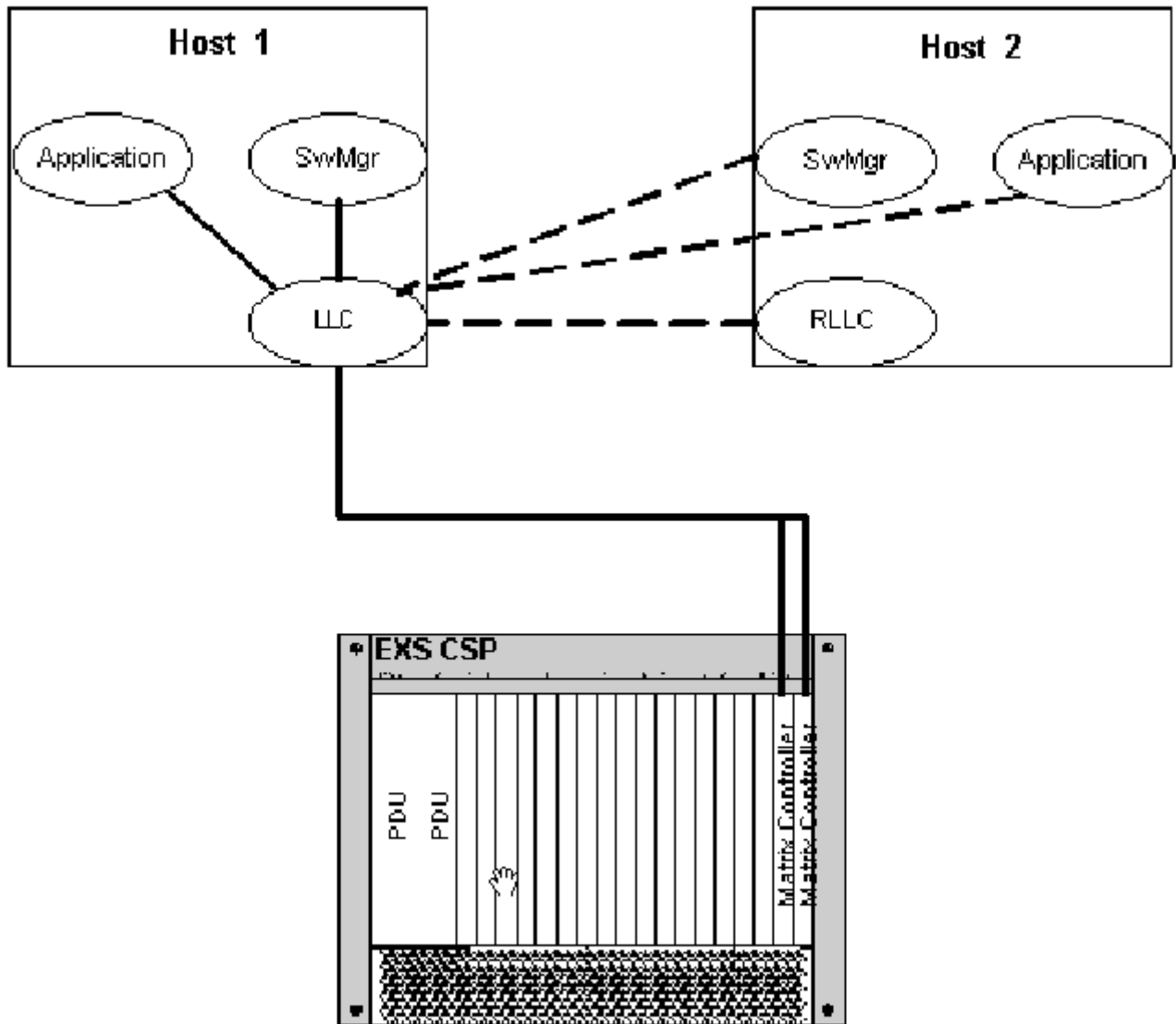
Host 2 is acting as the Secondary Host and is running the following:

- The RLLC, which is connected to the PLLC (rather than the CSP Matrix Series 3 card).
- A secondary SwitchManager, which takes over management of the node should the primary SwitchManager fail.
- A second copy of the custom application. The LLC shares traffic between the two applications.

Line Key In the following diagrams we use different lines to illustrate different states of the connection.

- Solid Line
A solid line illustrates an established connection.
- Bold Solid Line
A bold solid line illustrates a new established connection.
- Dotted Line
A dotted line illustrates a lost connection.

Figure 4-1 SwitchKit Redundancy Setup



Primary Host Failure

Should the Primary Host fail, RLLC recognizes the failure by the lack of Poll message responses from the PLLC and it becomes the active LLC.

If the primary host fails, SwitchManager, the Converged Services Administrator (CSA), and custom applications, continue processing as before. When the applications attempt to connect to the LLC again, they first attempt to connect to the PLLC, which fails, and will then connect to the RLLC. This is done automatically and only requires that the IP addresses for the primary & redundant host are defined. There are no additional application requirements for this.

Primary SwitchManager Failure If the primary SwitchManager fails, the secondary SwitchManager receives an application disconnect message. It then takes over control of the configuration and management of the switch.

Important! When SwitchManager starts, it configures the entire CSP unless an alternative option is used. It is recommended that both the primary and redundant SwitchManagers in a production environment are configured to use Smart Mode.

Primary LLC Failure Should the primary LLC fail, the RLLC recognizes the failure by the lack of response to Poll messages between them. The RLLC becomes the active LLC. It takes over the connections to all matrices in the CSP and accepts connections from all applications.

5 Building a Configuration File

Purpose When you run SwitchManager, it first queries the CSP to find out its current state, the number of nodes, and the type of cards in each node. Then, it reads the configuration file and converts each configuration message to the corresponding SwitchKit message. SwitchManager sorts these messages in the order required by the switch and sends them to the switch. SwitchManager does not validate the data provided in the configuration file. If the data is incorrect, the messages will be rejected by the switch with a NACK.

SwitchManager waits to receive an acknowledgment (ACK) for each message before sending the next message. If the ACK is not received in 10 - 15 seconds, it will resend the message. If the resent message is not acknowledged, SwitchManager continues with the next message.

Building a Configuration File

Overview The format of the configuration file is a series of configuration message specifications. The order of the parameters does not matter and white space and comments (delimited by #) are allowed. Example configuration files are provided in the folder: `sample/cfg`.

Some parameters of each SwitchKit configuration message are optional. The parameters used depend on the form of the message you are using, and which AIBs are being populated. Depending on the parameter, the value can be an integer (for example, 3), a quoted string (e.g., "hello there"), an unsigned data array, or a range.

Important! Configuration files should be created using CSA. If you create a configuration file by hand or alter one created by CSA, be certain the data for each message is properly assigned. Errors in data format can cause SwitchManager to fail.

Syntax Configuration messages are not case sensitive. A message specification has the following format:

```
message_name(parameter1=value1, parameter2=value2,...);
```

Example `AssignSpan(node = 0, span = 1, slot=1, offset=0);`

Message Format The message formats are specified in the *Messages.api.h* file for C Structures and *CMessages.api.h* file for C++ classes. You can reference these file to see what the message and parameter names are. In the configuration file, the `XL_`, `SK_`, `XLC_`, and `SKC_` prefix is always omitted.

AddLLCNode The *AddLLCNode* message specifies the node ID and CSP Matrix Series 3 card IP addresses for each node. For configuration messages (such as AnswerSupervision) that use the range parameter, the node ID can be specified in the message but will be validated by the node ID in the *AssignSpan* message. Configuration messages such as, *AssignSpan* must have the node ID that is assigned with the *AddLLCNode* message.

All-In-Service and All-Out-of-Service The *AllInService* and *AllOutOfService* messages allow you to bring all components (for example, spans and channels) of your configuration in or out of service with one message. These messages can be added to any configuration file. If these messages are included in a configuration file, SwitchManager brings all components out of service prior to any

configuration and brings all components back into service after configuration is complete. If your configuration requires that some components remain out of service after configuration, you can add individual *ServiceStateConfigure* messages for each component in your configuration file.

Range Parameter For configuration messages that apply to a range of channels, instead of specifying the range with four different parameters, you can use the special range parameter. The range argument can contain the name of a channel group. This channel group must be defined in the same configuration file; it cannot refer to a previously-defined channel group in a different configuration file. By using this feature, you can set up your channel groups only once in your file, and all other messages can just refer to those groups.

The range parameter has the following format:

StartSpan:StartChannel - EndSpan:EndChannel.

Example: TrunkTypeConfig from Span 0, Channel 0, to Span 6, Channel 23, would be specified as:

```
TrunkTypeConfig(range = 0:0 - 6:23, trunkType = 0x01)
```

Data Array To specify a data array as an argument, list all the bytes individually, separated by a colon (:). The data array must be preceded by the Number of Data Locations.

Example: To do a SystemConfig with a Data array of the bytes 0x01 0x00 0x0a, specify it as:

```
SystemConfig(
    ConfigType = 0x03,
    Data = 0x00:0x01:0x01:0x01:0x05:0x01:0x00:0x0a;
```

Configuration Stages SwitchManager automatically re-orders all the messages in the file into a legal configuration sequence. To force a certain ordering, use ConfigStage directives. Sets of messages can be grouped into stages. SwitchManager first sorts all the stages in numerical order, and then, within each stage, re-orders the messages into a legal sequence. All messages in stage n are guaranteed to be sent before any message in stage n+1 is sent.

In the example below, Messages 1 and 2 might be re-ordered, and Messages 3 and 4 might be reordered, but both Messages 1 and 2 are guaranteed to be sent before either Messages 3 or 4.

```
ConfigStage 1:
    msg1();
```



```

    msg2();
ConfigStage 2:
    msg3();
    msg4();

```

Poll Interval Excel recommends that the poll interval between LLC and the switch be between one and five seconds. If the poll interval is not set, it defaults to two seconds. Turning off polls will disable some of the SwitchKit functionality and is strongly discouraged.

Configuration Notification Many applications require the CSP configuration to be complete prior to custom setup or call processing. To determine when the configuration is complete, an application may register for Configuration Status messages. When the configuration is complete, the LLC broadcasts a Configuration Status message (*ConfigStatusMsg*) to the registered applications. You register an application to receive Configuration Status messages using the Message Register function (*sk_msgRegister*).

The Configuration Status message contains the following information:

- Event = (Configuration Begin, Configuration End or Configuration File Parse Error)
- ConfigType = (initial configuration, dynamic configuration, Card Reconfig, ConfigTagReset, etc.)
- Number of messages sent
- Number of messages receiving a Negative Acknowledgement (NACK)

This information can be used to synchronize an application with the Configuration Status as determined by SwitchManager.

Configuration Status messages are sent to the applications before DS0 status changes are returned by the switch.

Configurable Message Timeout If a message is not completed within the message timeout period, the LLC times out. The typical message timeout value is 15 seconds, although it varies by message type. In most cases, the timeout value determined by the LLC is sufficient. However, it may not be sufficient when you are sending a large number of configuration messages to a single node. If you experience LLC timeouts during configuration, you can increase the default timeout value.

To increase the default Message Timeout value, issue the *AdjustMessageTimeout* message. The value you enter is added to the default value. For example, if the default value is 15 seconds and you enter 10 seconds, LLC will time out if it does not receive an acknowledgment message from the switch within 25 seconds (15 + 10).

SwitchManager adjusts the timeout values when configuring a multi-node system. Once configuration is complete, the timeout value is returned to the default.

API Messages Used for Configuration

Overview This section outlines the messages available for various types of configuration related to the following:

- Common SwitchKit API
- Common EXS API
- Common Call Control
- Channel Associated Signaling
- Common Channel Signaling
- Digital Signal Processor
- System-related
- PPL

Finding the Messages Information about each of the messages in this section is in the SwitchKit API Reference. The messages in the sub-categories that follow have links to them in the API reference. In the chapters that follow this one, you find API messages related to device connectivity, line card switchovers and server status.

Common SwitchKit

AddLLCCard
AddLLCNode
AdjustMessageTimeout
AllInService
AllOutOfService
AssociateChanGroup
ClearAllChanGroups
ClearChanGroup
ConfigChanGroup
HexTool
RecordedAnnouncement

Common EXS API

AnswerSuperviseConfig
AssignSpan
BusyOutFlagConfig
DistantReleaseConfig
FlashTimingConfig
GenericCardConfigure
GenericCardQuery
InpulsingParametersConfig
InseizeInstrListConfig
IPAddressConfig
LocalReleaseConfig
LoopTimingConfig
MultiHostConfig
OuseizeInstrListConfig
ProductLicenseDownload
ResetConfig
ResourceAttributeConfig
ServiceStateConfig
Standby Line Card Config
SynchPriorityConfig
SystemConfig
TimeSet

Common Call Control

CrossConnectChannel
CrossConnectSpan
CrossDisconnectChannel
CrossDisconnectSpan

Channel Associated Signaling

DS3GenericMessage
E1SpanConfig
T1SpanConfig
LoopBackConfig
PCMEncodingConfig
ReceiveSignalingConfig
Span Filter Config
StartDialConfig
TransmitSignalingConfig
TrunkTypeConfig

Common Channel Signaling

CCSRedundancyConfig
SS7 Configuration Messages
SS7CICConfig
SS7CLLConfig
SS7ISUPMessageFormatConfig
SS7SCCPTCAPConfig
SS7SignalingLinkConfig
SS7SignalingLinkSetConfig
SS7SignalingRouteConfig
SS7SignalingStackConfig
SS7TUPMessageFormatConfig
VitualSlotConfig
VirtualSpanControl
ISDN & V5 Configuration Messages
BChannelConfig
DChannelAssign

DChannelDeAssign
DChannelFacilityListConfig
ISDNInterfaceConfig
ISDNTerminalConfig
V5Config

Digital Signal Processor

ConferenceCreate
ConferenceDeleted
ConferenceDeleteRequest
ConnectToConference
ConnectOneWayToConference
ConnectOneWayForced
CPAClassConfig
CPAPatternConfig
DSPCacheModify
DSPSIMMConfig
PlayFileModify
PlayFileStart
PlayFileStop
RecAnnConnect
RecAnnDelete
RecAnnDisconnect
RecAnnDownload
RecAnnDownloadInit
RecAnnFSConvert
RecAnnFSDefrag
RecAnnSingleDelete
RecordFileModify

RecordFileStart
RecordFileStop
RecAnnConnect
Resource Create
ResourceDelete
Resource Modify
TransmitCadencePatternConfig
TransmitToneConfig

System-Related

AssignEXSHostSlave
NodeConfig
RingConfig

PPL

PPLAssign
PPLAuditConfig
PPLConfig
PPLDelete
PPLEventRequest
PPLTimerConfig
PPLTransmitSignalConfig

Configuration Message Syntax

Overview The following is a syntax explanation of the CSP configuration messages.

Generic Syntax

```
MessageName(
    Parameter = Value,
    Parameter = Value,
    Parameter = Value);
```

- Enter configuration parameters as provided through the source code.
- Configuration values depend on the customer specifications but must be in a valid range.
- If the message name is followed by empty parentheses, no parameters are required.

Entering Comments To place comments after a message in your configuration file, put a pound symbol (#) at the beginning of the comment. Comments can be placed immediately after the end of the message or on a separate line, as follows:

```
MessageName(Parameter = Value); # comment
or
MessageName (Parameter = Value, Parameter = Value);
# comment
```

Entering Data Arrays To specify a data array as an argument, simply list all the bytes individually, separated by a colon (:). The Number of Data Locations must precede the data array. Example to do a SystemConfig with a data array of the bytes 0x01 0x00 0x0A:

```
SystemConfig(ConfigType = 0x03, Data = 0x01:0x00:0x0A);
```

Presentation To make the parameters easier to distinguish, message formats are presented in the documentation for the API messages in the following manner:

```
MessageName (
    0001 Parameter = Value,
    0002 Parameter = Value,
    0003 Parameter = Value);
```

You can enter messages in the configuration file in this format or in one string, as long as the syntax is correct.

Example The format of the *AnswerSuperviseConfig* message is as follows:


```
AnswerSuperviseConfig(  
    0004 Node = integer,  
    0005 Range = StartSpan:StartChan - EndSpan:EndChan,  
    0006 AnswerMode = integer);
```

To configure spans 0-1 on Node 1 to propagate answer to distant end,
you would enter the following:

```
AnswerSuperviseConfig(  
    0007 Node = 1,  
    0008 Range = 0:0 - 1:23,  
    0009 AnswerMode = 1);
```

6 Logging and Configuration Updates

Purpose This chapter provides information about log files, configuration updates, and upgrading to a multi-node system.

Log Files

Overview All modules within SwitchKit generate log files. You can also generate your own log files, and have them managed by SwitchKit. Log file entries can also be displayed on the screen. By default, SwitchManager and LLC send their output to the screen as well as to the log file, while the applications write it only to the log file. The behavior of the LLC and SwitchManager can be changed through the environment variables.

The following logs are created by the SwitchKit processes at run-time:

Log	created by...
Alarm	SwitchManager
Maintenance	LLC, SwitchManager and each SwitchKit application
Messages	LLC (optional)
Socket	LLC (optional)
User-defined	LLC (optional)

Alarm Log File name: alarm.log

SwitchManager generates this file. It logs all actions taken by the SwitchManager in configuring the switch, and the text of any alarm conditions that occur on the switch. If a switch software fault occurs, it is written to the alarm.log.

Maintenance Log File name: maintenance.xxx.log

This type of log is generated by the LLC, the SwitchManager, and all applications linked with the SwitchKit API library. The actual file names are:

- *maintenance-llc.PID.log*
- *maintenance_switchmgr.PID.log*
- *maintenance_appName.PID.log*
- *maintenance_app.PID.log*

where PID is the process ID of that application.

Important! For the *maintenance_app.PID.log*, the outstanding message limit is 6552 per application. A warning is printed by the

API when the key wrapping occurs: "App has more than 6553 outstanding messages, message collision may occur!".

Messages Log File name: message.log

This file is generated by the LLC and shows all messages sent between the LLC and the switch, using the EXS API. The EXS API uses a hexadecimal numbering scheme. To interpret EXS API message formats, see the EXS API hexadecimal format information for each message.

The message.log includes the following abbreviations:

Abbreviation	Description
=>	ACK (acknowledgement) message
->	outbound or inbound message
H REQ	LLC internal request manager
H LLC	LLC internal module
H ART	LLC internal artificial ACK, likely generated due to ACK timer expiration
H21	External Host Application Number 21
X2001	External Device with logical link value 2001 (200 = NodeType Call Server 1=node ID)
H	LLC Host
UNKNN	Unknown, the LLC was unable to resolve the sender. This should not happen.

Examples

```

Jan 29 2004 19:57:58 X2001 ->H : 00 ab 00 4d 01 00 10 01 00 03 02 09 00 05 00 00
Jan 29 2004 19:57:58 H LLC =>H REQ : 00 9e 00 00 01 00 10 01 00 03 02 08 00 05 00 00
  (orig msg was 00 9e 00 00 01 )
Jan 29 2004 19:57:58 H REQ ->X2001 : 00 b5 00 00 01 01 1d 04 13 39 3a
Jan 29 2004 19:57:58 H LLC =>H REQ : 00 9f 00 01 01 00 10 (orig msg was 00 9f 00 01
  01 14 )
Jan 29 2004 19:57:58 H REQ ->X2001 : 00 83 00 01 01 00 01 01 01 01
Jan 29 2004 19:26:17 H ART =>H REQ : 00 9f 00 04 01 f0 09 (orig msg was 00 9f 00 04
  01 14 )
Jan 29 2004 19:06:31 UNKNN=>H1 : 00 83 00 1b 01 00 10 00 01 01 01 14 f3 00 00 43
04 01 04 00 00 01 79 02 00 00 01 01 01 06 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 f3 00 00 00 00 00 (orig msg was 00 83 00 1b 01 00 01 01 01 14 )

```

Socket Log SwitchKit LLC can create and maintain a log file named, `socket.log`, which logs all the messages going from the CSP to the host and from the host to the CSP. This log is different from `messages.log`, because it logs everything that goes across the socket, prior to processing, including the length of the message.

The socket log file includes the following abbreviations:

Abbreviation	Description
X	CSP
1	Logical Node ID
[10.10.156.24]	Socket IP address
H	Host

Example Socket.log

The following example socket log shows an inbound and outbound message:

```
Jan 11 2002 12:59:57 X1[10.10.156.24]->H : 00 ab 00 38 01 00 10 01 00 05
00 01 01 01 00 00
Jan 11 2002 12:59:57 H->X1[10.10.156.24] : 00 ab 00 38 01
```

For further information refer to *Installing the CSA (2-19)*.

User-specified log files By using the function `sk_getManagedFile`, you can create your own log files, to be managed by SwitchKit.

Log File Management

Overview SwitchKit manages the logs files so they are not overwritten and not allowed to continue to grow indefinitely. When attempting to open a log file for the first time, if a file of that name already exists, it is renamed, using a random three-digit number. For example, whenever the LLC starts, if there is already a *messages.log* file, it is renamed to *messages.nnn.log*, where *<nnn>* is a random three-digit number. All log files used by SwitchKit are automatically closed after a specified amount of time. By default, this is every 24 hours. The first time that *logfile* is closed, it is renamed to *logfile_date.001.suffix*. The suffix of the closed file is preserved. The *logfile* will once again contain current information. Once the closeout period has passed again, *logfile* is moved to *logfile_date_002.suffix*. Closed files can be archived or deleted, without affecting the current logging.

Log Directory By default, the log files are written to the current directory of the program. Alternatively, you can specify ***SK_LOG_DIR*** as the directory in which to write these log files.

Important! Remove log files often to avoid filling your disk space and causing erratic system behavior. If you do not move your log files regularly, lack of disk space may cause a system outage.

Close Time You can specify the time the file is closed by modifying the default behavior. By default, each file is closed every day at midnight local time. The environment variable ***SK_FILE_CLOSEOUT_HOUR*** can be used to modify the time to close the files. The default is 0 (midnight). See ***SK_FILE_CLOSEOUT_HOUR*** (2-32) for more information.

Close Duration Alternatively, the environment variable ***SK_FILE_LIFETIME*** can specify in hours how frequently the files are closed. For example, if ***SK_FILE_LIFETIME*** is specified to be 6, then the log files will be closed 4 times a day, at midnight, 6a.m., noon, and 6p.m. See ***SK_FILE_LIFETIME*** (2-33) for more information. ***SK_FILE_CLOSEOUT_HOUR*** and ***SK_FILE_LIFETIME*** should not both be used on a system.

Maximum Size ***SK_FILE_MAX_SIZE*** specifies the size, in megabytes, to which the log file is allowed to grow before cycling. When a log cycles because the size is too big, the existing timer is not reset. The log will still cycle

at the proper time. See *SK_FILE_MAX_SIZE (2-34)* for more information. When logs roll because of size, optionally, they can all roll using *SK_FILE_EXPIRE_ALL_AT_ONCE*.

SK_FILE_MAX_SIZE can be used with all other log file management environment variables.

Clearing Logs

To clear a log file, perform the following steps:

1. Start the Converged Services Administrator (CSA).
2. From the CSA File menu, select **Provisioning**→**System**→**Clear Logs**.
3. A pop-up window appears. Select the files you want to clear. Click **OK**.

Debugging Channel Management Issues

There is additional logging available to debug channel management issues. As this is CPU-intensive logging, it should only be done with the help of an Excel support engineer. Please contact Excel technical support for assistance.

Updating Configuration

Overview You can dynamically update the CSP configuration in SwitchManager using the Converged Services Administrator (CSA). Dynamic Configuration allows you to reconfigure the CSP, add new trunk groups, or download a voice prompt without restarting SwitchManager. A new configuration file is sent to the switch with the SK_DynamicConfig message.

Whenever a dynamic configuration is sent to SwitchManager, the configuration file and any attached files will be sent to the directory:

```
$SK_LIB_DIR/dynamicConfig/
```

Also, the local file name will attach a timestamp to the file name. This ensures that received dynamic configurations (and attached files) have unique and simple names. The file name uses the following format:

```
YYYY.MM.DD_HH.MM.SS_filename
```

Example of a file name:

```
2005.12.08_14.12.47_addnode.cfg
```

If the `$SK_LIB_DIR/dynamicConfig` directory does not exist, the file transfer will fail. You should not place your local configuration files in the `dynamicConfig` directory. This directory is meant for SwitchManager's use. Any local user-created configurations should be stored elsewhere.

The messages in the new configuration file are sent immediately to the CSP and the new messages merge with those in the current configuration file. You can create a merged configuration file by sending the SK_WriteCfgFile message.

Important! Dynamic configurations are stored in memory by SwitchManager. After many dynamic configurations, local memory may be consumed. You should be aware of how much memory is used by SwitchManager. If memory usage is too great, ensure that the base configuration file used is updated to reflect the correct

configuration of the switch and restart SwitchManager using Smart Mode or Maintain Only options.

Configuring a Multi-Node System

Modifications to the configuration are required for running SwitchKit on a multi-node CSP. You can find an example of a multi-node configuration file in `samples/cfg/exnet.cfg`.

Configuration File Modifications

Modifications required for multi-node systems are as follows:

EXNET® Ring Configuration

Configure the ring with:

```
RingConfig(
    entity=1,
    slot=0xff,
    data=0xff); #DeAssign all rings
RingConfig(
    entity=1,
    slot=10,
    data=1); #Assign ring 1
RingConfig(
    entity=2,
    ring=1,
    data=1); #Configure ring 1 for trans/rec
```

Logical Node ID

For each `assignSpan` command, you must specify the logical node ID. So, for an `assignSpan`, you would have:

```
AssignSpan(node=3, span=9, slot=3, offset=0);
```

Span IDs are unique across the system. An application can send a message without specifying a node ID, and the LLC will automatically fill in the appropriate Node ID.

7 Device Connectivity

Purpose This chapter provides information about device connectivity.

Add LLC Node Feature

Overview A node is connected to the Low-Level Controller (LLC) through a TCP/IP socket. That socket is opened by the LLC when the node is connected.

For the LLC to connect to a node, SwitchManager needs an *AddLLCNode* message in the configuration file. The message includes a logical node ID and either one or two IP addresses, depending on the setup.

The IP address(es) for the node are the addresses of the primary CSP Matrix Series 3 card and, if applicable, the secondary CSP Matrix Series 3 card.

Functionality To connect a node, you must do the following:

- Start LLC without specifying any nodes on the command line. (The `-i` option is not valid anymore.)
- Start SwitchManager with a configuration file that specifies the nodes the LLC will connect to.

Important! With the introduction of the *SK_AddLLCNode* message, the IP addresses in the Config Switch window are obsolete. Also, Node ID 0xff (255) is no longer allowed. Each Node must be assigned a number even if in a single node configuration.

When the LLC is requested to connect to a node via the `AddLLCNode()` command, LLC will maintain that connection until told to disconnect. If the specified IP address is not reachable, LLC will continue attempting the connection forever. This condition will be logged in the LLC maintenance log file.

Example The following is an example of how to set up the message in the configuration file:

```
AddLLCNode(matrix1="10.10.55.10", matrix2="10.10.55.11",
  RequestedNode=0x08);
```

Return Values The Status field in the acknowledgment can contain the following values:

If the return value is	then...
SK_SUCCESS (0x10)	node was added successfully.

If the return value is	then...
SK_DUP_IP_ADDR (-29)	one or both IP addresses are already in use by the LLC for another logical node ID.
SK_INTERNAL_LIMIT_REACHED (-30)	no more nodes can be added to the LLC. Current limit is 128 nodes.
SK_BAD_IP_ADDRESS (-27)	no response from that IP address was received after 15 seconds.
SK_BAD_LICENSE (-28)	no valid license exists for the new node.
SK_INTERNAL_ERROR (-21)	an unexpected error occurred. Refer to the LLC maintenance.log for more details.
SK_NODE_DOESNT_EXIST (-35)	a delete command was sent for a nonexisting node.

AddLLCNode

Purpose Use *SK_AddLLCNode* at the initial configuration of your system and to dynamically add a node while LLC is running.

Description The node must already have software downloaded through TFTP. Sending *SK_AddLLCNode* for a node that needs software will fail.

If you are using the floating IP address functionality for the CSP Matrix Series 3 cards, you should use the fixed IP addresses for the CSP Matrix Series 3 cards in the AddLLCNode() message. If you don't, the LLC will inaccurately reflect the state of the switch.

Sent by SwitchManager

Type SwitchKit API message

Add the Node To add a node, you must send a configuration file with SwitchManager. The configuration file should contain at least the following information:

```
AddLLCNode (matrix1="10.10.55.10", matrix2="10.10.55.11",
             RequestedNode=0x08);
```

If you are adding a node with a single CSP Matrix Series 3 card and you use the Matrix2 argument in your message, be sure to enter the 0-length string as its value.

Arguments The following table shows the arguments you can change:

Argument	Description
Action	Specifies if a node gets added (0 = default), updated (1), or deleted (2).
Matrix1	A text field corresponding to the IP address of the primary CSP Matrix Series 3 card.
Matrix2	A text field corresponding to the IP address of the secondary CSP Matrix Series 3 card, if it exists. If there is only one CSP Matrix Series 3 card in the node, this field must be set to "", that is, the 0-length string.

Argument	Description
NodeType	Needs to be specified only for ISDN and H.323 device server nodes. In those cases, NodeType should be: <i>SK_LSS_DS_SYSTEM</i> or 0x10. Otherwise, it will default to the appropriate NodeType.
RequestedNode	<p>RequestedNode is the logical node ID requested to be assigned to the new node. The API will attempt to assign this node:</p> <ul style="list-style-type: none"> • If there is no node ID currently on the node; • If it is currently assigned a different node ID. <p>It is not a guarantee that this node will be assigned. In particular, the node ID might already be in use on the ring. If so, AddLLCNode will fail, and no connection will be established with the node. If the LLC is already connected to a node with the specified logical node ID, the AddLLCNode is considered an update, if treated accordingly.</p>

Configuration

```
AddLLCNode (
    Node = integer,
    ConnectionID = integer,
    Matrix1 = string,
    Matrix2 = string,
    RequestedNode = integer,
    Action = integer);
```

C Structure

```
typedef struct {
    char Matrix1[30];
    char Matrix2[30];
    int RequestedNode;
    //int PhysicalNode; reserved for internal use only
    //unsigned short NodeType; reserved
    //unsigned short CallControlNode; reserved
    UBYTE Action;
} SK_AddLLCNode;
```

C Structure Response

```
typedef struct {
    int Status;
    int ActualNode;
    UBYTE DownloadStatus;
} SK_AddLLCNodeAck;
```

C++ Class

```
class SKC_AddLLCNode : public SKC_ToolkitMessage {
public:
```



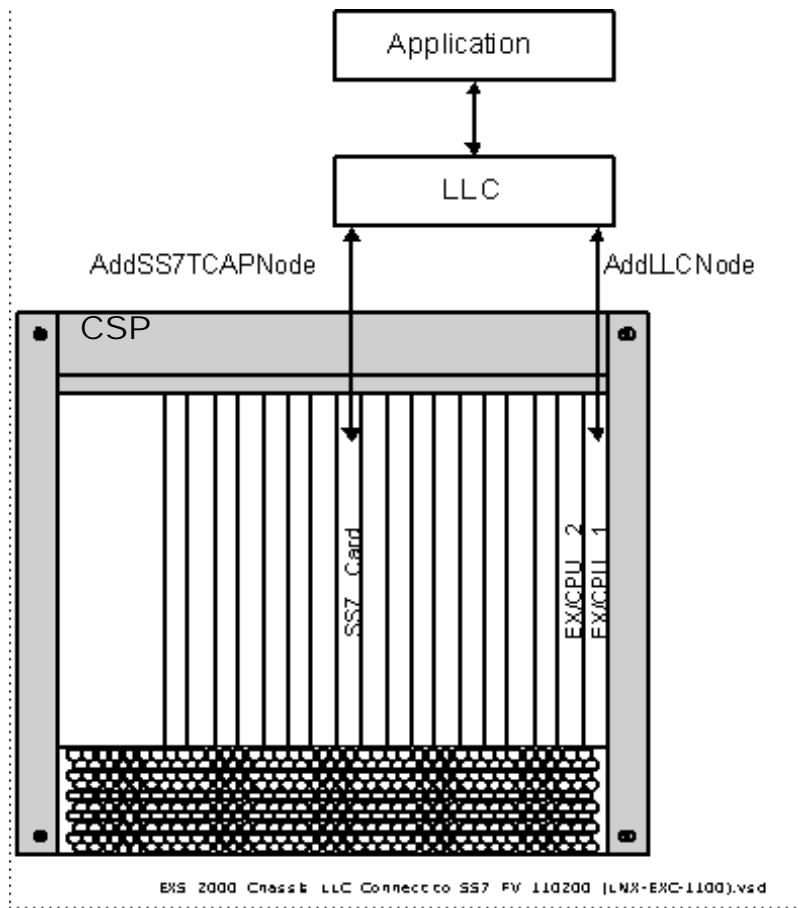
```
const char *getMatrix1() const;
void setMatrix1(const char *x);
const char *getMatrix2() const;
void setMatrix2(const char *x);
int getRequestedNode() const;
void setRequestedNode(int x);
//int getPhysicalNode() const; reserved
//void setPhysicalNode(int x); reserved
//unsigned short getNodeType() const; reserved
//void setNodeType(unsigned short x); reserved
//unsigned short getCallControlNode() const; res.
//void setCallControlNode(unsigned short x); res.
UBYTE getAction() const;
void setAction(UBYTE x);
};
```

C++ Class Response

```
class SKC_AddLLCNodeAck : public SKC_ToolkitAck {
public:
    int getStatus() const;
    void setStatus(int x);
    int getActualNode() const;
    void setActualNode(int x);
    UBYTE getDownloadStatus() const;
    void setDownloadStatus(UBYTE x);
};
```

Connecting LLC to an SS7 card

Description A user application can connect through the LLC directly to an SS7 card in the node, enabling faster TCAP messaging. See the illustration below:



How to Connect The configuration steps should be done through a configuration file sent via SwitchManager. Once established, the TCAP connection status can be monitored by an application. This can be done by registering for and utilizing the data contained within *GenericLLCReports*.

The next table lists the steps for the configuration:

Step	Action
1	Set up the connection between the LLC and CSP Matrix Series 3 card in the node using the <i>SK_AddLLCNode()</i> configuration message.
2	Set up the connection between the LLC and the SS7 card in the node using the <i>SK_AddSS7TCAPCard</i> configuration message. Successive <i>SK_AddSS7TCAPCard</i> messages must be sent for each Stack/SSN pair that you wish to route to the SS7 card.
3	<p>Send down two <i>XL_SS7SCCPTCAPConfig</i> messages. For both messages, set the ConfigType field to SCCP/TCAP host configuration to 0x08.</p> <p>For the first message set the data as follows: Data [0] Sub-system Number Data [1] Option: 0x02 Delete a host Data [2] Local/Matrix host: 0xFF Matrix host Data [3] Host Port ID: 0x00 Port 0x3142</p> <p>For the second message set the data as follows: Data [0] Sub-system Number Data [1] Option: 0x01 Add a host Data [2] Local/Matrix host: 0xFE Local host Data [3] Host Port ID: 0x00 Port 0x3142</p>
4	Applications that wish to receive <i>PPLEventIndication</i> messages related to the TCAP connection must register by calling sk_pplTCAPMessageRegister() . This function sends the <i>SK_TCAPMessageRegister</i> message. On registering for PPL event indications, the user application must be set up to handle the <i>SK_TCAPMessageRegisterAck</i> message. See the Handler functions in the <i>SwitchKit Programmer's Guide</i> .

Supported Messages

The connection from the LLC to the CSP Matrix Series 3 card remains intact. Therefore, an application must still process all messages that it normally processes. The LLC to SS7 card connection can handle only the following messages if registered for them, by calling **sk_pplTCAPMessageRegister()**:

- *XL_PPLEventIndication* - contains TCAP messaging information from the switch.
- *XL_PPLEventRequest* - sends TCAP requests into the switch.

- *XL_PollMessage* - The system type is 0x80. This message can also be used to determine which SS7 card is active and which is in standby.

Return Values The Status field can contain the following values:

If the return value is	then...
0x10	a node was added successfully.
SK_DUP_IP_ADDR	one or both IP addresses are already in use by the LLC. Notes: Only a string comparison is done, so, for example, if “localhost” and “127.0.0.1” were given, no conflict would be detected.
SK_INTERNAL_LIMIT_REACHED	no more nodes can be added to LLC. Current limit is 128 nodes.
SK_BAD_IP_ADDRESS	no response from that IP address was received after 15 seconds.
SK_BAD_LICENSE	no valid license exists for the new node.
SK_INTERNAL_ERROR	an unexpected error occurred. Refer to the LLC <code>maintenance.log</code> for more details.
SK_NODE_DOESNT_EXIST	a delete command was sent for a non-existing node.

TC_U_Abort TC_U_ABORT messages are sent by the LLC when an application drops and has outstanding dialogues from a TCAP transaction. Then the LLC sends one *PPLEventRequest* (TC_U_ABORT) to the CSP for each of them. Next, the LLC puts the LLC's Dialogue object into the NOT_ALLOCATED state after receiving an ACK, NACK, or timeout from the request.

AddSS7TCAPCard

Type EXS SwitchKit API message

Overview The *AddSS7TCAPCard* message is used to make a direct TCP connection between the LLC Host and a SS7 card for the purpose of routing TCAP traffic on a separate link. This message creates the appropriate AddLLCCard messages and sends them to the LLC.

Sent by Host, using Converged Services Administrator (CSA), webCSA, or SwitchManager configuration file.

Arguments The following table shows the arguments that you can modify:

Argument	Description
CardIP1	The primary SS7 card IP address in the Excel platform.
CardIP2	The standby SS7 card IP address, if it exists. If there is only one SS7 card in the node, this field should be set to "", that is, the 0-length string.
StackID	The ID of the TCAP stack to which the LLC connects.
SSN	The Subsystem Number configured on the SS7 card to which the LLC should connect.
ControlNode	The logical node ID assigned to the Matrix Controller which controls the chassis where the SS7 card resides.
Action	0x00 To add a connection 0x01 To remove a connection and all RoutingIDs

C Structure

```
typedef struct {
    BaseFields Base;
    char CardIP1[30];
    char CardIP2[30];
    UBYTE StackID;
    UBYTE SSN;
    int ControlNode;
    UBYTE Action;
    UBYTE reserved84[5];
} SK_AddSS7TCAPCard;
```

C Structure Response None.

C++ Class class *SKC_AddSS7TCAPCard* : public SKC_DummyMessage {
public:
 SKC_AddSS7TCAPCard(int sz = 0);
 ~SKC_AddSS7TCAPCard();
 SK_DECLARE_CLASS(SKC_AddSS7TCAPCard,SKC_DummyMessage)

 virtual MsgStruct *getStructPtr();
 virtual const MsgStruct *getStructPtr() const;
 virtual int getTag() const;

 const char *getCardIP1() const;
 void setCardIP1(const char *x);
 const char *getCardIP2() const;
 void setCardIP2(const char *x);
 UBYTE getStackID() const;
 void setStackID(UBYTE x);
 UBYTE getSSN() const;
 void setSSN(UBYTE x);
 int getControlNode() const;
 void setControlNode(int x);
 UBYTE getAction() const;
 void setAction(UBYTE x);

C++ Class Response None.

TCAPMessageRegister

Description Use this message to register an application for all TCAP-related PPL Event Indications given a specific Origination Point Code (OPC), Subsystem Number(SSN) or stack and SSN.

Important! Only events of Component Type 0x70 (TCAP TUSI) and 0x67 (SCCP SUSI) will be sent to the registered application.

Sent by Function `sk_pplTCAPRegister()`

Arguments The following table shows the arguments you can change:

Argument	Description
OPC	The Origination Point Code (Stack ID) of the SS7 TCAP stack generating the PPL Event Indications.
Stack	Stack value
Stack_SSN	a combination of Stack and SSN created by using the <code>getSS7TCAP_MultiCard_RoutingID(Stack, SSN)</code> macro
RegisterType	Registration type. Use the following constants to specify either OPC, SSN or Stack/SSN combination: SK_TCAP_REG_OPC (0x00) - OPC Registration SK_TCAP_REG_SSN (0x01) - SSN Registration SK_TCAP_REG_STACK_SSN 0x02 - Registers a combination of stack and subsystem number using the <code>getSS7TCAP_MultiCard_RoutingID(stack,SSN)</code> macro.
RegisterFlag	Indicates a registration or de-registration using the following constants: SK_TCAP_ACTIVATE_REGISTER (0x01) -Register SK_TCAP_DEACTIVATE_REGISTER (0x00) -Deregister

```

C Structure      typedef struct {
                    int OPC;
                    int STACK_SSN;
                    int SSN;
                    UBYTE RegisterType;
                    UBYTE RegisterFlag;
                    UBYTE reserved31[2];
                    } SK_TCAPMessageRegister;

C Structure Response  typedef struct {
                    int Status;
                    } SK_TCAPMessageRegisterAck;

C++ Class        class SKC_TCAPMessageRegister : public SKC_ToolkitMessage
                    {
                    public:
                    int getOPC() const;
                    void setOPC(int x);
                    int getSTACK_SSN() const ;
                    void setSTACK_SSN(int x) ;
                    int getSSN() const ;
                    void setSSN(int x) ;
                    UBYTE getRegisterType() const ;
                    void setRegisterType(UBYTE x) ;
                    UBYTE getRegisterFlag() const ;
                    void setRegisterFlag(UBYTE x)
                    };

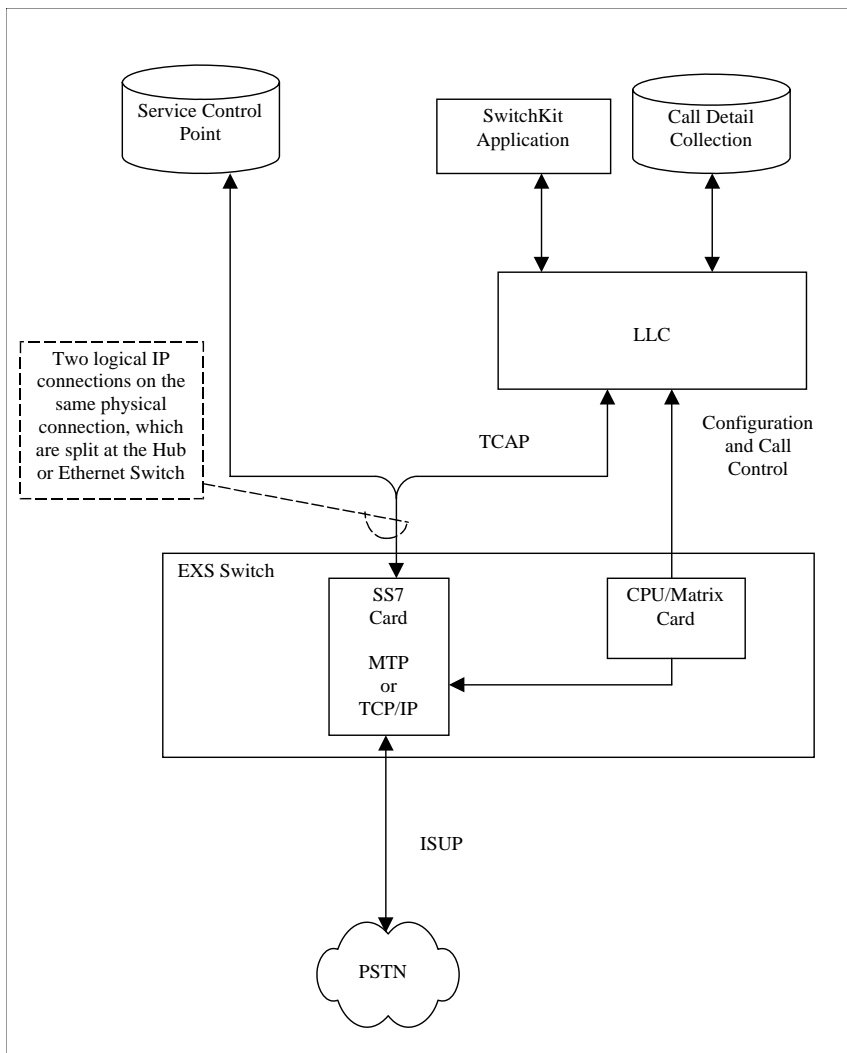
C++ Class Response  class SKC_TCAPMessageRegisterAck : public SKC_ToolkitAck
                    {
                    public:
                    int getStatus() const;
                    void setStatus(int x);
                    };

```


TCAP Routing Feature

- Overview** The LLC can connect to a TCAP-enabled node in two ways:
- Through the Matrix socket (Matrix host option using MTP)
 - Directly to the SS7 card I/O (local host option using TCP/IP)

Figure 7-1 Local Host Option



TCAP Routing provides host support for the local host option. Messages can be routed from an LLC's client application to a card level device contained within a node. Components, such as SS7 TCAP sockets, do not have unique logical node IDs. They share a node ID with a pre-assigned device and are uniquely identified by some other

user-defined value. TCAP Routing provides LLC socket connectivity to those SS7 components. For example, messages sent to SS7 TCAP sockets will contain the same node ID as the CSP Matrix Series 3 card managing the chassis, but will be routed directly to the SS7 socket. That socket is uniquely identified by a RoutingID that is derived internally from a stack ID and an SSN for a redundant SS7 card pair.

TCAP Routing Supports SwitchKit Applications

TCAP Routing provides support for SwitchKit applications connected to the LLC to register for TCAP messages based on an originating point code (OPC), sub-system number (SSN) or Stack/SSN pair. When a PPL Event Indication message enters the LLC from the network, and the component ID is TCAP-specific (0x70), the message is routed to the "least busy" of the registered applications. For more information on how inbound TCAP routing is accomplished, see *LLC: Application Load Balancing for TCAP (7-18)*. For more information on registering your applications, see `sk_pplTCAPRegister()` / `sk_pplTCAPUnRegister()` / `...OnConnection()` functions in the *SwitchKit Programmer's Guide, Register Functions* .

Operating System Requirements

- Windows NT®
- Windows® 2000
- Red Hat Linux, version 8.0
- HP-UX, version 11.0
- Solaris SPARC, version 8

Options supported

- Routes messages at the card level without the logical node being specified by the application, based on a defined routing tag that is unique to the system (such as SS7 stack ID).
- Supports redundant card pairs of the same card type..
- Provides a means for registering applications for TCAP specific messages.

Sub-component Router

The sub-component router is a module within the LLC process that allows messages to be routed from an LLC's client application to a card-level device contained within a node. That card is controlled by a CSP Matrix Series 3 card that has an assigned node ID. In other words, the card is controlled by that node. The generic requirements for sub-component routing are as follows:

- The device must have a defined type which will be referred to as CardType.

- The SS7 card must be controlled by a CSP Matrix Series 3 card that has been previously assigned a logical node and is already known by the LLC. This logical node is referred to as the `ControlNode`.
- RoutingIDs are used to address a pair of card-level devices to ensure uniqueness across the system. Application developers should obtain RoutingIDs by using the defined macros provided through the SwitchKit API. See *RoutingID Macros* in the *AddLLCCard* message.
- A logical node can contain several devices of a particular `CardType`, but can only contain one logical link for each set of devices addressed by a particular `RoutingID`.
- Additional routing IDs can be added to an individual logical link by sending additional *AddLLCCard* messages.

Routing Outbound TCAP Messages

When sending an outbound message to a particular sub-component device, there is a subset of the SwitchKit API messages that the LLC routes or attempts to route. This subset is distinguished from other message by the `CardType`. The key concept here is: for the LLC to extrapolate the correct socket address, it must walk through the `OutboundMessage` and retrieve the predetermined unique `RoutingID`. It does this when the node ID of the message is set to `0xff` (unassigned) and a TCAP socket has been added to the LLC for that `CardType` using an *AddLLCCard* message. For Outbound routing to be setup within the LLC the following are required:

- The node must be added to the LLC using the *AddLLCNode* message.
- The *AddLLCCard* message must be sent, one per `RoutingID`.
- The SS7 card must be properly configured using the API message, `SS7 SCCP/TCAP Configure 0x0077`. The LLC requires that the stack be assigned and the Local Host (`0xFE`) or Matrix Host (`0xFF`) option be specified. Although the SS7 card defaults to the Matrix Host (`0xFF`) option, the LLC does not. You must specify one or the other. It is recommended that you delete the opposing option before adding the new.

Routing Inbound TCAP Messages

The following is required when attempting to register a SwitchKit application for TCAP specific messages:

- The application must register for TCAP messages using one of two SwitchKit API functions:

`sk_pplTCAPRegister`

`sk_pplTCAPRegisterOnConnection`

Using SwitchManager

When using SwitchManager for configuration, you should use the *AddSS7TCAPCard* message instead of the *AddLLCCard* message.

Using CSA

TCAP Routing can be enabled in CSA from the SS7 Configuration view. When TCAP is enabled CSA will add an *AddSS7TCAPCard* message for each SS7 stack configured on that card. The remaining TCAP configuration should be done using the **SS7 SCCP/TCAP configuration** menu item. This configuration should be done for each stack. See the *Converged Services Administrator User's Guide* for configuring SS7 TCAP Routing.

LLC: Application Load Balancing for TCAP

Overview This feature facilitates routing between the LLC and applications, based on the number of Inbound Initial Dialogue Primitives (IDP) each registered SwitchKit application has. The LLC is capable of performing the following:

- Monitoring dialogue states
- Providing a registration mechanism for SwitchKit applications to receive TCAP-related *PPLEventIndication* messages based on a combination of stack and Sub System Number (SSN).
- Routing TCAP-related *PPLEventIndication* messages to the registered application with the lowest number of active dialogues

How to Keep Track of Dialogue States

Once dialogue monitoring has been enabled, the LLC automatically keeps track of dialogue states. Each state can be derived from any one of the multiple TCAP primitives listed in the next table. TCAP Primitives are extracted from TCAP-related *PPLEvent* messages received by the LLC.

The following terms represent the three possible states the LLC can derive within its dialogue management system for a given dialogue ID:

- NOT_ALLOCATED - Dialogue ID is not allocated.
- ALLOCATED_A_SIDE - Dialogue ID has been allocated by an application using a *PPLEventRequest* message.
- ALLOCATED_B_SIDE - Dialogue ID has been allocated by the CSP using a *PPLEventIndication* message. The next table shows the type of primitive for each LLC state. Primitives are both requests and indications, unless shown otherwise.

Important! The next table contains the following notation.

** If a TCAP_RESTART Event is received, all Dialogues on that stack should end (enter the NOT_ALLOCATED state).

*** Detection of a Dialogue Termination TLV always results in the LLC putting that dialogue into the NOT_ALLOCATED state. The LLC does not check these events for the dialogue termination TLVs.

Event Criteria (TCAP Primitives)	APP Initiated	Network Initiated	Receive Dialogue	LLC Action
	<i>A Side</i>	<i>B Side Event</i>	<i>B Side Event</i>	
TC-BEGIN	X	X		Allocate
TC_QUERY_WITH_PERMISSION	X	X		Allocate
TC_QUERY_WITHOUT_PERMISSION	X	X		Allocate
TC_BEGIN_PRIMITIVE_SET	X	X		Allocate
TC_QUERY_WITH_PERMISSION_PRIMITIVE_SET	X	X		Allocate
TC_QUERY_WITHOUT_PERMISSION_PRIMITIVE_SET	X	X		Allocate
TC_END		X	X	Deallocate
TC_RESPONSE		X	X	Deallocate
TC_RESULT_L		X	X	Deallocate
TC_RESULT_NL		X	X	Deallocate
TC_U_ERROR		X	X	Deallocate
TC_L_CANCEL		X	X	Deallocate
TC_INVOKE_L		X	X	Deallocate
TC_INVOKE_NL		X	X	Deallocate
TC_U_REJECT		X	X	Deallocate
TC_L_REJECT		X	X	Deallocate
TC_R_REJECT		X	X	Deallocate
TC_INVOKE		X	X	Deallocate
TC_RESPONSE	X			Deallocate
TC_END	X			Deallocate
TCAP_RESTART**	X	X		Deallocate
TC_RESPONSE_PRIMITIVE_SET***				Deallocate
TC_END_PRIMITIVE_SET***				Deallocate
TC_P_ABORT***		X		Deallocate
TC_DIALOGUE_TIMEOUT***		X		Deallocate

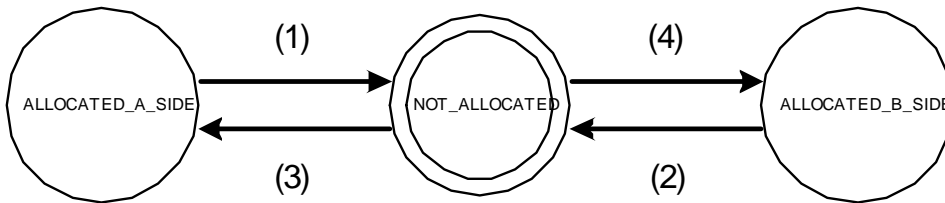
Event Criteria (TCAP Primitives)	APP Initiated	Network Initiated	Receive Dialogue	LLC Action
	<i>A Side</i>	<i>B Side Event</i>	<i>B Side Event</i>	
TC_NOTICE***		X		Deallocate
TC_TCAP_INITIATED_ABORT***		X		Deallocate
TC_U_ABORT***	X	X		Deallocate

As TCAP-related PPL events are passed in and out of the LLC, a dialogue monitor parses each PPL event and derives the current state of each dialogue using the state machine outlined in the next figure.

Diagram of Transitional States

In *Figure 7-2, Finite State Machine Used by the LLC's Dialogue Monitoring System (7-20)*, each number represents a group of transitional events (TCAP primitives) that result in a particular finite state (Dialogue State). The events (TCAP Primitives) represented by each number are listed in the table and correspond to the transition arrows in the state diagram

Figure 7-2 Finite State Machine Used by the LLC's Dialogue Monitoring System



Dialogue state and application data are then stored for maximum retrieval efficiency using a series of C++ STL containers. Application statistics can then be queried to provide load balancing information such as the number of active dialogues given an application number, or the number of active dialogues given a RoutingID. These statistics are then used by the LLC to determine Best Response Transaction Allocation.

How the LLC Determines Best Response Transaction Allocation

When a PPL event indication with component ID 0x70 (TCAP TUSI) is received by the LLC, it determines whether or not this message is an Initial Dialogue Primitive (IDP). If this is an IDP, then the stack, SSN, and dialogue are extracted from the message and are queried to

determine which registered application has the least number of active dialogues. If the numbers are equal, a random application number is chosen. Otherwise, the application number with the lowest number of active dialogues is chosen and the PPL event indication is sent to that application. If the dialogue was activated by the A side (activated by a PPL event request) then the remaining dialogues are sent to the application that activated it.

**Limitations/Assumptions/
Dependencies**

The LLC must be configured for TCAP Routing. In addition to this configuration, to enable TCAP Application Load Balancing you must do the following:

```
//set the following environment variable to activate the
//LLC's Dialogue Monitor
SK_DIALOGUE_MONITOR=1

//call this SK function from your app
sk_pplTCAPRegister(int what, UBYTE registerType);

//derive the function parameters from the following
    macros and constants
int what = getSS7TCAP_MultiCard_RoutingID(stackID, ssn);
UBYTE registerType = SK_TCAP_REG_STACK_SSN = 0x02;
```


Matrix ConfigureDS Configure : DeviceServerEx

- Purpose** Use the *SK_DeviceServerEx* message to configure the IP Signaling Series 3 card.
- Description** This message is used by the CSA to configure the IP Signaling Series 3 card. Its configuration syntax corresponds to the message fields of the *XL_DeviceServerConfigure* and *XL_IPCallServerConfigure* messages.
- Sent by** Host, via CSA configuration or SwitchManager configuration file.
- Arguments** The following table shows the arguments you can change:

Arguments	Description
ObjectType	0x0020
ID	Device Server ID
CallServerID	The LNI of the CSP Matrix Series 3 card
Slot	Slot for the Device Server
ConfigTag	ignored by SMgr
CompatFlat	any number
PollInterval	see the ServerConfig message
AllowedMissedPolls	see the ServerConfig message
Port	Device Server IP Port
PrimaryCSPort	Port for CSP Matrix Series 3 card to listen on
SecondaryCSPort	Port for CSP Matrix Series 3 card to listen on
IpAddress	IP at Device Server
PrimaryCSIpAddress	IP at CSP Matrix Series 3 card
SecondaryCSIpAddress	IP at CSP Matrix Series 3 card
DeviceType	0x01
Platform	see the DeviceServerConfig message
IPDCUsage	see the DeviceServerConfig message
DSInterfaceSupport	see the DeviceServerConfig message

Configuration *DeviceServerEx* (
 ObjectType = integer,
 Id = integer,
 CallServerId = integer,
 Slot = integer,
 ConfigTag = integer,
 CompatFlag = integer,
 PollInterval = integer,
 AllowedMissedPolls = integer,
 Port = integer,
 PrimaryCSPort = integer,
 SecondaryCSPort = integer,
 IpAddress = string,
 PrimaryCSIpAddress = string,

```
SecondaryCSIPAddress = string,  
DeviceType = integer,  
Platform = integer,  
IDDCUsage = integer,  
DSInterfaceSupport = BYTE ARRAY);
```

8 Controlled Line Card Switchover

Purpose This chapter provides information about a controlled line card switchover.

Controlled Line Card Switchover Feature

Overview SwitchManager provides the ability to configure line card redundancy. Once configured, SwitchManager will initiate a line card switchover whenever necessary.

By default SwitchManager forces the switchback from the N+1 card as soon as the failed card is available and no calls are presently active through LLC on the N+1 line card. If your call processing is switch resident, LLC will not know calls are active, and the switchback will happen immediately, potentially resulting in lost calls.

With the *SK_SwitchBackFromStandbyConfig* message the switchback from the redundant line card to the primary card becomes configurable.

The two message *SK_SwitchBackFromStandbyConfig* and *SK_SwitchBackFromStandby* allow you to perform a controlled switchover in a N+1 line card redundancy setup. In case a line card was out of service and comes back online, you can control when and how the primary card gets back into service.

Arguments The following table shows the user modifiable arguments. Note that the action argument has different tasks in the messages: *SwitchBackFromStandbyConfig* and *SwitchBackFromStandby*.

:

Argument	Description
Action	<p>In <i>SK_SwitchBackFromStandbyConfig</i> the action field can be set to:</p> <p>0 = default behavior, perform the switchback when LLC determines the card is not handling any calls.</p> <p>1 = perform switchback only after receiving the SwitchBackFromStandby message through the Converged Services Administrator (CSA) or other host application.</p> <p>In <i>SK_SwitchBackFromStandby</i> the action field can be set to:</p> <p>0 = graceful switchback after LLC determines that the standby card is not handling any calls.</p> <p>1 = forced switchback immediately.</p>
Node	Specifies the node where the switchback is supposed to happen.
StandbySlot	Specifies the standby slot of the redundant card.

Return Values

The following table shows the possible return values for the switchback messages:

If the return value is	then...
SK_SUCCESS (0x10)	the switchback was successful.
SK_SWITCHBACK_FAILED (0xf00b)	the switchback failed, refer to the alarm.log for further detail.

SwitchBackFromStandby

Overview Use the *SK_SwitchBackFromStandby* message to initiate the switchback from a standby line card to the primary. This message can only be sent if the action field of the message *SK_SwitchBackFromStandbyConfig* was set to 1 during configuration.

Sent by Application or the Converged Services Administrator (CSA)

Arguments The following table shows the arguments that you can modify:

Argument	Description
Action	In <i>SK_SwitchBackFromStandby</i> the action field can be set to: 0 = graceful switchback after LLC determines that the standby card is not handling any calls. 1 = forced switchback immediately.
StandbySlot	Specifies the slot number of the standby line card.

C Structure

```
typedef struct {
    UBYTE StandbySlot;
    UBYTE Action;
} SK_SwitchBackFromStandby;
```

C Structure Response

```
typedef struct {
    int Status;
} SK_SwitchBackFromStandbyAck;
```

C++ Class

```
class SKC_SwitchBackFromStandby : public
    SKC_ToolkitMessage {
public:
    UBYTE getStandbySlot() const;
    void setStandbySlot(UBYTE x);
    UBYTE getAction() const;
    void setAction(UBYTE x);
};
```

C++ Class Response

```
class SKC_SwitchBackFromStandbyAck : public
    SKC_ToolkitAck {
public:
    int getStatus() const;
    void setStatus(int x);
};
```

9 Server Status Change

Purpose This chapter provides information about optimizing your system's and application's performance by using the Server Status Change message instead of using Polls.

Server Status Change Feature

Overview The Server Status Change feature informs all registered client applications about any change in the switch status. The feature is enabled with the message *SK_ServerStatusChange*, managed by the LLC to inform the clients of a node's status at all times.

This feature removes the requirement for SwitchKit applications to monitor the poll messages and respond accordingly. Applications can register for the *SK_ServerStatusChange* message and base decisions on it.

Server Status Change Events The *SK_ServerStatusChange* message is created within LLC, and responds to all registered applications for any one of the following events:

- The data contained within two consecutive poll message changes (SK_PM_CHANGE).
- A socket connection is closed (SK_LINK_DOWN).
- A socket connection is opened, and the CSP indicates that is ready to communicate. (SK_LINK_UP).
- An application connects to the LLC (SK_SSC_UPDATE).
- The LLC receives a NodeStatusReport message.
- The LLC receives a response to a NodeStatusQuery.

ServerStatusChange

Type	SwitchKit API message
Description	The LLC sends the <i>ServerStatusChange</i> message containing all the information needed by an application to determine the current status of a node. The message is sent when the LLC discovers a change in CSP status.
C Structure	<pre>typedef struct { unsigned short Status; UBYTE SystemType; UBYTE MatrixSide; UBYTE MatrixState; UBYTE AdjMatrixState; UBYTE StatusBits; UBYTE Reserved1; UBYTE Reserved2; unsigned short ExcelPort; int PhysicalNode; UBYTE LogicalNode; UBYTE HostNode; UBYTE NodeStatus; UBYTE MessageTrigger; UBYTE FromPrimary; UBYTE SocketStatus; unsigned short UpdatedFieldBits; UBYTE reserved40[1]; } <i>SK_ServerStatusChange</i>;</pre>
C++ Class	<pre>class <i>SKC_ServerStatusChange</i> : public SKC_DummyMessage { public: unsigned short getStatus() const; void setStatus(unsigned short x); UBYTE getSystemType() const; void setSystemType(UBYTE x); UBYTE getMatrixSide() const; void setMatrixSide(UBYTE x); UBYTE getMatrixState() const; void setMatrixState(UBYTE x); UBYTE getAdjMatrixState() const; void setAdjMatrixState(UBYTE x); UBYTE getStatusBits() const; void setStatusBits(UBYTE x); UBYTE getReserved1() const; void setReserved1(UBYTE x); UBYTE getReserved2() const;</pre>

```

void setReserved2(UBYTE x);
unsigned short getExcelPort() const;
void setExcelPort(unsigned short x);
int getPhysicalNode() const;
void setPhysicalNode(int x);
UBYTE getLogicalNode() const;
void setLogicalNode(UBYTE x);
UBYTE getHostNode() const;
void setHostNode(UBYTE x);
UBYTE getNodeStatus() const;
void setNodeStatus(UBYTE x);
UBYTE getMessageTrigger() const;
void setMessageTrigger(UBYTE x);
UBYTE getFromPrimary() const;
void setFromPrimary(UBYTE x);
UBYTE getSocketStatus() const;
void setSocketStatus(UBYTE x);
unsigned short getUpdatedFieldBits();
void setUpdatedFieldBits(unsigned short x);
};

```

Conditions for Sending the ServerStatusChange Message

The following list shows the conditions that will cause the LLC to send a *ServerStatusChange* message:

- SK_PM_CHANGE - sent when the first poll is received by the LLC for a CSP Matrix Series 3 Card. Also sent any time a value changes in the Poll. The UpdatedFieldBits field indicates which fields have changed since the last poll. The following is a possible scenario:
 1. The adjacent matrix has lost connection to the LCC running on the host. Either the LAN connection is severed or the host has severe CPU deprivation. Diagnose the LAN connection. Determine if the matrix card can receive and return ping messages. If the LAN connection is good, check the performance of the host.
- SK_SSC_UPDATE - sent under the following conditions:
 1. When an application registers for the ServerStatusChange notification via *sk_msgRegister()*, an SSC_UPDATE is sent for each node the LLC is currently connected to which has a CSP Matrix Series 3 Card in the active state.
 2. When an application issues an *AddLLCNode*, an SSC_UPDATE may be generated if the LLC already knew about the node and the LLC is currently connected to the active CSP Matrix Series 3 Card for that node. In other words, if the Link is considered to be up.

- SK_LINK_UP - sent under the following conditions:
 1. This message is sent to indicate that there is a connection to the active CSP Matrix Series 3 Card for this node and the CSP Matrix Series 3 Card is ready to be configured. This will be sent if this is a new condition (i.e. we were not previously connected to the active CSP Matrix Series 3 Card).
 2. Sent if an application has performed an *AddLLCNode*, the LLC is currently connected to the specified node, and the LLC is connected to the active CSP Matrix Series 3 Card of that node.
 3. When an application registers for the *ServerStatusChange* notification via *sk_msgRegister()*, and the LLC is connected to the active CSP Matrix Series 3 Card of a node.
- SK_LINK_DOWN - sent under the following conditions:
 1. This message is sent to indicate that there is NO connection to the active CSP Matrix Series 3 Card for a node. This will be sent if this is a new condition (that is, we were previously connected to the active CSP Matrix Series 3 Card).
 2. Sent if an application has performed an *AddLLCNode*, and the LLC is not currently connected to the specified node's active CSP Matrix Series 3 Card.
 3. When an application registers for the *ServerStatusChange* notification via *sk_msgRegister()*, and the LLC is supposed to be connected to a node but is not connected to the node's active CSP Matrix Series 3 Card.

More on SK_LINK_UP and SK_LINK_DOWN

- Registering for *ServerStatusChange* will result in an SK_LINK_UP or SK_LINK_DOWN for each node controlled by the LLC. If an SK_LINK_UP is sent, it will be followed by an SK_SSC_UPDATE with the last poll from the active CSP Matrix Series 3 Card.
- Performing an *AddLLCNode* will result in an SK_LINK_UP or SK_LINK_DOWN for the specified node if the LLC is already controlling the node. If an SK_LINK_UP is sent, it will be followed by an SK_SSC_UPDATE with the last poll from the active CSP Matrix Series 3 Card, if the LLC is currently connected to the CSP Matrix Series 3 Card in the active state.

- SK_PM_CHANGE, SK_SOCKET_CHANGE, SK_LINK_UP and SK_LINK_DOWN are generated to indicate a change in state.
- SK_NSQ_CHANGE or SK_NSR_CHANGE are generated upon arrival of *NodeStatusQueryAck* or *NodeStatusReport* from the CSP.

Arguments The following table indicates what arguments are valid for a specific message trigger:

MessageTrigger	Argument	Description
SK_PM_CHANGE or SK_SSC_UPDATE	PhysicalNode	Represents the requested Node ID
	LogicalNode	Represents the current Node ID as reported by the CSP in <i>PollMessage</i>
	SocketStatus	1 always. Socket must be connected for this indication to occur.
	Status	Status from <i>PollMessage</i>
	SystemType	System Type(Poll Source) from <i>PollMessage</i>
	MatrixSide	CSP Matrix Series 3 Card ID from <i>PollMessage</i>
	MatrixState	Matrix State(Card State) from <i>PollMessage</i>
	AdjMatrixState	Adjacent Card State from <i>PollMessage</i>
	StatusBits	Card Status from <i>PollMessage</i>
	CSP Port	Double Byte after Multi Host Connection Status Byte 2 in <i>PollMessage</i>
	UpdatedFieldBits	Bit mask indicating which fields in the poll were updated. See table below for details of the meaning of the bits. Only set for SK_PM_CHANGE
SK_LINK_UP or SK_LINK_DOWN	PhysicalNode	Requested Node ID
	LogicalNode	Requested Node ID

Constants The following table contains information for the field UpdatedFieldBits. You can combine any number of settings in this bit mask:

Constants	Changed Argument in <i>Server StatusChange</i> message
SK_STATUS_CHANGE	Status
SK_SYSTEMTYPE_CHANGE	SystemType
SK_MATRIXSIDE_CHANGE	MatrixSide
SK_MATRIXSTATE_CHANGE	MatrixState
SK_ADJMATRIXSTATE_CHANGE	AdjMatrixState
SK_STATUSBITS_CHANGE	StatusBits
SK_LOGICALNODE_CHANGE	LogicalNode

10 Configuration Tutorial

Purpose This chapter provides a configuration tutorial.

Tandem Configuration

Overview The EXS SwitchKit installation CD provides several configuration sample files. This section uses the **tandem.cfg** configuration file to explain the set up. To run this file, you must have the T1 card in Slot 1 and the card must loop back. The DSP card must be in Slot 0. If your EX's hardware configuration is different, you must change the slot specifications in the configuration file.

File Setup **AddLLCNode**

With respect to the node connectivity feature of EXS SwitchKit, the first message in a configuration file should be *AddLLCNode*. The parameters you need to specify are RequestedNode, Matrix1, and Matrix2. If your system has only one CSP Matrix Series 3 card, you still need to list the Matrix2 parameter, but the value will be an empty quoted string.

AllInService / AllOutOfService

You send *AllInService* to bring all switch entities in service on completion of the specified configuration. *AllOutOfService* brings all switch entities out of service prior to implementing the specified configuration.

ClearAllChanGroups

With *ClearAllChanGroups* you remove all channels from all logical channel groups.

AssignSpan

With *AssignSpan* you map the physical spans with the logical span ID.

AssociateChanGroup

When the spans are assigned, you use the *AssociateChanGroup* message to define the name of a logical channel group and to associate a range of channels to that logical channel group.

AnswerSuperviseConfig

With *AnswerSuperviseConfig* you configure answer supervision mode for a range of channels.

TrunkTypeConfig

Use the *TrunkTypeConfig* message to configure the trunk interface signaling for the specified range of channels (start dial type, filters, timers, etc. are set to defaults).

LocalReleaseConfig

With *LocalReleaseConfig* you configure the specified range of channels for a local end release mode of park or release.

DistantReleaseConfig

With *DistantReleaseConfig* you configure the specified range of channels for a distant end release mode of park or release.

LoopTimingConfig

With *LoopTimingConfig*, you set or clear loop timing sources.

InseizeInstrListConfig

The *InseizeInstrListConfig* message configures a list of inseize control instructions for later real time call processing. You can configure one control instruction per message. If you need to configure four control instructions, you must send four messages, as shown in our sample configuration file.

OutseizeInstrListConfig

Use the *OutseizeInstrListConfig* message to configure a list of outseize control instructions. As with *InseizeInstrListConfig* you must send one message for every control instruction.

InpulsingParametersConfig

The *InpulsingParametersConfig* message configures impulse data collection parameters for the specified range of channels.

DSPSIMMConfig

With the *DSPSIMMConfig* you configure the DSP SIMMs for the individual DSP functions as specified.

A Appendix

Purpose This section includes answers to questions that are commonly asked by Excel customers.

Frequently Asked Questions

Installation and Start-Up Questions

Where do I install the software?

For UNIX install (default):

- SwitchKit = installed in /usr/local/switchkit
- License = /usr/local/switchkit/sk_license.dat
- Symbolic link to the system software location = /usr/local/switchkit/system_software.bin

For Windows NT® install:

- SwitchKit = C:\ProgramFiles\Excel Switching Corporation\SwitchKit
- Licenses = C:\Program Files\Excel Switching Corporation\SwitchKit\sk_license.dat
- Software = C:\Program Files\Excel Switching Corporation\SwitchKit\System-software.bin

How can I modify the default behavior of the SwitchKit modules?

The behavior of the components of the SwitchKit can be modified by using a system-wide defaults file. This file is /usr/local/switchkit/Defaults or C:\Program Files\Excel Switching Corporation\SwitchKit\Defaults. Entries in the file are of the form <fieldName> : value.

fieldName is not case sensitive. Comments begin with the pound sign (#). Blank lines are allowed. The valid fieldNames are compress_polls, llc_host, system_software_file, and password.

In what order should I start the modules?

You must start the LLC before any other modules are run. We suggest you start the SwitchManager before starting any applications.

Why won't the LLC run?

If the LLC fails to run, it most likely means that it could not connect to the switch, or did not find a valid license file. Also, make sure that you have permission to write to the directory it started from, so that it can

create and write to its log files. Pay careful attention to the error messages it outputs to standard out as well as looking at the maintenance.log file.

Why won't the SwitchManager run?

If the SwitchManager fails to run, it most likely means that it could not connect to the LLC, or was unable to parse the configuration file it was given as input. Pay careful attention to the error messages it outputs to standard out, and to the SwitchManager maintenance.log file.

Do I have to run the SwitchManager?

It is not necessary to run the SwitchManager module in order to run applications, but the SwitchManager is responsible for configuring the switch, maintaining the switch, and handling alarms and other fault conditions. Therefore, it is advisable that the SwitchManager always be running in order to handle these conditions. For more information, refer to Running SwitchKit Components.

What should I do when SwitchManager shows a NACK for AssignSpan?

SwitchManager prints the following when you have not downloaded a system software license to your CSP:

```

**Critical: *****
**Critical: *                               *
**Critical: *   NO SYSTEM SOFTWARE LICENSE ON   *
**Critical: *   NODE       :0x1                 *
**Critical: *   MATRIX1:10.10.67.15            *
**Critical: *   MATRIX2:none                   *
**Critical: *   PLEASE DOWNLOAD ...           *
**Critical: *                               *
**Critical: *****

```

A system software license is required to unlock the CSP software. For more information, see the *CSP Developer's Guide: Overview, DSP Resource Points (9-95)*.

In a configuration file, does the order of configuration messages matter?

No. SwitchManager sorts the messages and sends them in the correct order.

In a message, does the order of parameters matter?

No. But spelling and the use of commas to separate the parameters in a message is important.

What are the advantages and disadvantages of running Switchkit (LLC & SwitchManager) as a Windows NT® service?

The advantages of running SwitchKit as a service:

- Services can be configured to start automatically when Windows NT® is rebooted. This saves time not having to have someone manually start the service. A Windows NT® utility in the control panel called TweakUI allows the user the option to automatically log in to NT.
- DOS windows take additional memory and other system resources that can affect performance. Running Switchkit in a DOS window prints the LLC Maintenance & SwitchMgr Alarm logs to the screen. This also takes additional memory and affects system performance to print each message.
- In SwitchKit 8.0 Service Icon's have been added to the system tray so the user can quickly see the status of LLC & Switchmgr. These icons identify the service as Active/Primary or Standby/Secondary. There are also options on the service to open the log files. These icons also change color to indicate the current state.

The only disadvantage is in releases, Switchkit 5.6 and below, the service icons are not available so there is no easy way to identify whether LLC & SwitchMgr are running or not.

Messages and Logging Where are the log files?

By default, the logs are written to the current directory of the application. To change this, set SK_LOG_DIR.

What messages are sent to the switch and what messages are sent to the host?

The messages.log file is the place to look. The LLC will normally log all messages passing to the switch in the messages.log file. Use this file to see precisely what messages are being sent to the switch, and how the switch is responding.

Can I stop the LLC from writing to the messages .log file on my production system?

You can set the LLC to log everything or to log nothing.

The time stamp in the SwitchKit .log files is not correct. What causes this?

LLC uses a system call to obtain the timestamp for logging. If the system call returns the incorrect information, the timestamps in the .log files will be inaccurate. This can happen when the system time is changed on a machine.

Configuration**How can I tell the result of the configuration given to the SwitchManager?**

The SwitchManager will notify you of the status of every configuration message. It will notify when messages are successful, and when they fail (with the reason why), and it will notify when messages are not sent (e.g. if the board the message is for is not present). Pay careful attention to the alarm.log generated by the SwitchManager to understand the status of the configuration.

I want a channel group with just channels 3 and 7. How can I build non-contiguous channel groups?

AssociateChannelGroup messages are cumulative. Just issue two *AssociateChannelGroup* messages, one with just channel 3, and another with just channel 7.

Should I set the Poll Interval for the switch, and, if so, what should I set it to?

Normally, you should not have to set the Poll Interval. The LLC automatically sets it to the default, 2 seconds, upon start-up. If you do set the Poll Interval yourself, consider the value carefully. The LLC needs to check a variety of things periodically, such as dropped messages by the switch, loss of switch connection, etc. It is recommended that the poll interval be between 1 and 5 seconds. Turning off polls will disable some of the functionality of the SwitchKit, and is strongly discouraged.

Where are some sample configuration files and source code?

For UNIX, the source code is located in /usr/local/switchkit/samples/src and the configuration files are located in /usr/local/switchkit/samples/cfg

For NT, the source code is located in C:\Program Files\EXCEL\SwitchKit\samples and the configuration files are located in C:\Program Files\EXCEL\SwitchKit\samples\cfg

There are various sample applications for both C programmers (*.c) and C++ programmers (*.cpp). SimpleTandem and CallSim are the example applications described in the documentation. A sample configuration file for these applications and for a specific hardware and network configuration is tandem.cfg.

I do an `sk_requestOutseizedChannel` on a channel group that I've configured in my config file, but it says `SK_NO_CHANNELS`. Why?

The LLC allocates channels that are in service and are not in any kind of alarm state, such as red or yellow alarm, and that have not already been allocated. For all available channels, the maintenance.log file of the LLC should show “Channel <offset> has come up”. Also, make sure that the channel group has been configured properly.

Error Messages Where can I find SwitchKit status codes and error responses?

The Windows NT® default path to status codes and error responses is:

```
C:\Program Files\Excel Switching
  Corporation\SwitchKit\include\SK_API.h
```

The UNIX default path to status codes and error responses is:

```
/usr/local/switchkit/include/SK_API.h
```

LAN How do I distribute the SwitchKit modules and my call processing applications across a local or wide area network?

To run modules on separate host machines from the LLC, the modules must be told where the LLC is running and must be able to connect to that machine over the network. The location of the machine is specified through the environment variable `SK_LLC_HOST`, or through the `llc_host` field in the system-wide defaults file as mentioned above. The value of this variable can be either a host name, or an IP address in dot notation.

Is this secure? Can anyone else connect to my LLC?

If you want secure connections to the LLC, set the `SK_PASSWORD` environment variable, or the password field in the system-wide defaults file on both the machine that is running the LLC and the machine that is running the module that will connect to the LLC. Only by knowing the value of that password will anyone be able to connect to the LLC.

Applications **How do I load share a single application across multiple machines?**

Run the application on all the machines you wish to load share it across, and the load sharing will be accomplished automatically. Because the application will always be registering for the same inbound channel groups, the LLC will evenly distribute incoming calls among the various instances of the application. For more information, refer to the SwitchKit Programmer's Guide and the Redundant Application Feature.

How does an application establish a connection with the LLC?

Refer to the SwitchKit Programmer's Guide, Connection Management Functions. The SwitchKit API offers two connection models to set up a connect between your applications and the LLC.

When I first start my application, channels are allocated correctly, but after a while, I start getting SK_NO_CHANNELS error messages?

Make sure that any channel that is assigned to the application, either from an inbound call or via `sk_requestOutseizedChannel`, is returned to the LLC via `sk_returnChannel`, when the application is finished with it. Just like a dynamic memory leak, any channels that are not returned to the LLC will not be available to be allocated. It is important to realize that you must return not only the channels that you explicitly allocate via `sk_requestOutseizedChannel`, but also the channels that the LLC is implicitly allocating to you on inbound calls because of `sk_watchChannelGroup`.

I'm trying to send a message to the switch, but my application doesn't get a reply. In fact, there isn't even any mention of it in messages.log. What happened?

If a message that is supposed to be sent to the switch does not appear in messages.log, the most likely explanation is that your application was not able to send it to the LLC correctly. Check the return value from your call to the appropriate send function.

Why does any application that I compile, when run, give an error message about static initializers not being called?

All applications using the SwitchKit libraries must be linked with a C++-aware linker. In particular, the static initializers in the SwitchKit library must be called, and only a C++-aware linker knows how to call them. Whether your C compiler can handle static initializers depends on your system and the compiler. For example, cc on SCO UNIX will work fine, whereas cc on UNIXWare will not. GCC will work on all platforms, and is recommended.

CSA How can I save custom configurations without the CSA view resetting every time we restart?

By default when you connect to an LLC, CSA opens an event view for that LLC. The default filter is all events selected, except for PPL events. CSA stores all event views and filters in CSA's default file. On restart (assuming **Open with last setting** is checked), CSA will re-open all the event views with the corresponding filters.

What file does the output of the event viewer dump to?

The messages seen in the event viewer are stored in memory. Therefore, the event viewer does not move messages to other files. CSA currently stores a maximum of 2500 messages in memory. When new messages come in, CSA begins to delete old messages. This is done to limit CSA's memory usage. All messages, even those that CSA or event viewer may or may not be registered to receive, are stored in the alarm.log file.

Example Configuration File

Overview The following configuration file provides example code which can be used to configure a CSP.

Remarks The remarks or lines of code beginning with a pound sign (#) were added to this file after it was generated. The limitations of this page present the code with spaces or carriage returns that you would not see in a Command Prompt window. These spaces or carriage returns do not have any impact on the configuration.

```
example.cfg # SSAInternal Messages for use with CSA configuration
mode only.
SSAInternalMsg(Node_TypeTable(NodeType=1,requestedNode=1)
;
SSAInternalMsg(SSACardTable(Node=1,32=115,18=240,17=240,1
=21,4=121,5=96,9=24,10=118,11=113,0=25);

AllInService();
AllOutOfService();

AddLLCNode(Node=1,Matrix1="208.209.43.52",Matrix2="",Requ
estedNode=0x01,PhysicalNode=0xffffffff,NodeType=0x00,C
allControlNode=0x01,Action=0x00);

ProductLicenseDownload(Node=1,ICBCCount=0x01,ICBData=2:36:
16:70:70:66:74:72:65:82:86:90:66:65:76:76:50:48:75);

# De-assign all spans
AssignSpan(Node=1,Slot=0xff,Span=0xffff,Offset=0xff);
# E1 Spans Slot 0
AssignSpan(Node=1,Span=30,Slot=0,Offset=0);
AssignSpan(Node=1,Span=31,Slot=0,Offset=1);
AssignSpan(Node=1,Span=32,Slot=0,Offset=2);
AssignSpan(Node=1,Span=33,Slot=0,Offset=3);
AssignSpan(Node=1,Span=34,Slot=0,Offset=4);
AssignSpan(Node=1,Span=35,Slot=0,Offset=5);
AssignSpan(Node=1,Span=36,Slot=0,Offset=6);
AssignSpan(Node=1,Span=37,Slot=0,Offset=7);
# T1 Spans Slot 9
AssignSpan(Node=1,Span=0,Slot=9,Offset=0x0);
AssignSpan(Node=1,Span=1,Slot=9,Offset=0x1);
AssignSpan(Node=1,Span=2,Slot=9,Offset=0x2);
AssignSpan(Node=1,Span=3,Slot=9,Offset=0x3);
AssignSpan(Node=1,Span=4,Slot=9,Offset=0x4);
AssignSpan(Node=1,Span=5,Slot=9,Offset=0x5);
AssignSpan(Node=1,Span=6,Slot=9,Offset=0x6);
```

```
AssignSpan(Node=1,Span=7,Slot=9,Offset=0x7);
AssignSpan(Node=1,Span=8,Slot=9,Offset=0x8);
AssignSpan(Node=1,Span=9,Slot=9,Offset=0x9);
# VDAC Spans Slot 5
AssignSpan(Node=1,Span=16,Slot=5,Offset=0x0);
AssignSpan(Node=1,Span=17,Slot=5,Offset=0x1);
AssignSpan(Node=1,Span=18,Slot=5,Offset=0x2);
AssignSpan(Node=1,Span=19,Slot=5,Offset=0x3);
AssignSpan(Node=1,Span=20,Slot=5,Offset=0x4);

# Loop Timing. Type (Primary = 1, Secondary = 2)
LoopTimingConfig(Node=1, Slot=0, Offset=0, Type=1,
  Action=1);
LoopTimingConfig(Node=1, Slot=9, Offset=0, Type=2,
  Action=1);

# T1 Span format 0x52 ESF, B8ZS, Clear Channel
T1SpanConfig(Node=1,Span=0x00,Format1=0x52,Format2=0x06);
T1SpanConfig(Node=1,Span=0x01,Format1=0x52,Format2=0x06);
T1SpanConfig(Node=1,Span=0x02,Format1=0x52,Format2=0x06);
T1SpanConfig(Node=1,Span=0x03,Format1=0x52,Format2=0x06);
T1SpanConfig(Node=1,Span=0x08,Format1=0x52,Format2=0x01);

# T1 Span format 0x09 D4, AMI, CAS
T1SpanConfig(Node=1,Span=0x04,Format1=0x09,Format2=0x06);
T1SpanConfig(Node=1,Span=0x05,Format1=0x09,Format2=0x06);
T1SpanConfig(Node=1,Span=0x06,Format1=0x09,Format2=0x06);
T1SpanConfig(Node=1,Span=0x07,Format1=0x09,Format2=0x06);
T1SpanConfig(Node=1,Span=0x09,Format1=0x09,Format2=0x06);

# E1 Span format HDB3, Clear Channel
E1SpanConfig(Node=1,Span=30,SARegisters=248,TS16Registers
  =208,Format=9,LineLength=4);
E1SpanConfig(Node=1,Span=31,SARegisters=248,TS16Registers
  =208,Format=9,LineLength=4);
E1SpanConfig(Node=1,Span=32,SARegisters=248,TS16Registers
  =208,Format=9,LineLength=4);
E1SpanConfig(Node=1,Span=33,SARegisters=248,TS16Registers
  =208,Format=9,LineLength=4);

# E1 Span format AMI, CAS
E1SpanConfig(Node=1,Span=34,SARegisters=248,TS16Registers
  =208,Format=0,LineLength=4);
E1SpanConfig(Node=1,Span=35,SARegisters=248,TS16Registers
  =208,Format=0,LineLength=4);
E1SpanConfig(Node=1,Span=36,SARegisters=248,TS16Registers
  =208,Format=0,LineLength=4);
E1SpanConfig(Node=1,Span=37,SARegisters=248,TS16Registers
  =208,Format=0,LineLength=4);

# SS7 ANSI Stack 0 looped back to Stack 1
```

```
SS7SignalingStackConfig(Node=1,Slot=0x0b,Stack=0x00,OPC=0
x01,NumModules=0x03,Data=1:0:2:0:3:0);
SS7SignalingStackConfig(Node=1,Slot=0x0b,Stack=0x01,OPC=0
x02,NumModules=0x03,Data=1:0:2:0:3:0);

# SS7 ITU Stack 2 looped back to Stack 3
SS7SignalingStackConfig(Node=1,Slot=0x0b,Stack=0x02,OPC=0
x03,NumModules=0x03,Data=1:1:2:1:3:1);
SS7SignalingStackConfig(Node=1,Slot=0x0b,Stack=0x03,OPC=0
x04,NumModules=0x03,Data=1:1:2:1:3:1);

SS7SignalingLinkSetConfig(Node=1,StackID=0x00,LinkSetID=0
x01,APC=0x02);
SS7SignalingLinkSetConfig(Node=1,StackID=0x01,LinkSetID=0
x02,APC=0x01);
SS7SignalingLinkSetConfig(Node=1,StackID=0x02,LinkSetID=0
x03,APC=0x04);
SS7SignalingLinkSetConfig(Node=1,StackID=0x03,LinkSetID=0
x04,APC=0x03);

SS7SignalingLinkConfig(Node=1,StackID=0x00,LinkSetID=0x01
,LinkID=0x00,Span=0,Channel=0x00,SLC=0x00,DataRate=0x0
1,ElectricalInterface=0x00);
SS7SignalingLinkConfig(Node=1,StackID=0x01,LinkSetID=0x02
,LinkID=0x01,Span=1,Channel=0x00,SLC=0x00,DataRate=0x0
1,ElectricalInterface=0x00);
SS7SignalingLinkConfig(Node=1,StackID=0x02,LinkSetID=0x03
,LinkID=0x02,Span=30,Channel=0x00,SLC=0x02,DataRate=0x
01,ElectricalInterface=0x00);
SS7SignalingLinkConfig(Node=1,StackID=0x03,LinkSetID=0x04
,LinkID=0x03,Span=31,Channel=0x00,SLC=0x02,DataRate=0x
01,ElectricalInterface=0x00);

SS7SignalingRouteConfig(Node=1,StackID=0x00,Destination=0
x00,RouteID=0x00,DPC=0x02,LinkSetID=0x01,Priority=0x01
,CombinedLinkSetRef=0xffff);
SS7SignalingRouteConfig(Node=1,StackID=0x01,Destination=0
x01,RouteID=0x01,DPC=0x01,LinkSetID=0x02,Priority=0x01
,CombinedLinkSetRef=0xffff);
SS7SignalingRouteConfig(Node=1,StackID=0x02,Destination=0
x02,RouteID=0x02,DPC=0x04,LinkSetID=0x03,Priority=0x01
,CombinedLinkSetRef=0xffff);
SS7SignalingRouteConfig(Node=1,StackID=0x03,Destination=0
x03,RouteID=0x03,DPC=0x03,LinkSetID=0x04,Priority=0x01
,CombinedLinkSetRef=0xffff);

# SS7 CIC's on T1 spans
SSAInternalMsg(SS7CICConfigGroup(Node=1,StackID=0,BaseCIC
Number=0,BaseCICSpan=0,BaseCICChannel=1,NumCICs=23,Cal
lControlUserPart=0,Range=0:0-
0:0,NumFreeCICNumsInSpan=0,DPC=2);
```

```
SSAInternalMsg(SS7CICConfigGroup(Node=1,StackID=1,BaseCIC
  Number=0,BaseCICSpan=1,BaseCICChannel=1,NumCICs=23,CallControlUserPart=0,Range=0:0-
  0:0,NumFreeCICNumsInSpan=0,DPC=1);
SSAInternalMsg(SS7CICConfigGroup(Node=1,StackID=0,BaseCIC
  Number=24,BaseCICSpan=2,BaseCICChannel=0,NumCICs=24,CallControlUserPart=0,Range=0:0-
  0:0,NumFreeCICNumsInSpan=0,DPC=2);
SSAInternalMsg(SS7CICConfigGroup(Node=1,StackID=1,BaseCIC
  Number=24,BaseCICSpan=3,BaseCICChannel=0,NumCICs=24,CallControlUserPart=0,Range=0:0-
  0:0,NumFreeCICNumsInSpan=0,DPC=1);
SS7CICConfig(Node=1,StackID=0,BaseCICNumber=0,BaseCICSpan
  =0,BaseCICChannel=1,NumCICs=23,DPC=2,CallControlUserPart=0);
SS7CICConfig(Node=1,StackID=1,BaseCICNumber=0,BaseCICSpan
  =1,BaseCICChannel=1,NumCICs=23,DPC=1,CallControlUserPart=0);
SS7CICConfig(Node=1,StackID=0,BaseCICNumber=24,BaseCICSpan
  =2,BaseCICChannel=0,NumCICs=24,DPC=2,CallControlUserPart=0);
SS7CICConfig(Node=1,StackID=1,BaseCICNumber=24,BaseCICSpan
  =3,BaseCICChannel=0,NumCICs=24,DPC=1,CallControlUserPart=0);

# SS7 CIC's on E1 spans
SSAInternalMsg(SS7CICConfigGroup(Node=1,StackID=2,BaseCIC
  Number=0,BaseCICSpan=0,BaseCICChannel=1,NumCICs=23,CallControlUserPart=0,Range=0:0-
  0:0,NumFreeCICNumsInSpan=0,DPC=3);
SSAInternalMsg(SS7CICConfigGroup(Node=1,StackID=3,BaseCIC
  Number=0,BaseCICSpan=1,BaseCICChannel=1,NumCICs=23,CallControlUserPart=0,Range=0:0-
  0:0,NumFreeCICNumsInSpan=0,DPC=4);
SSAInternalMsg(SS7CICConfigGroup(Node=1,StackID=2,BaseCIC
  Number=24,BaseCICSpan=2,BaseCICChannel=0,NumCICs=24,CallControlUserPart=0,Range=0:0-
  0:0,NumFreeCICNumsInSpan=0,DPC=4);
SSAInternalMsg(SS7CICConfigGroup(Node=1,StackID=3,BaseCIC
  Number=24,BaseCICSpan=3,BaseCICChannel=0,NumCICs=24,CallControlUserPart=0,Range=0:0-
  0:0,NumFreeCICNumsInSpan=0,DPC=3);
SS7CICConfig(Node=1,StackID=2,BaseCICNumber=60,BaseCICSpan
  =30,BaseCICChannel=1,NumCICs=29,DPC=4,CallControlUserPart=0);
SS7CICConfig(Node=1,StackID=3,BaseCICNumber=60,BaseCICSpan
  =31,BaseCICChannel=1,NumCICs=29,DPC=3,CallControlUserPart=0);
SS7CICConfig(Node=1,StackID=2,BaseCICNumber=90,BaseCICSpan
  =32,BaseCICChannel=0,NumCICs=30,DPC=4,CallControlUserPart=0);
SS7CICConfig(Node=1,StackID=3,BaseCICNumber=90,BaseCICSpan
  =33,BaseCICChannel=0,NumCICs=30,DPC=3,CallControlUserPart=0);
```

```
# T1 Instructions
InseizeInstrListConfig(Node=1,Range=9:0-
  9:23,InstrNum=255,InstrType=0x00,InstrData1=0x00,Instr
  Data2=0x00);
InseizeInstrListConfig(Node=1,Range=9:0-
  9:23,InstrNum=1,InstrType=0x07,InstrData1=0x00,InstrDa
  ta2=0x00);
InseizeInstrListConfig(Node=1,Range=9:0-
  9:23,InstrNum=2,InstrType=0x03,InstrData1=0x01,InstrDa
  ta2=0x00);
InseizeInstrListConfig(Node=1,Range=9:0-
  9:23,InstrNum=3,InstrType=0x06,InstrData1=0x00,InstrDa
  ta2=0x00);
InseizeInstrListConfig(Node=1,Range=9:0-
  9:23,InstrNum=4,InstrType=0x04,InstrData1=0x00,InstrDa
  ta2=0x00);
ImpulsingParametersConfig(Node=1,Range=9:0-
  9:23,Stage=0x01,NumDigitStrings=0x01,StageCompleteTime
  out=0x3e8,AddrSignallingType=0x01,String1Method=0x01,S
  tring1Data=0x04,String2Method=0x00,String2Data=0x00);
TrunkTypeConfig(Node=1,Range=4:0-7:23,TrunkType=0x01);
TrunkTypeConfig(Node=1,Range=5:0-7:23,TrunkType=0x01);
TrunkTypeConfig(Node=1,Range=9:0-9:23,TrunkType=0x03);
StartDialConfig(Node=1,Range=9:0-
  9:23,StartDialType=0x01,StartDialValue=0x04);
AnswerSuperviseConfig(Node=1,Range=9:0-
  9:23,AnswerMode=2);
AnswerSuperviseConfig(Node=1,Range=10:0-
  13:29,AnswerMode=2);

# E1 Instructions
InseizeInstrListConfig(Node=1,Range=34:0-
  37:29,InstrNum=255,InstrType=0,InstrData2=0,instrData1
  =0);
InseizeInstrListConfig(Node=1,Range=34:0-
  37:29,InstrNum=1,InstrType=7,instrData1=0
  ,InstrData2=0);
InseizeInstrListConfig(Node=1,Range=34:0-
  37:29,InstrNum=2,InstrType=3,instrData1=1
  ,InstrData2=0);
InseizeInstrListConfig(Node=1,Range=34:0-
  37:29,InstrNum=3,InstrType=3,instrData1=2
  ,InstrData2=0);
InseizeInstrListConfig(Node=1,Range=34:0-
  37:29,InstrNum=4,InstrType=2,instrData1=15,InstrData2=
  3);
InseizeInstrListConfig(Node=1,Range=34:0-
  37:29,InstrNum=5,InstrType=3,instrData1=4
  ,InstrData2=0);
InseizeInstrListConfig(Node=1,Range=34:0-
  37:29,InstrNum=6,InstrType=2,instrData1=13,InstrData2=
  6);
InseizeInstrListConfig(Node=1,Range=34:0-
  37:29,InstrNum=7,InstrType=6,instrData1=0
  ,InstrData2=0);
```

```

InseizeInstrListConfig(Node=1,Range=34:0-
  37:29,InstrNum=8,InstrType=4,instrData1=0
  ,InstrData2=0);
ImpulsingParametersConfig(Node=1,Range=34:0-
  37:29,Stage=1,NumDigitStrings=1,StageCompleteTimeout=1
  000,AddrSignallingType=3,String1Method=5,String2Method
  =0,String1Data=10,String2Data=0);
ImpulsingParametersConfig(Node=1,Range=34:0-
  37:29,Stage=2,NumDigitStrings=1,StageCompleteTimeout=1
  000,AddrSignallingType=3,String1Method=5,String2Method
  =0,String1Data=255,String2Data=0);
ImpulsingParametersConfig(Node=1,Range=34:0-
  37:29,Stage=4,NumDigitStrings=1,StageCompleteTimeout=1
  000,AddrSignallingType=3,String1Method=5,String2Method
  =0,String1Data=1,String2Data=0);
AnswerSuperviseConfig(Node=1,Range=30:1-
  30:30,AnswerMode=2);
AnswerSuperviseConfig(Node=1,Range=31:1-
  31:30,AnswerMode=2);
AnswerSuperviseConfig(Node=1,Range=32:0-
  32:30,AnswerMode=2);
AnswerSuperviseConfig(Node=1,Range=33:0-
  33:30,AnswerMode=2);
AnswerSuperviseConfig(Node=1,Range=34:0-
  37:29,AnswerMode=2);

# DSP Configuration
DSPSIMMConfig(Node=1,Slot=0x01,SIMM=0x01,DSP0Func=0x10,DS
  P1Func=0x12,DSP2Func=0x14,DSP3Func=0x08,dsp0func=22,ds
  p1func=24,dsp2func=5,dsp3func=1);
DSPSIMMConfig(Node=1,Slot=0x01,SIMM=0x02,DSP0Func=0x02,DS
  P1Func=0x02,DSP2Func=0x02,DSP3Func=0x23,dsp0func=1,dsp
  1func=2,dsp2func=16,dsp3func=35);
DSPSIMMConfig(Node=1,Slot=0x01,SIMM=0x03,DSP0Func=0x01,DS
  P1Func=0x01,DSP2Func=0x01,DSP3Func=0x01,dsp0func=1,dsp
  1func=2,dsp2func=18,dsp3func=35);
DSPSIMMConfig(Node=1,Slot=1,Simm=0,dsp0func=28,dsp1func=2
  8,dsp2func=28,dsp3func=28);
RecordedAnnouncement(Node=1,Slot=0x01,SIMM=0x00,ID=0x03,F
  ormat=0x00,Encoding=0x01,File="C:\PROGRA~1\EXCEL~1\SWI
  ~1/
  C__Program_Files_Excel_Switching_Corporation_SwitchKit
  _cfg_greeting-u.vox",CSAFilename="C:\Program
  Files\Excel Switching
  Corporation\SwitchKit\cfg\greeting-u.vox");

# VDAC Configuration. SIP & H.323
IPAddressConfig(Node=1,Slot=0x05,DataType=0x00,TLVCount=1
  3,Data=1:9:255:10:10:144:100:255:255:255:0:5:5:255:10:
  10:144:1:1:9:3:10:10:144:104:255:255:255:0:5:5:3:10:10
  :144:1:1:9:2:10:10:144:103:255:255:255:0:5:5:2:10:10:1
  44:1:1:9:1:10:10:144:102:255:255:255:0:5:5:1:10:10:144
  :1:1:9:0:10:10:144:101:255:255:255:0:5:5:0:10:10:144:1
  :9:4:0:0:0:0:10:4:0:0:0:0:03:00);

```



```
VOIPProtocolConfig(Node=1,Slot=4,TLVCount=0x01,Data=1:200:0:1:5);
VOIPProtocolConfig(Node=1,UnknownAIB,TLVCount=17,Data=01:0xC8:00:01:04:0x02:0x64:00:0x08:48:46:48:46:48:46:48:00:0x02:0x62:00:02:0x13:0xc4:0x02:0x65:00:0x0b:76:85:67:69:78:84:45:79:78:80:00:0x02:0x66:00:0x09:48:48:48:48:48:48:48:00:0x02:0x67:00:04:0x00:0x00:0x01:0xf4:0x02:0x69:00:04:0x00:0x00:0x0f:0xa0:0x02:0x6a:00:01:0x07:0x02:0x6b:00:01:0x0b:0x02:0x72:00:04:0x00:0x00:0x0e:0x10:0x02:0x75:00:01:0x01:0x02:0x79:00:01:0x00:0x02:0x7a:00:01:0x00:0x02:0x70:00:02:0x00:0x01:0x02:0x7b:00:04:0x00:0x00:0x07:0x08:0x02:0x7c:00:04:0x00:0x00:0x01:0x2c);
ResourceAttributeConfig(Node=1,Slot=0x05,DataType=0x00,TLVCount=0x01,Data=1:208:0:1:2);
ResourceAttributeConfig(Node=1,Slot=0x05,DataType=0x00,TLVCount=0x14,Data=0:9:0:6:62:4:208:209:43:56:1:0:0:1:1:1:1:0:1:1:1:2:0:1:0:1:3:0:1:1:1:194:0:1:75:1:195:0:4:0:0:0:150:1:196:0:1:7:1:197:0:1:0:1:198:0:1:1:1:199:0:1:1:1:209:0:1:0:1:210:0:1:0:1:211:0:1:0:1:235:0:4:0:0:0:0:1:225:0:1:1:1:226:0:1:0:1:236:0:4:0:0:0:0:1:223:0:1:0:1:212:0:1:0);
ResourceAttributeConfig(Node=1,Slot=0x05,DataType=0x00,TLVCount=0x14,Data=0:9:0:6:62:4:208:209:43:57:1:0:0:1:1:1:1:0:1:1:1:2:0:1:0:1:3:0:1:1:1:194:0:1:75:1:195:0:4:0:0:0:150:1:196:0:1:7:1:197:0:1:0:1:198:0:1:1:1:199:0:1:1:1:209:0:1:0:1:210:0:1:0:1:211:0:1:0:1:235:0:4:0:0:0:0:1:225:0:1:1:1:226:0:1:0:1:236:0:4:0:0:0:0:1:223:0:1:0:1:212:0:1:0);
ResourceAttributeConfig(Node=1,Slot=0x05,DataType=0x00,TLVCount=0x14,Data=0:9:0:6:62:4:208:209:43:58:1:0:0:1:1:1:1:0:1:1:1:2:0:1:0:1:3:0:1:1:1:194:0:1:75:1:195:0:4:0:0:0:150:1:196:0:1:7:1:197:0:1:0:1:198:0:1:1:1:199:0:1:1:1:209:0:1:0:1:210:0:1:0:1:211:0:1:0:1:235:0:4:0:0:0:0:1:225:0:1:1:1:226:0:1:0:1:236:0:4:0:0:0:0:1:223:0:1:0:1:212:0:1:0);
ResourceAttributeConfig(Node=1,Slot=0x05,DataType=0x00,TLVCount=0x14,Data=0:9:0:6:62:4:208:209:43:59:1:0:0:1:1:1:1:0:1:1:1:2:0:1:0:1:3:0:1:1:1:194:0:1:75:1:195:0:4:0:0:0:150:1:196:0:1:7:1:197:0:1:0:1:198:0:1:1:1:199:0:1:1:1:209:0:1:0:1:210:0:1:0:1:211:0:1:0:1:235:0:4:0:0:0:0:1:225:0:1:1:1:226:0:1:0:1:236:0:4:0:0:0:0:1:223:0:1:0:1:212:0:1:0);
SSAInternalMsg(VOIPRouting(Node=1,Slot=5,Module=5,IP_Address=10.10.144.101-10.10.144.102-10.10.144.103-10.10.144.104-,Range=16,NumberOfChannels=160);

PPLConfig(Node=1,Range=16:0-20:31,ComponentID=0x61,Entity=0x01,ConfigData=17:2:0:12:8:8:1:20:0:21:0:22:11:50:3:51:19:52:1:53:6:54:6:55:2:56:0:57:4:58:15:59:1:60:11);

PPLTool(Node=1,CfgFile="C:\PROGRAM FILES\EXCEL SWITCHING CORPORATION\SWITCHKIT\CFG\CONFIG_1.CFG");
```

```
SSAInternalMsg(IntFileName(CfgFile="C:\PROGRAM
FILES\EXCEL SWITCHING
CORPORATION\SWITCHKIT\CFG\CONFIG_1.CFG",Node=1,EnableR
outeNode1=1));

# ISDN Span 8 NI-2 User Side.
DChannelAssign(Node=1,Slot=10,DChannelSpanID=8,DChannel=2
3,SecondaryDChannelFacility=0x00,DChannelType=0x01,Data=0xff);
ISDNInterfaceConfig(Node=1,Span=0x08,Channel=0x17,Entity=
0x01,Data=0:9);
ISDNInterfaceConfig(Node=1,Span=0x08,Channel=0x17,Entity=
0x05,Data=0:0);
BChannelConfig(Node=1,Range=8:0-
8:22,Entity=0x01,Value=0x01);

ClearAllChanGroups();
AssociateChanGroup(Range=0:1-0:23,ChannelGroup="SS7ANSI-
OUT");
AssociateChanGroup(Range=1:1-1:23,ChannelGroup="SS7ANSI-
IN");
AssociateChanGroup(Range=2:0-2:23,ChannelGroup="SS7ANSI-
OUT");
AssociateChanGroup(Range=3:0-3:23,ChannelGroup="SS7ANSI-
IN");
AssociateChanGroup(Range=30:1-30:30,ChannelGroup="SS7ITU-
OUT");
AssociateChanGroup(Range=31:1-31:30,ChannelGroup="SS7ITU-
IN");
AssociateChanGroup(Range=32:1-32:30,ChannelGroup="SS7ITU-
OUT");
AssociateChanGroup(Range=33:1-33:30,ChannelGroup="SS7ITU-
IN");
AssociateChanGroup(Range=4:1-7:23,ChannelGroup="E&M");
AssociateChanGroup(Range=8:0-8:22,ChannelGroup="ISDN");
AssociateChanGroup(Range=9:0-9:7,ChannelGroup="FXS-LS");
AssociateChanGroup(Range=16:0-20:31,ChannelGroup="VDAC");
AssociateChanGroup(Range=34:0-37:29,ChannelGroup="E1R2");

# 0 - Round Robin
# 1 - Ascending
# 2 - Descending
# 3 - LRU (Least Recently Used) is allocated first
# 4 - MRU (Most Recently Used) is allocated first
ConfigChanGroup(ChannelGroup="SS7ANSI-OUT", Entity=0,
Parameter=3);
ConfigChanGroup(ChannelGroup="SS7ANSI-IN", Entity=0,
Parameter=3);
ConfigChanGroup(ChannelGroup="SS7ITU-OUT", Entity=0,
Parameter=3);
ConfigChanGroup(ChannelGroup="SS7ITU-IN", Entity=0,
Parameter=3);
```

```
ConfigChanGroup(ChannelGroup="E&M", Entity=0,  
Parameter=3);  
ConfigChanGroup(ChannelGroup="ISDN", Entity=0,  
Parameter=3);  
ConfigChanGroup(ChannelGroup="FXS-LS", Entity=0,  
Parameter=3);  
ConfigChanGroup(ChannelGroup="VDAC", Entity=0,  
Parameter=3);  
ConfigChanGroup(ChannelGroup="E1R2", Entity=0,  
Parameter=3);
```