Redpaper

**Jeff L. Smith**
**Edward Burr**

# SNA Modernization Strategy for Access Node Connectivity

## Introduction

This IBM® Redpaper™ publication describes the implementation of the Systems Network Architecture (SNA) modernization strategy with respect to access node connectivity. There are many established SNA applications used in financial, medical, and manufacturing industries that run on access nodes (remote endpoints) to execute and track transactions. As Internet usage increases in these industries, the established SNA applications remain critical because of business process requirements.

This paper describes the strategy of consolidating the underlying SNA resources and management to data centers to be more efficient and cost-effective. The applications on the endpoints remain untouched and the connectivity becomes modernized. The cloud-like strategy of SNA modernization for client/server implementation is described with scenarios and examples.

This paper is for developers who want to learn more about implementing SNA modernization strategies in order to make their data centers more efficient and cost-effective.

## SNA modernization

The SNA modernization strategy allows SNA applications to take advantage of modern network enhancements without changing the applications. To implement this strategy in a widely distributed network of SNA nodes, the Remote API Client/Server feature provides a cloud-like solution for connectivity.

The Remote API Client supports AIX, Linux, and Microsoft Windows applications in both 32-bit and 64-bit environments. The server code is supported by the Communications Server for Data Center Deployment v7.0, and Communications Server for AIX, Linux, and Linux on System z v6.4, and is shipped in the media pack of these servers.

Figure 1 shows an overview of the SNA modernization strategy.

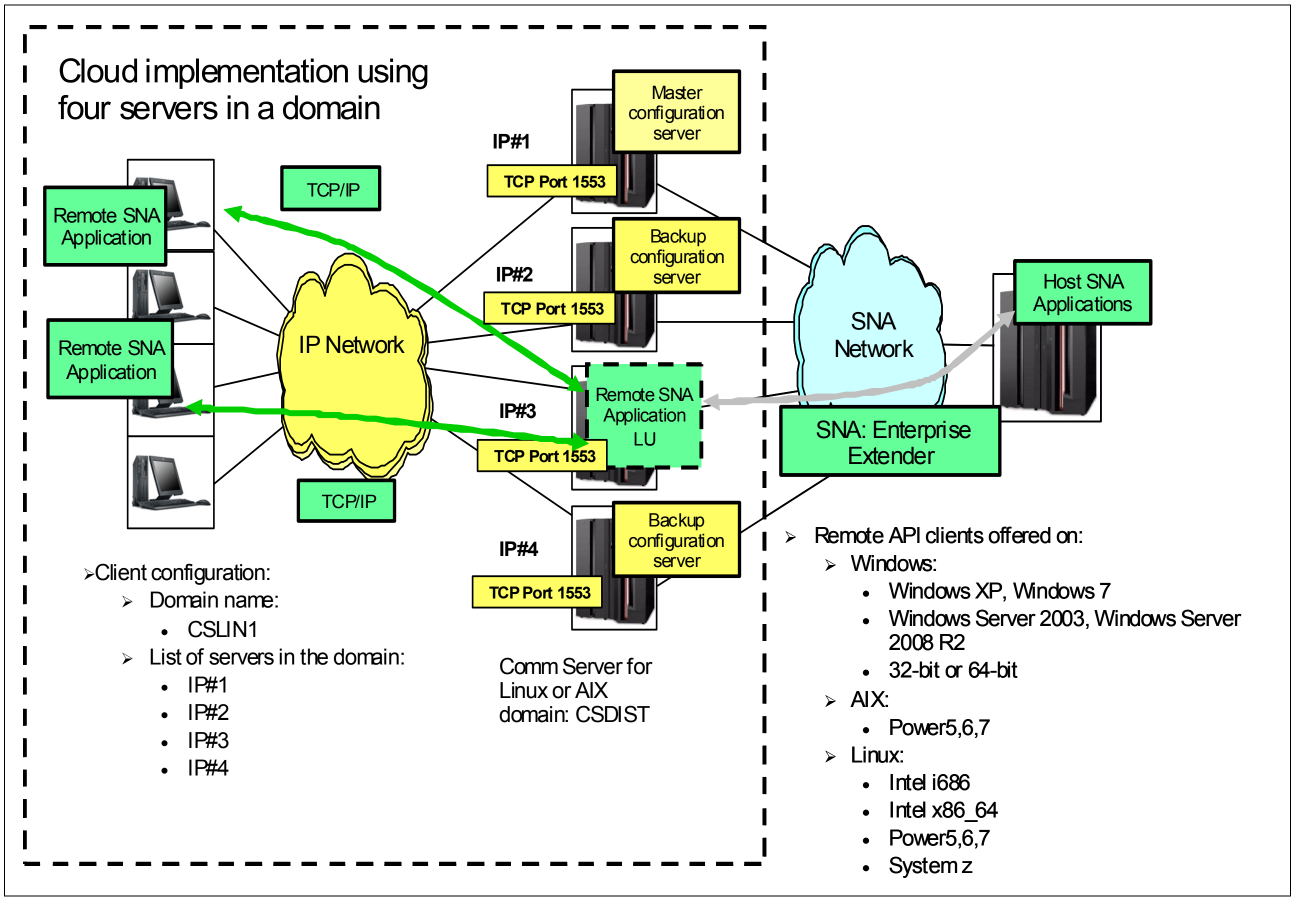


*Figure 1 SNA modernization overview*

## How it works

The Remote API Client provides SNA libraries locally on the client system for an application program to use. The client software runs in the user space and establishes a TCP/IP connection to port 1553 on the Communications Server server. The connection is used to transport library calls between the client and the server. When a program issues a call to the SNA libraries, the call is transported to the server for processing. The response is transported back to the client and then passed to the program. The SNA program that runs on the client system is not aware that it is relying directly on the server for any connectivity.

Clients connect to a domain of servers that can have up to nine servers. The domain is similar to a cloud in which SNA resources on multiple servers can be accessed by the clients in the network. This domain is not related to an IP network domain (the last part of a fully qualified host name) or a Microsoft Windows domain. Instead, the domain is a separate concept that identifies a group of SNA servers and their clients. The Remote API Client is configured to initially connect to a domain and a list of servers in the domain. When a client requests a specific resource, the server provides connection information to the client so that the client can establish a direct TCP/IP connection with the server that has the desired resource. The SNA application is unaware of the specific server that is providing the SNA resource access.

## Benefits

The primary benefit of the Remote API Client/Server implementation is that the SNA resources are consolidated into the data center. This consolidation saves maintenance, installation, and configuration costs because a full SNA node is not needed on every machine. The Remote API Client requires only a few parameters to be fully configured, compared to an average SNA node that requires numerous parameters. The consolidation into the data center also centralizes the SNA skills that are needed to manage the connectivity of the application to the mainframe host applications. Often, you can consolidate the SNA resources without significant changes to either the mainframe or application configuration.

The current System z and Power platforms support Linux and AIX in virtual engines that do not include any additional footprint for the server implementations. Allocating the processor, memory, and network space for the server does not add to the physical layout in the data center. The distributed Communications Server supports these virtual environments with the latest Advanced Peer-to-Peer Networking/High Performance Routing (APPN/HPR) enhancement for virtual network performance, known as *Progressive Adaptive Rate Base (ARB) protocol*. Progressive ARB allows the distributed Communications Server to optimize the processor and bandwidth in a virtual server.

When applications migrate to use Remote API Client/Server, changes to the application code are not required. In some instances, programs might need to re-link their applications because the linkage was defined as static. The goal of SNA modernization is to provide IP connectivity for mature SNA applications with minimal changes to the applications.

**Important:** Older C programs can experience an issue with 32-bit compatibility on Microsoft 64-bit operating systems. The compatibility issue concerns how the operating system resolves the C-runtime library path. Applications link with a manifest file, which helps Microsoft Windows find the appropriate C-runtime library to use for the application.

More information about C Run-Time Error is available at the following website:

http://msdn.microsoft.com/en-US/library/ms235560%28v=VS.80%29.aspx

Figure 2 shows the differences between the traditional style of SNA networking and the style that uses the Remote API.

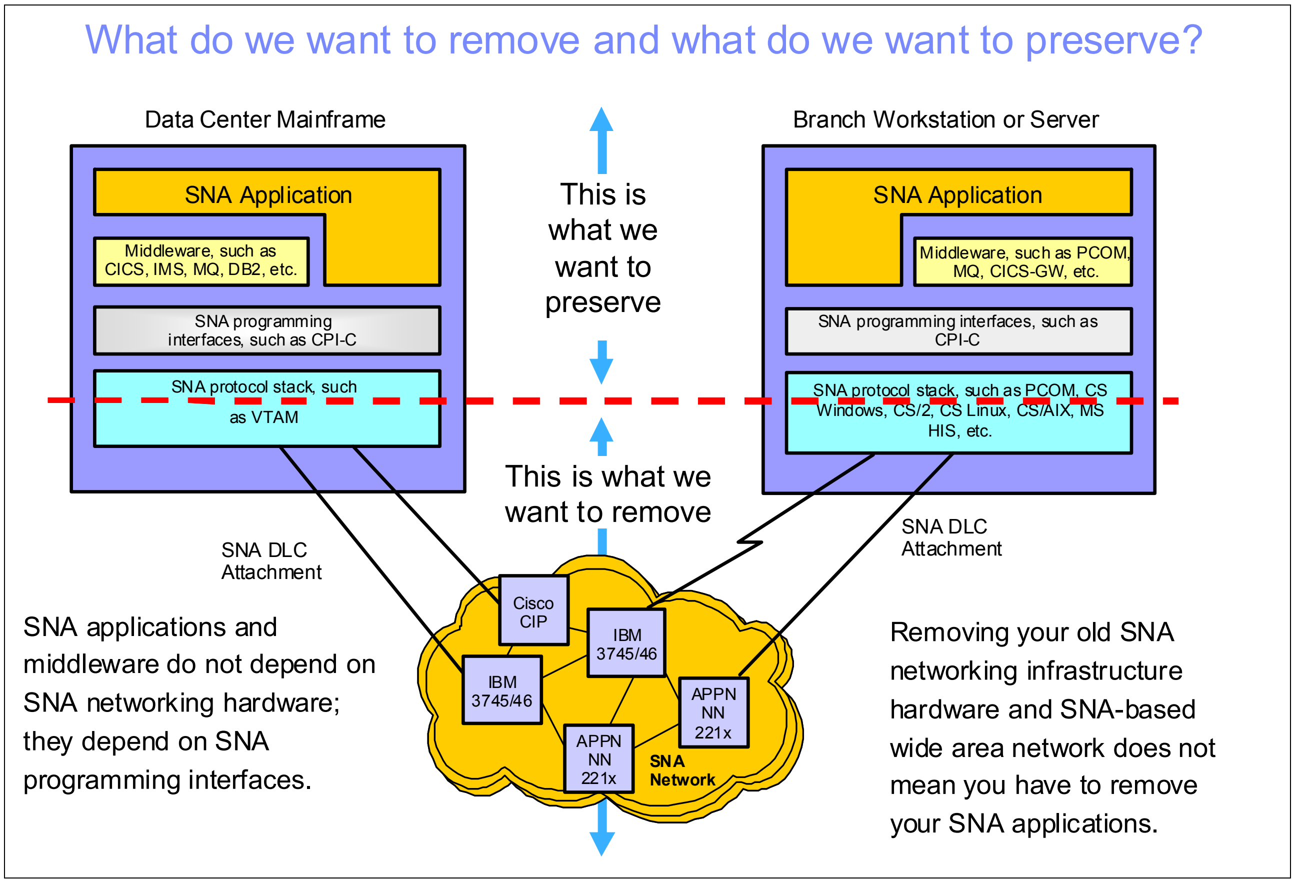
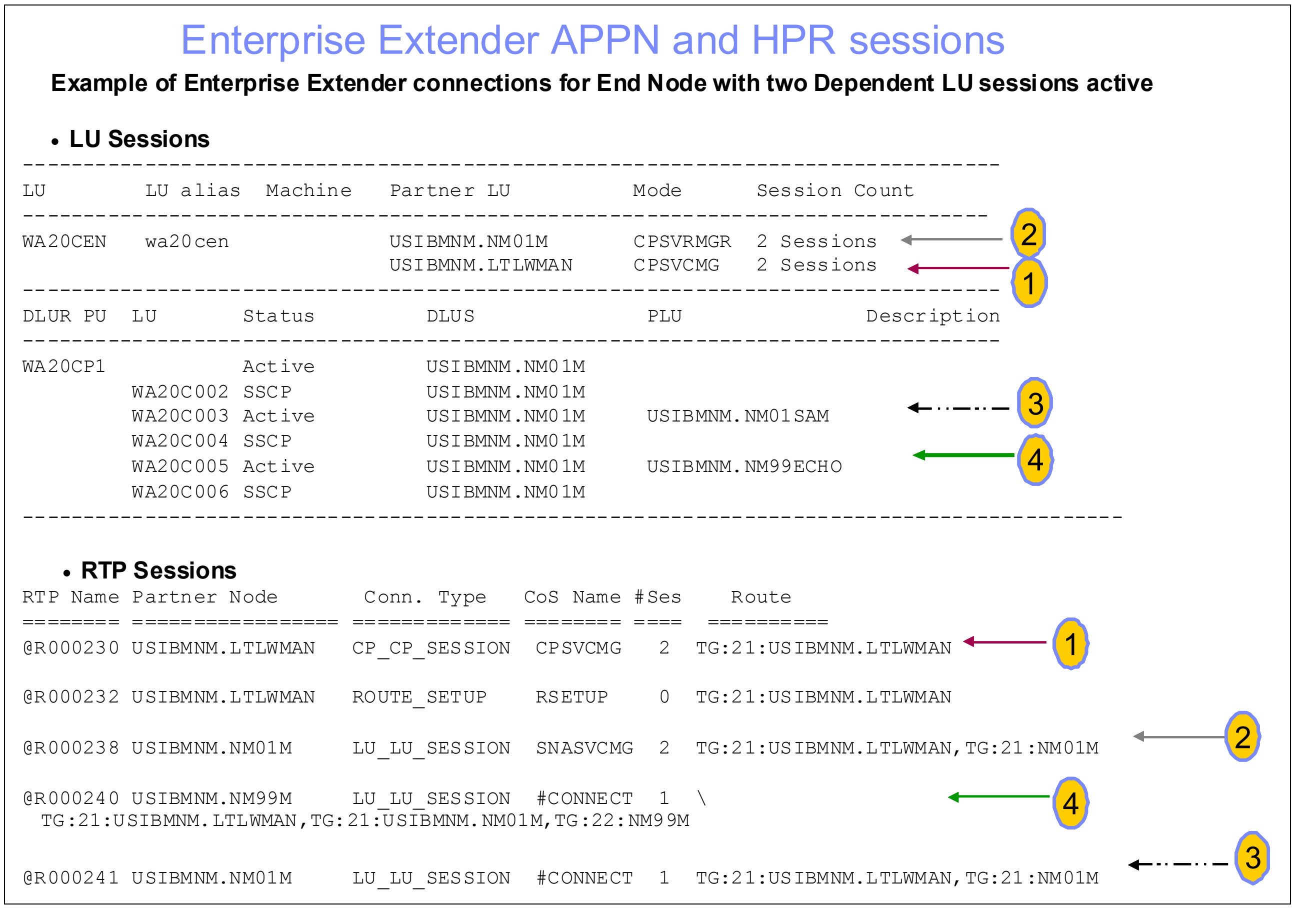

What do we want to remove and what do we want to preserve?
Data Center Mainframe
SNA Application
Middleware, such as CICS, IMS, MQ, DB2, etc.
SNA programming interfaces, such as CPI-C
SNA protocol stack, such as VTAM
This is what we want to preserve
This is what we want to remove
Branch Workstation or Server
SNA Application
Middleware, such as PCOM, MQ, CICS-GW, etc.
SNA programming interfaces, such as CPI-C
SNA protocol stack, such as PCOM, CS Windows, CS/2, CS Linux, CS/AIX, MS HIS, etc.
SNA DLC Attachment
SNA DLC Attachment
Cisco CIP
IBM 3745/46
IBM 3745/46
APPN NN 221x
APPN NN 221x
SNA Network
SNA applications and middleware do not depend on SNA networking hardware; they depend on SNA programming interfaces.
Removing your old SNA networking infrastructure hardware and SNA-based wide area network does not mean you have to remove your SNA applications.

*Figure 2 Removing old SNA networking infrastructure hardware and SNA-based wide area network (WAN)*

For example, an SNA application might run on several thousand desktops or notebooks that run Windows. For these instances of the application, a user might need only one logical unit (LU) to access the host application. The application can use the IBM Personal Communications (PCOMM) software to provide an SNA node, which is configured to connect directly to the mainframe. Replacing the PCOMM SNA stack with the Remote API Client provides the same SNA libraries for the application to use. In the data center, LUs can be consolidated into a few servers, replicated for redundancy, or pooled for easier, more manageable access. The number of Enterprise Extender connections on the mainframe is reduced from one for every remote user to one for every server.

Enterprise Extender often is the method by which the distributed SNA applications are connected to a mainframe application. The connection is made by using HPR and encapsulating SNA traffic in User Datagram Protocol/Internet Protocol (UDP/IP) packets. For Independent LU6.2 applications, this connection requires two CP-CP (Control Point Service Manager (CPSVCMG)) sessions per SNA node. For Dependent LU6.2 or Dependent LU0-3 applications, two CP-CP sessions and two Dependent Logical Unit Requester/Dependent Logical Unit Server (DLUR/DLUS) sessions (in addition to the actual sessions the application uses) are required to manage the connectivity.

As shown in Figure 3, there are six HPR sessions. A minimum of four HPR overhead sessions are required for every SNA node in this network. For 1,000 workstations, this means that 4,000 overhead HPR sessions are required.



Enterprise Extender APPN and HPR sessions

**Example of Enterprise Extender connections for End Node with two Dependent LU sessions active**

- **LU Sessions**

```
--------------------------------------------------------------------------------
LU         LU alias  Machine    Partner LU          Mode      Session Count
-------------------------------------------------------------------------------
WA20CEN    wa20cen              USIBMNM.NM01M       CPSVRMGR  2 Sessions
                                USIBMNM.LTLWMAN     CPSVCMG   2 Sessions
--------------------------------------------------------------------------------
DLUR PU  LU       Status         DLUS                PLU                Description
--------------------------------------------------------------------------------
WA20CP1           Active         USIBMNM.NM01M
         WA20C002 SSCP           USIBMNM.NM01M
         WA20C003 Active         USIBMNM.NM01M       USIBMNM.NM01SAM
         WA20C004 SSCP           USIBMNM.NM01M
         WA20C005 Active         USIBMNM.NM01M       USIBMNM.NM99ECHO
         WA20C006 SSCP           USIBMNM.NM01M
------------------------------------------------------------------------------------------
```

- **RTP Sessions**

```
RTP Name Partner Node       Conn. Type   CoS Name #Ses    Route
======== ================= ============= ======== ====  ==========
@R000230 USIBMNM.LTLWMAN   CP_CP_SESSION  CPSVCMG   2   TG:21:USIBMNM.LTLWMAN

@R000232 USIBMNM.LTLWMAN   ROUTE_SETUP    RSETUP    0   TG:21:USIBMNM.LTLWMAN

@R000238 USIBMNM.NM01M     LU_LU_SESSION  SNASVCMG  2   TG:21:USIBMNM.LTLWMAN,TG:21:NM01M

@R000240 USIBMNM.NM99M     LU_LU_SESSION  #CONNECT  1   \
  TG:21:USIBMNM.LTLWMAN,TG:21:USIBMNM.NM01M,TG:22:NM99M

@R000241 USIBMNM.NM01M     LU_LU_SESSION  #CONNECT  1   TG:21:USIBMNM.LTLWMAN,TG:21:NM01M
```

*Figure 3 Enterprise Extender APPN and HPR sessions*

The Remote API Client uses one TCP/IP session to connect to each server. The servers provide the data session resources without needing as many overhead or control sessions. From a z/OS Communications Server (or VTAM) perspective, a network with 1,000 workstations can reduce the SNA overhead sessions from 4,000 to 4, 8, or 12 (depending on the number of servers that support the clients). The Communications Servers also consolidate LUs or applications into similar HPR/IP sessions. The number of actual HPR/IP (or EE) sessions can be a few dozen, as compared to a few thousand sessions.

## Managing SNA resources

Applications connect to each other using LUs. The SNA architecture tracks dependent LU resources from the IP clients so that the LUs on the mainframe can display the IP addresses of the clients that use them. Independent LUs, such as APPC applications, do not have this architecture in SNA. However, the distributed Communication Servers that support Remote API Client/Server can track the client domain name server (DNS) host name by using the LU. The servers can show the DNS host name of the last client to access an LU.

Example 1 shows an example of a dependent LU display.

*Example 1 Dependent LU display*

```
Example of Dependent LU display:

CS Display Dependent LU Active status: Fri Jan 27 16:20:56 2012

=  PU:    NAU: LU:      Status:             Activity:
=  ===    ==== ===      =======             =========
 CSLDEM3  128  SL8LU80  Active Application = LUA w500jls.raleigh.ibm.com
 CSLDEM3  129  SL8LU81  Active Application = LUA wa20c.rtp.raleigh.ibm.com

Example of Independent LU display:

Status display:
CS LINUX Status at time is: Fri Jan 27 16:27:53 2012
-----------------------------------------------------------------------------
LU       LU alias    Machine   Partner LU        Mode     Session Count
-----------------------------------------------------------------------------
LTLWGNA  ltlwgna                                            Inactive
LTLWGNN  ltlwgnn               USIBMNM.NM01M    CPSVCMG   2 Sessions
                               USIBMNM.NM01M    CPSVRMGR  2 Sessions
                               USIBMNM.DICSR2   CPSVCMG   2 Sessions
                               USIBMNM.CSDBLNN  CPSVCMG   2 Sessions
LTLWNAM  ltlwnam   w500jls.ra  USIBMNM.CSDBMG1  #INTER    1 Session
                               USIBMNM.CSDBMG1  SNASVCMG  1 Session
LTLOC     wa20c    wa20c.rtp.  USIBMNM.DICSR1X  #INTER    1 Session
                               USIBMNM.DICSR1X  SNASVCMG  1 Session


Command query of local LU definitions:
snaadmin -d query_local_lu

lu_name = LTLWNAM
list_name = ""
description = Name Server Admin LU
lu_alias = ltlwnam
nau_address = 0
syncpt_support = NO
lu_session_limit = 0
default_pool = NO
pu_name = ""
lu_attributes = NONE
allowed_sscp_id = 0
disable = NO
sys_name = w500jls.raleigh.ibm.com
timeout = 60
```

## Windows 64-bit strategy

The Remote API Client is supported on 64-bit versions of the Windows operating system. This 64-bit version support is significant because Communications Server for Windows and the SNA stack shipped with PCOMM support only 32-bit versions of the Windows operating system. The suggested solution to work around this bit version difference is to install a server on AIX, Linux, or Linux on System z. The Remote API Client on the Windows 64-bit operating system also needs to be installed to connect to the server. The Remote API Client provides both 32-bit and 64-bit SNA libraries, so applications can migrate from older Windows 32-bit operating systems with minimum changes to run on Windows 64-bit operating systems. As of this writing, the Microsoft Host Integration Server (HIS) provides only 64-bit SNA APIs on Windows 64-bit operating systems, and not 32-bit APIs. Figure 4 shows three configurations of remote APIs.

Three configurations of remote APIs

**IBM Communications Server for AIX, Linux, Linux on System z - Remote API Client/Server**

- Consolidation of the SNA protocol stacks into System z is possible using CS Linux on System z
- A domain of servers can be a mix of Linux servers (Intel, System z, Power) or AIX servers
  - o Cannot mix AIX and Linux servers in the same domain
- Remote API Client for Win-x64 provides 32-bit and 64-bit API libraries:
  - o IBM SNA API Clients have only 32-bit API support in Win-x64
  - o MS SNA Client currently has only 64-bit API support on Win-x64

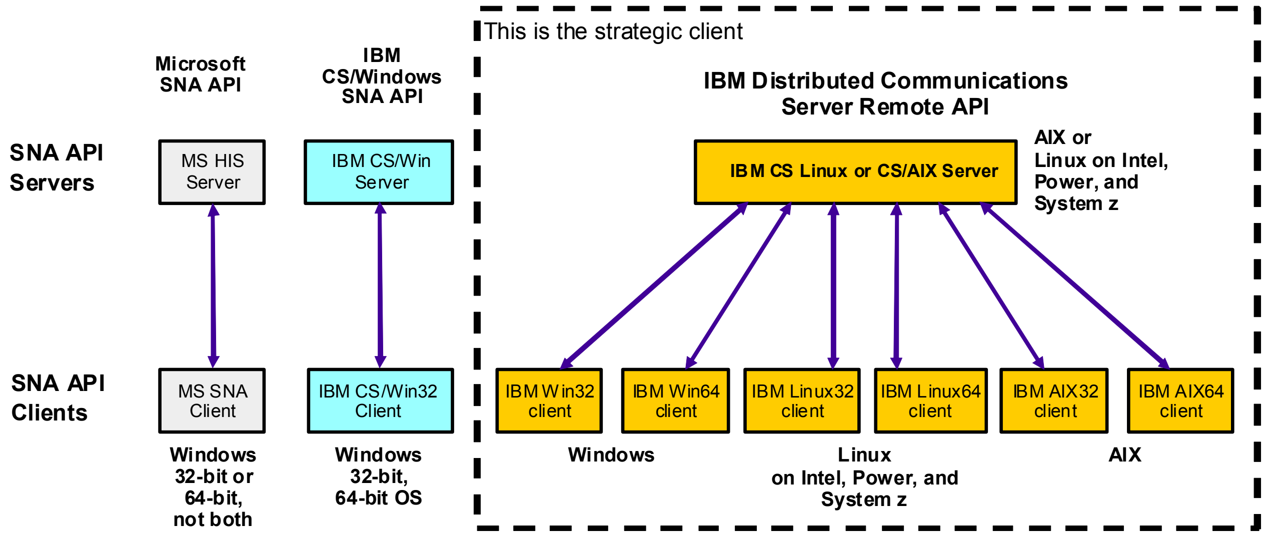

This is the strategic client
Microsoft SNA API
IBM CS/Windows SNA API
IBM Distributed Communications Server Remote API
SNA API Servers
MS HIS Server
IBM CS/Win Server
IBM CS Linux or CS/AIX Server
AIX or Linux on Intel, Power, and System z
SNA API Clients
MS SNA Client
IBM CS/Win32 Client
IBM Win32 client
IBM Win64 client
IBM Linux32 client
IBM Linux64 client
IBM AIX32 client
IBM AIX64 client
Windows 32-bit or 64-bit, not both
Windows 32-bit, 64-bit OS
Windows
Linux on Intel, Power, and System z
AIX

*Figure 4 Remote API configurations*

Alternative compatibility solutions for 64-bit Windows systems might be to use the SNA API Client that is shipped with Communications Server for Windows or the Microsoft HIS SNA client. These clients provide 32-bit APIs in a 32-bit compatibility mode on 64-bit Windows. The clients also use TCP/IP to connect to a Windows server.

These solutions require much more configuration work than a Remote API Client/Server implementation and limit the number of clients for every server. The Remote API Client/Server requires only three or four parameters and supports thousands of clients for every server. These SNA API clients and HIS SNA clients connect only to Windows servers. The Remote API Client connects only to a domain of AIX, Linux, or Linux on System z servers.

The SNA API Client and HIS SNA client provide connectivity to specific servers for SNA resource connectivity. The client can be configured only to connect to one specific server for the LU resources needed. The servers do not participate in a domain to share resources. The Remote API Client uses a cloud-type of implementation in which the SNA resources are duplicated across the domain such that there is no single point of access or failure.

## High availability

A key concept to implementing the Remote API Client/Server is to map local LUs on the servers to the client applications on the remote clients. This mapping helps to match clients to the sessions the clients have with the mainframe applications.

In SNA, multiple SNA nodes (servers) can have the same LU aliases, but each SNA node must have a unique LU name in the SNA network. There might be two servers in a domain, “servd1a” and “servd1b”. On both servers, there might be an LU alias “D1LU”. On servd1a, the LU alias might be defined for local LU SRV1ALU, and on servd1b, the same LU alias might be defined as the alias for local LU SRV1BLU. When a client connects to a server and requests to start a transaction program (TP), the client identifies the LU alias as D1LU. If the client is connected to servd1a, the local LU SRV1ALU is used. If servd1a is not available, the other server provides the SNA connectivity. Thus, SRV1BLU is used as the local LU for the client application. By having duplicate LU aliases on the servers in a domain, the client-independent LU6.2 applications always have access to the partner from one of the servers.

Some implementations use dependent LU6.2 connections to access the mainframe. On the full SNA stack products, PCOMM, Communications Server for Windows, and Microsoft HIS, a dependent LU provides one session for the specified LU. If the LU is not available, or the processing unit that owns that dependent LU is not active, the connection fails. When the Remote API Client/Server is implemented, dependent LU6.2 resources can be placed into one default LU pool. If an application requests an LU that is in the pool, the LU is provided to the application if the LU is available. If the LU is busy or already connected, the server provides another LU from the pool.

This pooling of dependent LU6.2 LUs allows applications that issue short transactions a way to share LUs. The number of configured LUs can be less than the number of connecting clients if the rate of using the pooled-dependent LU6.2 resource is less than the number of clients concurrently executing. The Remote API Client/Server queues outstanding TP applications if there are more requests than the pool has available LUs. This feature allows for applications to share common resources, thus reducing the definitions needed on the mainframe.

Dependent LU 0-3 applications can be configured to use multiple pools across servers in a domain. As the dependent LUs of one server are filled, LUs are dynamically allocated from the other servers that share the pool name in the domain.

## Server configuration

The client/server function on AIX and Linux is enabled by defining a client/server domain and the master server for that domain. Optionally, other servers in the domain can be designated as backup servers. Although not required, it is suggested that all other servers in the domain are designated as backup servers.

The concept of master and backup does not mean that one server takes over the resources and connectivity of the master if the master fails. Instead, the master server maintains an internal data store about all the resources controlled by each server in the domain. The backup servers are designated to be ready to take over that function if the master fails.

After the master server is selected, all servers are configured with the server information. This configuration is defined with the **snanetutil** command. The SNA daemons must be stopped to use this command:

```
# snaadmin term_node
# sna stop
# snanetutil my_master.example.com my_domain
# sna start
```

The **snanetutil -d** command is used to delete the current node from the domain, making it a stand-alone server again. Remote API Clients cannot connect to a stand-alone server.

Without any parameters, the **snanetutil** command displays the current domain configuration. The **snanetutil -?** or **snanetutil -help** command shows the syntax of the possible options:

```
# snanetutil
DOMAIN NAME       = my_domain
NUMBER OF SERVERS = 2
SERVER LIST       = servd1a.example.com, servd1b.example.com

# snanetutil -?
Usage: snanetutil [master_mc [domain name] | -d]
```

If not specified, the default domain is `ibmcs-domain`.

To designate any other server as a backup after it is added to the domain, use the **snaadmin add_backup** command:

```
# snaadmin add_backup,backup_name=my_backup.example.com
```

The order in which the backups are defined determines the order in which they attempt to become the master if contact with the current master is lost.

## Client configuration

The Remote API Client configuration on Windows is stored in the registry under subkeys of the following key:

```
\\HKEY_LOCAL_MACHINE\SOFTWARE\SNA Client\SxClient\Parameters
```

As shown in Figure 5, the Windows client has a GUI configuration tool, `sxclconf.exe`, that configures the registry settings.

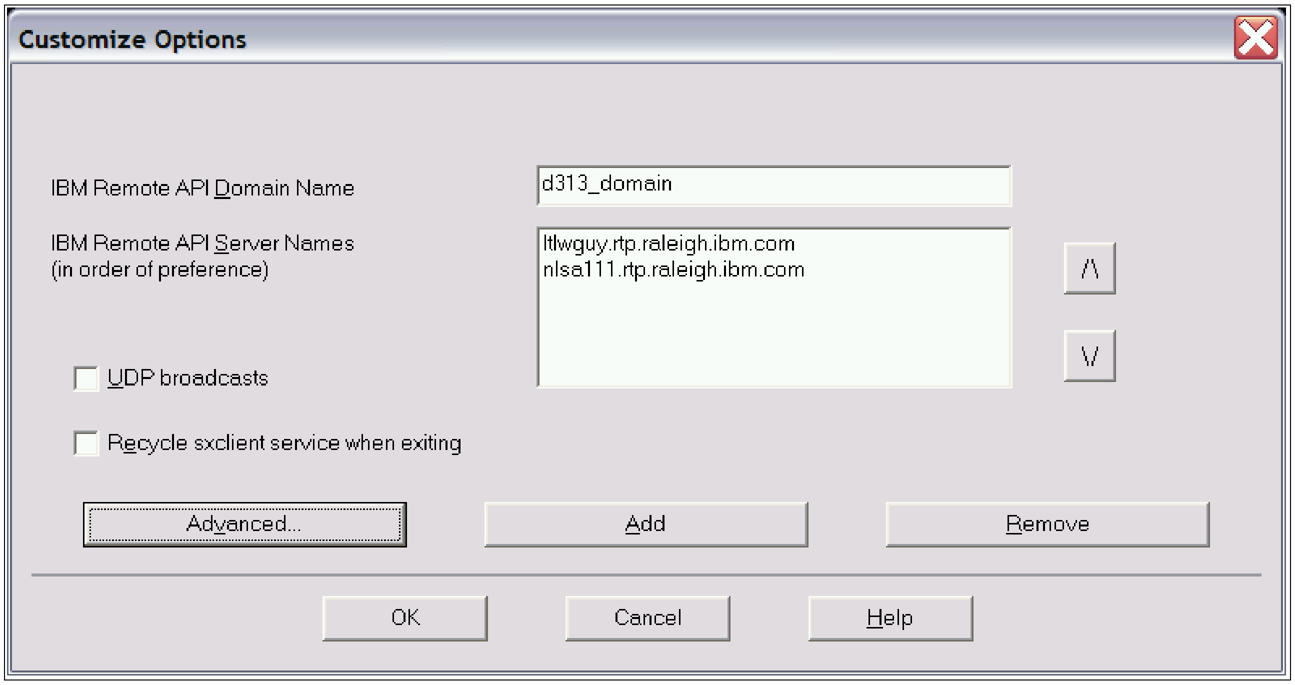

Customize Options
IBM Remote API Domain Name
d313_domain
IBM Remote API Server Names
(in order of preference)
ltlwguy.rtp.raleigh.ibm.com
nlsa111.rtp.raleigh.ibm.com
/\
\/
UDP broadcasts
Recycle sxclient service when exiting
Advanced...
Add
Remove
OK
Cancel
Help

*Figure 5 Customize options*

The only required parameters are the domain under the Configuration subkey and at least one of up to nine server entries under the Servers subkey: Server1, Server2, up to Server9. The client attempts to contact the servers in the order the servers are listed. After a server is contacted, the client can connect to other servers in the list to access SNA resources.

The client configuration on Linux and AIX is implemented by using a flat ASCII file that specifies the few parameters needed. For example, the following parameters are found in the Linux or AIX Remote API Client configuration file:

```
For Linux, /etc/opt/ibm/sna/sna_clnt.net, for AIX, /etc/sna/sna_clnt/net

domain=d313_domain

server_lost_timeout=20

ltlwguy.rtp.raleigh.ibm.com

nlsa111.rtp.raleigh.ibm.com
```

For more information about configuration options reference materials, see “Documentation” on page 11.

## Other configuration considerations

You must plan for logging and encryption configuration settings.

### Logging

Central logging is enabled by default. Every server and client send error reports to the master server so that all logging occurs in one place. The benefit of central logging is that problems are easier to troubleshoot. However, central logging can significantly increase network traffic if many errors are occurring, even if those errors are minor and do not affect the application.

It is possible to disable central logging, which causes each server and client to log all messages in local files, with the following command line:

`# snaadmin set_central_logging = NO` (or `YES` to enable it)

### Encryption

By default, the TCP/IP traffic between client and server is not encrypted; it is assumed that this traffic is occurring on a secure internal network. A secure IP tunnel can be used to transport this traffic. Alternatively, Communications Server provides a method for running the client/server connection over HTTPS by using WebSphere Application Server.

## Documentation

More information about the topics in this paper is available at the Communications Server product library:

http://www.ibm.com/support/docview.wss?uid=swg27019458

Chapters from administration guides and a Quick Beginnings guide are available for reference from the Communications Server library:

- Chapter 10, "Managing CS/AIX Client/Server Systems", in *IBM Communications Server for AIX Administration Guide V6.4*, SC31-8586-04.
- Chapter 10, "Managing Communications Server for Linux Client/Server Systems", in *IBM Communications Server for Linux Administration Guide V6.4*, SC31-6771-03.
- *IBM Communications Server for AIX Quick Beginnings V6.4*, GC31-8583-04:
  - Chapter 4, "Installing IBM Remote API Clients on Linux"
  - Chapter 5, "Installing IBM Remote API Clients on Linux for System z"
  - Chapter 6, "Installing IBM Remote API Clients on AIX Systems"
  - Chapter 7, "Planning for and Installing the Remote API Client on Windows"
  - See the Configuring Client/Server Functions section in Chapter 8, "Configuring and Using CS/AIX"

Guides for programmers also are available for reference from the Communications Server library:

- *IBM Communications Server for AIX or Linux APPC Programmer's Guide V6.4*, SC23-8592-00
- *IBM Communications Server for AIX or Linux CPI-C Programmer's Guide V6.4*, SC23-8591-00
- *IBM Communications Server for AIX or Linux Common Service Verb Programmer's Guide V6.4*, SC23-8589-00
- *IBM Communications Server for AIX or Linux LUA Programmer's Guide V6.4*, SC23-8590-00

## Requirements

The following products provide the support required for the Remote API Clients:

- Communications Server for AIX v6.4
- Communications Server for Linux v6.4
- Communications Server for Linux on System z v6.4

These products have been consolidated into one offering:

- Communications Server for Data Center Deployment v7.0

For more information, refer to:

http://www.ibm.com/common/ssi/ShowDoc.wss?docURL=/common/ssi/rep_ca/8/897/ENUS212-338/index.html&lang=en

The software requirements for the two releases are:

- Communications Server for AIX, Communications Server for Linux v6.4
- Communications Server for Data Center Deployment v7.0

### Communications Server for AIX, Communications Server for Linux v6.4

The platform and operating system have these requirements:

- Server:
  - AIX: 5.3, 6.1, and 7.1
  - Linux:
    - RHEL 4, 5, and 6
    - SLES 9, 10, and 11
- Client: (32-bit or 64-bit)
  - AIX: 5.3, 6.1 and 7.1
  - Linux:
    - RHEL 4, 5, and 6
    - SLES 9, 10, and 11

  - Microsoft Windows:
    - Windows XP
    - Windows Vista
    - Windows 7
    - Windows 2003 Server
    - Windows 2008 Server
    - Windows 2008 Server R2
- Firewall: Needs TCP/IP port 1553 open for client to initiate connection.

### Communications Server for Data Center Deployment v7.0

The platform and operating system have these requirements:

- Server:
  - AIX: 6.1, and 7.1
  - Linux:
    - RHEL 5, and 6
    - SLES 10, and 11
- Client: (32- or 64-bit)
  - AIX: 6.1 and 7.1
  - Linux:
    - RHEL 5, and 6
    - SLES 10, and 11
  - Microsoft Windows:
    - Windows Vista
    - Windows 7
    - Windows 2003 Server
    - Windows 2008 Server
    - Windows 2008 Server R2
    - Windows 2012 Server
- Firewall: Needs TCP/IP port 1553 open for client to initiate connection.

## Configuration examples

Several server configurations can be used to implement the SNA modernization strategy presented in this paper:

- Example 1: One server

  A single Communications Server server is used to provide connectivity to the mainframe. This server is configured as the master of its domain, and all clients are configured to connect to this server.

  This configuration is most common for small networks with only a few hundred users.

- Example 2: Multiple servers with unique resources

  Multiple servers are used to provide connectivity to one or more mainframes. The servers might have different and unrelated resources. One server is configured as the master, and the other servers are designated as backups. Clients are configured with multiple servers in the list; the order is not important. When a client requests a specific resource, the client is given the information to connect to the server that owns that resource.

- Example 3: Multiple servers with similar resources with pooling

  Multiple servers are used to provide connectivity to one or more mainframes. The servers all have similar resources, and it does not matter which specific resource the client uses. On each server, resources are grouped into a pool, and the same pool name is used on all the servers. When a client requests a pool name, it is given any available resource from that pool, which can be on any of the servers.

  This configuration is common for LU0-3, typically for terminals but also printers. A large pool of terminals is available. If a server fails, new resources are provided by the remaining servers.

- Example 4: Multiple servers with similar resources with alias spanning

  Multiple servers are used to provide connectivity to one or more mainframes. The servers have similar resources, and it does not matter which specific resource the client uses. On each server, an alias is defined for a local LU. The same alias name is used for a local LU on each of the servers. When a client requests an alias, the client is given any available local LU matching that alias, which can be on any of the servers.

  This configuration is common for LU 6.2 APPC applications. The client needs to know only the alias name, and the client can use the local LU with that alias on any server.

- Example 5: Multiple similar isolated servers with duplicate aliases for redundancy

  Multiple servers are defined as their own master, but are all in the same domain. As in examples 3 and 4, the same pool names, alias names, or both are used on each server. Clients are configured to connect to a specific server first and an alternative server second.

  This configuration is common for a traditional backup or failover scenario. One server is the primary server, and the second server is a hot standby. The servers do not share resources or communicate with each other. If the client is unable to contact the primary server, the client attempts to contact the second server and use the resources on that server.

- Example 6: Multiple similar isolated servers with duplicate aliases, behind a load balancer

  This configuration is similar to example 5, except the purpose here allows an external load balancer to manage loads among the servers. Additional configuration is required, however, because the clients have only a single server name with which to connect.

  Although each server has a unique host name, there is a special host name used by the clients to connect to the load balancer. The master in the client/server configuration on each server must be set to that special host name. Also, each server must be started to think its host name is that special host name within the client/server function.

  This configuration is more common for LU0-3 than for LU6.2 because it is more applicable to large pools of LUs.

## The team who wrote this paper

This paper was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

**Jeff L. Smith** is the Chief Programmer for the development of Distributed Communications Servers in Enterprise Network Solution for AIM, Software Group, IBM. He has worked at IBM for 23 years in various roles supporting multiprotocol networking software products. Jeff has a Masters of Science degree in Computer Engineering from Florida Atlantic University, and a Bachelor of Science degree in Physics from Appalachian State University,

**Edward Burr** is a Communications Server Software Analyst at RTP, North Carolina. Edward has worked for IBM for 15 years in various roles supporting Communications Server on AIX, Linux, and Windows platforms. Edward holds a B.S. in Science Education from the University of Oklahoma.

## Now you can become a published author, too!

## Stay connected to IBM Redbooks

# Notices

This document REDP-4871-00 was created or updated on March 26, 2013.

Send us your comments in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:
  ibm.com/redbooks
- Send your comments in an email to:
  redbooks@us.ibm.com
- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400 U.S.A.

## Trademarks