Technical
Documentation

SAP Explanatory Notes on the Usage of GSS API Version 2 for SNC

Draft Version

SAP AG

gssapi_sapnotes.doc 19.01.99

SAP AG
Neurottstraße 16
D-69190 Walldorf
Germany

Contents

1. „Secure Network Communications (SNC)“ for SAP R/3

“Secure Network Communications (SNC)” for SAP R/3 is a software layer introduced by SAP into various components of the R/3 System architecture to achieve protection of network communications [1,2]. Different levels of protection are possible, including strong authentication, data integrity checking and encryption (authorization is achieved with R/3 internal mechanisms, not via SNC). The SNC layer is included as an option in standard R/3 installations from R/3 Release 3.1 upwards.

When the SNC layer is activated, it requires the usage of an external security product that has been purchased, installed and configured to run with R/3 separately by the customer. The external security product used has to be certified for SNC within SAP’s Complementary Software Partner (CSP) program. This document provides additional technical information for the integration of external security products and the SNC layer within the R/3 System. This integration is achieved via the “Generic Security Services Application Program Interface Version 2 (GSS API v2)” [3,4].

This chapter contains basic architectural explanations about the usage of SNC within the various components of the R/3 System. It serves to better understand the particular requirements imposed on external security products to achieve proper integration with R/3. Chapter 2 contains SAP specific comments and additional information for each of the GSS API Version 2 function calls used by the SNC layer. Non-functional requirements, such as application binary portability, performance and memory management are described in Chapter 3. How to build the SNC Adapter Library to use the external security product with R/3 Release 3.1 and Release 4 Systems is described in Chapter 4. References to further documentation are given in Chapter 5.

1.1 SNC within the R/3 System Architecture

The fundamental architecture of the R/3 System is based on the three-tier model of distributed client/server processing. This model makes a clear distinction between the database, application and presentation layers as shown in Figure 1. The three-tier client/server architecture extends into a multi-tier architecture when, for example, Internet enabling components are added, as provided by SAP with its Internet Transaction Server (ITS).

The different software components in the distributed R/3 System architecture communicate via network connections using SAP specific protocols over TCP/IP services. For presentation servers (frontend PCs) the SAP Graphical User Interface (SAPgui) program is provided to support interactive dialog connections. The SAPgui talks to work processes running on one or several

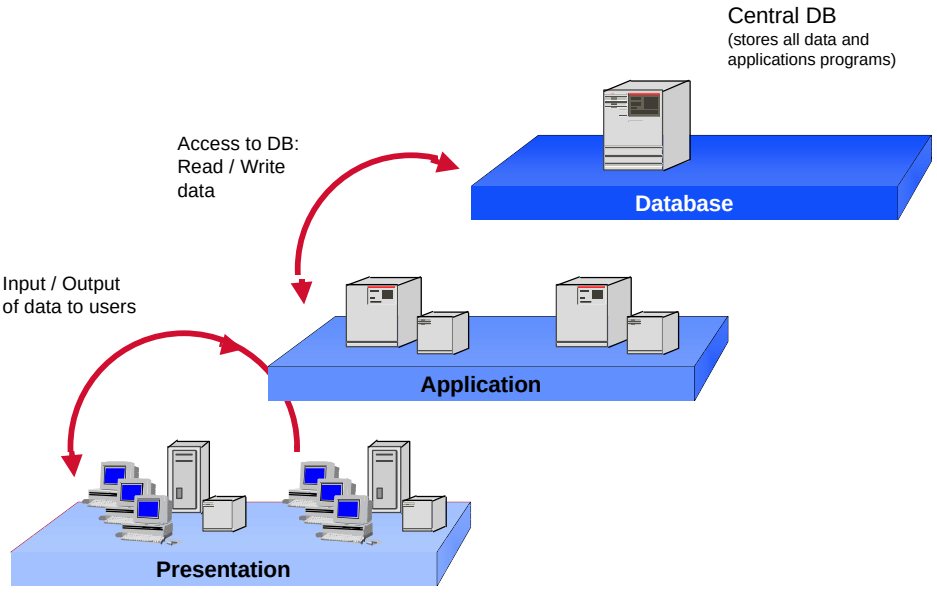

Figure 1. R/3 System Architecture

R/3 application servers. Alternative user interfaces, frontend services and external programs may also communicate with R/3 application servers via the SAP Remote Function Call (RFC). The SAP RFC is also used for communication between R/3 application servers and between R/3 and R/2 Systems. Each R/3 application server connects to a central database server. Printing services at the presentation server are also provided by the SAP Line Printer Daemon (SAPlpd). To restrict R/3 communications and to complement the services provided by conventional firewalls SAP provides an application-level gateway software for R/3 called SAProuter. Using the SAProuter, communication paths can be permitted or denied for the SAP specific protocols at service and port level.

In order to protect the communications of a distributed R/3 System, the SNC layer has been inserted into all of these R/3 System components. When the SNC layer is active, strong mutual authentication between R/3 System processes is supported and a security context is negotiated using the GSS API v2 function calls for each new connection being setup. Within security contexts, data integrity checking and encryption of data is possible.

The concept of the SNC layer within R/3 follows the same architectural guidelines as the GSS API v2. It provides application-level security by processing the data within R/3 System components using GSS API v2 function calls each time before and after data packets are transmitted by the transport layer services. The security services provided via GSS API v2 this way are independent from the transport service. In addition, GSS API v2 provides abstraction from the particular mechanisms used by the external security product to achieve the required level of protection. As a consequence, for example, both symmetric (secret-key) and asymmetric (public-key) based security products can be used for SNC without specific knowledge needed within the R/3 System.

1.2 Process Architecture of the R/3 Application Server

The internal process architecture of an R/3 application server has been designed to support efficient concurrent processing of user transactions while preserving a high level of robustness for the R/3 System instance. The most important processes making up an R/3 System instance are depicted in Figure 2.

The main process is the “Dispatcher (D)”, which creates a configurable number of “Work Processes (WP)” to process individual transaction steps (logical units of work). The Dispatcher establishes all the connections to frontends or other application servers and passes data between these frontends or other application servers and the Work Processes.

The actual data is processed by the so-called “Taskhandler” layer within each Work Process. To achieve high overall throughput of dialog steps, the concurrent dialog contexts of all active R/3 users on that application server are rolled-in and –out for each dialog step to avoid Work Process idle time when waiting for user input at the frontend. This means, however, that different Work Processes are actually being used over and over again during an R/3 user’s session between R/3 logon and logoff.

As a consequence for SNC, security contexts that have been setup using GSS API v2 calls are not tied to a single process but have to be passed between the Work Processes of an R/3 application server. Therefore, the corresponding GSS API v2 calls for exporting and importing of security contexts are a mandatory requirement for interoperability with R/3 (see Chapter 2).

Besides the need for passing security context information between Work Processes this process architecture also explains the importance of performance and other non-functional requirements of R/3 concerning the GSS API v2 calls. The processing time for some of these calls directly affects the dia-

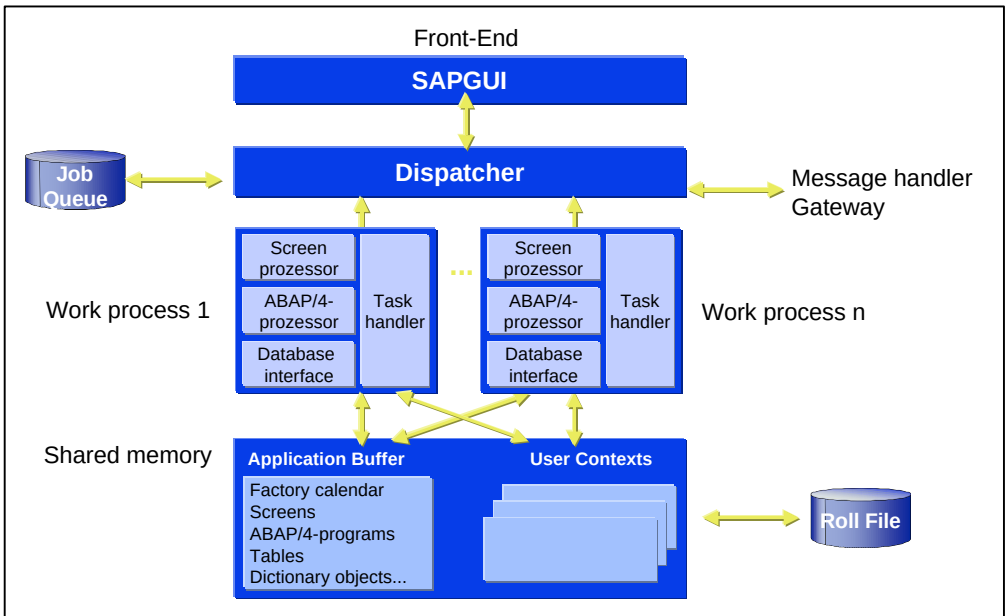

Figure 2. R/3 Application Server Process Architecture

log response time achieved for the R/3 users and the overall transaction throughput. Since Dispatcher and Work Processes are heavy-weight long-running processes, the memory management has to make sure that memory is not wasted and that no memory leaks occur.

1.3 R/3 Configuration and User Management

The SNC option in R/3 is activated using a set of R/3 profile parameters (see R/3 online documentation, or [5]). Before SNC can be activated, the R/3 system components involved and the R/3 users to be authenticated via SNC need to be defined and integrated within the infrastructure as imposed by the external security product used. This may involve, for example, the definition of users at the operating system and network levels, password assignments or generation and distribution of cryptographic keys. Such infrastructure requirements and administrative tasks prior to the use of SNC within R/3 are out of the scope of this documentation.

Within the R/3 System, a mapping takes place between R/3 system components and R/3 users' external identifications (as known by the external security product) and the R/3 user names. External names can be, for example, X.500 Distinguished Names, or Kerberos principal names. Within the R/3 System an access control list is maintained, represented by a mapping table that is used to map external names to R/3 user accounts. The maintenance of this user access control list is done via R/3 user maintenance transactions and will also be possible by accessing a Directory Service in the future. Only if a matching entry exists in the R/3 user access control list, an R/3 logon with an external name via SNC is possible. The R/3 session is then running under the R/3 users identified this way.

2. Functional Requirements for SNC (GSS API v2 Calls)

In this chapter, we list all the API calls that are used by the SNC layer to interface with the external security product. These calls are the ones defined by GSS API v2 in [1, 2]. We provide additional information and explanatory notes about the particular usage of these calls with R/3 where that is necessary.

2.1 Credential Management

Creation of credentials that are usable for initiating as well as accepting security contexts is a local matter of the security product. Usually, a Single Sign-On environment is in place to support this operation. SAP R/3 components will never trigger the creation of credentials or attempt the creation of credentials via proprietary extensions to the GSS-API v2 interface. If the call to GSS_Acquire_cred() fails during the attempt to establish a protected communication channel between any two R/3 peers/components, the communication attempt will be terminated with an error.

Some GSS API mechanisms or implementations may distinguish accepting and initiating credentials and there may be different procedures to make each one available to GSS API-aware applications.

Example: In Kerberos 5, GSS API implementations accepting credentials are often provided in a persistent form, like a file-based "keytab", and they may have a long or indefinite lifetime. Kerberos 5 initiating credentials on the other hand, consist of one or more tickets with limited lifetime in a ticket cache. The creation of Kerberos 5 initiating credentials usually distinguishes two modes of operation: an interactive process for human users and unattended operation for servers in distributed applications. The latter may require the presence of persistent accepting credentials and, varying across implementations/vendors, it may be performed inline and concealed or it may require an external tool that is regularly triggered by some scheduler service like cron.

It is a requirement for the smooth operation of R/3 that all external credential management operations, especially the credential refresh for servers, does not interrupt or interfere with the services provided via the GSS API interface. This may require careful design of credential-related shared resources on behalf of the security product provider to reliably support R/3's parallel multi-process access to both initiating and accepting credentials, all using the same identity.

There are three situations when the SNC library in R/3 will **temporarily** acquire a credentials handle:

(1) to inquire availability and remaining lifetime of a credential, usually during program startup

(2) to inquire availability and remaining lifetime of a credential when an established security context is about to expire or has already expired

(3) while/during the token exchange of a security context establishment

Please note that according to the GSS API specification successfully established security contexts remain valid independent from the credentials. In particular, credential handles (gss_cred_it_t) can be released without affecting fully established security contexts.

2.1.1 GSS_Acquire_cred

Used to acquire credential information for server processes and for interactive users. May block, depending on product specific implementation.

In the context of SAP R/3, no user interaction is expected with this call.

INPUT:	*desired_name*	`explicit name or default`
	lifetime_req	`indefinite`
	desired_mechs	`taken from the result of the GSS_Indicate_mechs call`
	cred_usage	`initiating or accepting`
OUTPUT:	*major_status*	
	minor_status	
	output_cred_handle	
	actual_mechs	
	lifetime_rec	

2.1.2 GSS_Release_cred

Used to indicate that a credential handle is no longer needed.

Does not have any side effects on existing security contexts as explained above.

INPUT:	*cred_handle*
OUTPUT:	*major_status*
	minor_status

2.1.3 GSS_Inquire_cred

Used to determine information about a credential.

INPUT: *cred_handle*

OUTPUT: *major_status*
minor_status
cred_name
lifetime_rec
cred_usage
mech_set

2.1.4 GSS_Add_cred (not used by SNC)

Used to append elements to an existing credential structure, allowing iterative construction of a multi-mechanism credential.

INPUT: *input_cred_handle*
desired_name
initiator_time_req
acceptor_time_req
desired_mech
cred_usage

OUTPUT: *major_status*
minor_status
output_cred_handle
actual_mechs
initiator_time_rec
acceptor_time_req
cred_usage
mech_set

2.1.5 GSS_Inquire_cred_by_mech

Used to extract per-mechanism information for a credential handle.

INPUT:	*cred_handle*
	mech_type
OUTPUT:	*major_status*
	minor_status
	cred_name
	lifetime_rec_initiate
	lifetime_rec_accept
	cred_usage

2.2 Context-Level Calls

2.2.1 GSS_Init_sec_context

Used by the initiator of a security context. The call results in a security token that is passed to the target.

		(*) derived from the mechanism type OID returned from the call to GSS_Indicate_mechs, all options indicated are requested
INPUT:	*claimant_cred_handle*	
	input_context_handle	
	targ_name	
	mech_type	from GSS_Indicate_mechs call
	deleg_req_flag	no delegation requested
	mutual_req_flag	requested
	replay_det_req_flag	(*)
	sequence_req_flag	not requested
	anon_req_flag	not used
	conf_req_flag	(*)
	integ_req_flag	(*)
	lifetime_req	indefinite
	chan_bindings	not used
	input_token	

OUTPUT:	*major_status*	
	minor_status	
	output_context_handle	
	mech_type	as requested
	output_token	
	deleg_state	as requested
	mutual_state	don't care
	replay_det_state	required if requested
	sequence_state	don't care
	anon_state	don't care
	trans_state	required
	prot_ready_state	not used
	conf_avail	required if requested
	integ_avail	required if requested
	lifetime_rec	Note: returned value is used to control refreshing cycles

2.2.2 GSS_Accept_sec_context

Used by the acceptor of a security context to process the security token information generated by the context initiator.

		(*) as expected from the mechanism type OID returned from the call to GSS_Indicate_mechs
INPUT:	*acceptor_cred_handle*	
	input_context_handle	
	chan_bindings	not used
	input_token	
OUTPUT:	*major_status*	
	minor_status	
	src_name	required
	mech_type	required
	output_context_handle	
	deleg_state	don't care
	mutual_state	don't care

replay_det_state	(*)
sequence_state	don't care
anon_state	don't care
trans_state	required
prot_ready_state	don't care
conf_avail	(*)
integ_avail	(*)
lifetime_rec	
delegated_cred_handle	don't care
output_token	

2.2.3 GSS_Delete_sec_context

Used to delete security context information when no longer required.

Called upon context expiration, connection termination or errors.

INPUT: *context_handle*

OUTPUT: *major_status*

minor_status

output_context_token

2.2.4 GSS_Process_context_token (not used by SNC)

Used to process received security tokens carrying context control information after successful security context establishment.

INPUT: *context_handle*

input_context_token

OUTPUT: *major_status*

minor_status

2.2.5 GSS_Context_time

Allows a caller to determine the length of time for which an established context will remain valid.

Called before each message protection call.

INPUT: *context_handle*

OUTPUT: *major_status*
minor_status
lifetime_rec Note: returned value is used to control refreshing cycles

2.2.6 GSS_Inquire_context

Returns status information describing context characteristics.

INPUT: *context_handle*

OUTPUT: *major_status*
minor_status
src_name
targ_name
lifetime_rec
mech_type
deleg_state
mutual_state
replay_det_state
sequence_state
anon_state
trans_state
prot_ready_state
conf_avail
integ_avail
locally_initiated
open

2.2.7 GSS_Wrap_size_limit

Allows a caller to determine the size of a token which will be generated by a GSS_Wrap() operation.

INPUT: *context_handle*
conf_req_flag
qop
output_size

OUTPUT: *major_status*

minor_status

max_input_size

2.2.8 GSS_Export_sec_context

Enables the export of an active context to be imported by another processes on an end system.

This call is required due to the SAP R/3 process architecture as explained in Chapter 1.

INPUT: *context_handle*

OUTPUT: *major_status*

minor_status

interprocess_token

2.2.9 GSS_Import_sec_context

Enables the import of an active context that was exported by another processes on an end system.

This call is required due to the SAP R/3 process architecture as explained in Chapter 1.

INPUT: *interprocess_token*

OUTPUT: *major_status*

minor_status

context_handle

2.3 Per-Message Calls

2.3.1 GSS_GetMIC

Used to obtain a per-message token containing data items which allow underlying mechanisms to provide data origin authentication and data integrity services.

INPUT: *context_handle*

qop_req `default`

message

OUTPUT:	*major_status*	
	minor_status	
	per_msg_token	

2.3.2 GSS_VerifyMIC

Used to validate the message in conjunction with the security token concerning data origin authentication and data integrity checking.

INPUT:	*context_handle*	
	message	
	per_msg_token	
OUTPUT:	*qop_state*	`don't care`
	major_status	
	minor_status	

2.3.3 GSS_Wrap

Used to provide data integrity protection and caller-requested data confidentiality on a per-message basis, encapsulating the original message into a single output token.

INPUT:	*context_handle*	
	conf_req_flag	`always requested`
	qop_req	`default`
	input_message	
OUTPUT:	*major_status*	
	minor_status	
	conf_state	`required`
	output_message	

2.3.4 GSS_Unwrap

Removes the optional confidentiality protection and verifies the message integrity protection from a token that was produced by GSS_Wrap and returns the original cleartext message.

INPUT:	*context_handle*	
	input_message	

OUTPUT:	*conf_state*	`required`
	qop_state	`don't care`
	major_status	
	minor_status	
	output_message	

2.4 Support Calls

2.4.1 GSS_Display_status

Provides a means for callers to translate GSS-API-returned major and minor status codes into printable string representations.

INPUT: *status_value*

status_type

mech_type

OUTPUT: *major_status*

minor_status

status_string_set

2.4.2 GSS_Indicate_mechs

Allows callers to determine the set of mechanism types available on the local system.

This function is called immediately after the security product has been loaded by the R/3 component. The mechanism set returned is checked for a known OID. The OIDs known to R/3, i.e. the known mechanism types, are used to define the „expectations“ of R/3 concerning the availability of integrity and confidentiality services, replay detection, mutual authentication and the nametype (OID) that is used by the mechanism for the native name syntax and to display canonical printable names. To use R/3 with mechanism types that it doesn't yet recognize, a special „shared library wrapper“ called *SNC-Adapter* must be provided.

INPUT:

OUTPUT: *major_status*

minor_status

mech_set

2.4.3 GSS_Compare_name

Compare two internal name representations to determine whether they refer to the same entity.

INPUT: *name1*

name2

OUTPUT: *major_status*

minor_status

name_equal

2.4.4 GSS_Display_name

Translates an internal name into a printable form with associated namespace type descriptor.

INPUT: *name*

OUTPUT: *major_status*

minor_status

name_string

name_type

2.4.5 GSS_Import_name

Creates an internal name suitable for input to other GSS-API routines from a contiguous octet string representation. The caller uses the nametype parameter to indicate the namespace or syntax of the contiguous octect string representation.

INPUT: *input_name_string*

input_name_type

OUTPUT: *major_status*

minor_status

output_name

2.4.6 GSS_Release_name

Release the storage associated with an internal name.

INPUT: *name*

OUTPUT: *major_status*

minor_status

2.4.7 GSS_Release_buffer

Release the storage associated with an OCTET STRING buffer that was allocated by another GSS-API call.

INPUT: *buffer*

OUTPUT: *major_status*

minor_status

2.4.8 GSS_Release_OID_set

Release the storage associated with an object identifier set object that was allocated by another GSS-API call.

INPUT: *buffer*

OUTPUT: *major_status*

minor_status

2.4.9 GSS_Create_empty_OID_set (not used by SNC)

Creates an object identifier set containing no object identifiers, to which members may be subsequently added using the GSS_Add_OID_set_member() routine.

INPUT:

OUTPUT: *major_status*

minor_status

oid_set

2.4.10 GSS_Add_OID_set_member (not used by SNC)

Adds an Object Identifier to an Object Identifier set. This routine is intended for use in conjunction with GSS_Create_empty_OID_set() when constructing a set of mechanism OIDs for input to GSS_Acquire_cred().

INPUT: *member_oid*

oid_set

OUTPUT: *major_status*

minor_status

2.4.11 GSS_Test_OID_set_member (not used by SNC)

Interrogates an Object Identifier set to determine whether a specified Object Identifier is a member. This routine is intended to be used with OID sets returned by GSS_Indicate_mechs(), GSS_Acquire_cred(), and GSS_Inquire_cred().

INPUT: *member*

set

OUTPUT: *major_status*

minor_status

present

2.4.12 GSS_Inquire_names_for_mech

Allows callers to determine the set of nametypes which are supportable by a specific locally-available mechanism.

INPUT: *input_mech_type*

OUTPUT: *major_status*

minor_status

name_type_set

2.4.13 GSS_Inquire_mechs_for_name (not used by SNC)

Returns the mechanism set with which the input_name may be processed.

INPUT: *input_name*

OUTPUT: *major_status*

minor_status

mech_types

2.4.14 GSS_Canonicalize_name

This routine reduces a GSS-API internal name input_name, which may in general contain elements corresponding to multiple mechanisms, to a mechanism-specific Mechanism Name (MN) output_name by applying the translations corresponding to the mechanism identified by mech_type.

INPUT: *input_name*

mech_type

OUTPUT: *major_status*

minor_status

output_name

2.4.15 GSS_Export_name

This routine creates a flat name representation, suitable for bytewise comparison or for input to GSS_Import_name() in conjunction with the reserved GSS-API Exported Name Object OID, from a internal-form Mechanism Name (MN) as emitted, e.g., by GSS_Canonicalize_name() or GSS_Accept_sec_context().

INPUT: *input_name*

OUTPUT: *major_status*

minor_status

output_name

2.4.16 GSS_Duplicate_name (not used by SNC)

This routine duplicates an internal name. The resulting destination name exists independent of the source name and it must be independently released when it is no longer needed.

INPUT: *src_name*

OUTPUT: *major_status*

minor_status

dest_name

3. Non-functional Requirements for SNC

to be written

3.1 Application Binary Portability

to be written

3.2 Token Size / Growth Restrictions

Most application level protocols of R/3 impose limits on the size of the messages that can be passed around. This results token size restrictions for context-level tokens emitted by GSS_Init_Sec_Context and GSS_Accept_Sec_Context, for MIC tokens emitted by GSS_Get_MIC and for the message growth resulting from the encapsulation by GSS_Wrap.

These are the interoperability requirements for SAP R/3:

Context-level Tokens	<= 25,000 Bytes
MIC Tokens	<= 1,500 Bytes
Wrap message growth	<= 1,500 Bytes

The message protection services need to support message sizes in the range [0..63,999] Bytes and may not exceed above limits for messages within this range.

3.3 Performance Requirements

CPU Time is often a scarce resource on R/3 application servers. Data protection with software-based cryptographic algorithms also needs CPU Time. To predict the additional hardware requirements/costs for running a certain R/3 installation with SNC-protected communication, one will need to know the performance impact caused by the authentication and data protection services. Since the performance impact of the data protection depends only on the amount of communicated data and does not affect the effort to process or create the data, it is not possible to come up with a single or precise number.

The following table gives some rough figures for a 10-35 % Performance Impact on the SAPgui-related load factor for on a Intel Pentium-90 class hardware.

GSS-API routine	CPU Time (Pentium 90 class)
GSS_Init_Sec_Context (1-3x)	600 millisec
GSS_Accept_Sec_Context (1-2x)	per context establishment
GSS_Export_Sec_Context (1x)	2 millisec
GSS_Import_Sec_Context (1x)	2 millisec
GSS_Wrap (1x)	1 Kbyte / millisec
GSS_Unwrap (1x)	1 Kbyte / millisec
GSS_Get_MIC (1x)	4 Kbyte / millisec
GSS_Verify_MIC (1x)	4 Kbyte / millisec
GSS_Context_Time (1x)	0.2 millisec

3.4 Memory Management

to be written

4. How to build the SNC Adapter

to be written

5. References

[1] "SAP Technology Infrastructure", Brochure, http://www.sap-ag.de/products/techno/index.htm

[2] "Secure Network Communications and Secure Store & Forward Mechanisms with the SAP R/3 System", White Paper, http://www.sap-ag.de/products/system/index.htm

[3] "Generic Security Service Application Program Interface", IETF Working Draft RFC2078bis, Version 2, Update 1

[4] "Generic Security Service API Version 2 : C-bindings", IETF Working Draft RFC2078bis

[5] "SNC User's Guide"