ESROS
Application Programming Interface

Neda Document Number: 103-101-06.03

Last Updated: Author unspecified

Doc. Revision: source unspecified

Neda Communications, Inc.

January 27, 1999

Contents

List of Tables

List of Figures

This document describes the Application Programming Interface for Efficient Short Remote Operation Services (ESROS).

Published by:
Neda Communications, Inc
17005 SE 31st Place
Bellevue, WA 98008

Figure 1: Implementation Architecture

1 Introduction

1.1 About This API

This document defines the ESROS' API. This definition conforms to RFC-2188 [1]. It is recommended that for this document to be of the most use to the reader, they should be familiar with RFC-2188[2] and Open C Platform [2].

Chapter 1 consists of an introduction to the API and the whole document.

Chapter 2 provides information about the interface to ESROS services.

Appendices include a Bibliography, a list of relevant Acronyms, ESROS API Example Usage, ESROS Program Man Pages.

1.2 Architecture

Figure 1, depicts the architecture of the complete ESRO protocols. ESROS-Daemon is responsible for implementation of ESRO-Protocol (RFC-2188. [1]) on both invoker and performer sides. ESROS-Daemon exposes the ESROS API (see chapter entitled ESRO API) to its users.

This chapter provides information about the interface to ESROS services. It is intended for the users of the ESROS sublayer.

The ESROS API is available in two different styles. In the first case the events are made available to the user of the API through function calls. This is known as the Function Call API. Functions of this API implementation all have the ESRO_ prefix. In the second case ESROS events trigger call backs to functions registered by the user of the ESROS API. This is known as Call back API. Functions of this API implementation all have the ESRO_CB_ prefix, in which CB stands for Call Back.

1.3 ESRO Service Primitives

This section describes the service primitives provided by the ESROP module, and the constraints on the sequence in which the ESROP primitives may occur. Each ESROP-User interacts with the ESROP module through one or more ESROP-SAPs.

Table 1 is a list of ESRO service primitive names.

The Neda ESROP upper interface conforms to the ESRO Service Definition [2]. The constraints on the sequence in which ESROP primitives may occur are explained in Reference [2].

1.3.1 SAP Management

An ESROP-User must create an ESROP-SAP before it can use any of the services provided by the ESROP module. Creation of an ESROP-SAP is accomplished through the ESROP_sapBind function. Parameters to ESROP_sapBind communicate to the ESROP module both an ESRO-SAP selector address and a set of functions for handling event primitives for that ESROP-SAP. ESROP event primitives are:

ESRO Service Primitives
ESROS-INVOKE.request ESROS-INOVKE-P.confirm ESROS-INVOKE.indication
ESROS-RESULT.request ESROS-RESULT.indication ESROS-RESULT.confirm
ESROS-ERROR.request ESROS-ERROR.indication ESROS-ERROR.confirm
ESROS-FAILURE.indication

Table 1: ESRO Service Primitives

Function	Description
ESROP_sapBind	Bind an ESROP-SAP and register an ESROP-User.
ESROP_sapUnbind	Unbind an ESROP-SAP and deregister an ESROP-User.

Table 2: ESROS-SAP Management

- ESROS-INVOKE.indication
- ESROS-RESULT.indication
- ESROS-ERROR.indication
- ESROS-FAILURE.indication

Deletion of an ESROP-SAP is accomplished through the ESROP_sapUnbind function. A summary of Neda ESROP-SAP management facilities follows.

1.3.2 Operation Invocation

The sequence of ESROP primitives in an OPERATION is illustrated in the time sequence diagram below.

To initiate an ESROP operation, the invoker ESROP-User entity issues an ESROS-INVOKE.request at the ESROP layer interface by invoking the function ESROP_invokeReq. The performer ESROP entity's ESROP-SAP is specified as one of the parameters of this action primitive.

The ESROS-INVOKE-P.confirm primitive is communicated to the invoker user through the return value/parameter of the ESROP_invokeReq function.

Figure 2: Time sequence diagram for ESRO Services

Service Primitive Name	Neda Function Name	Source
ESROESROS-INVOKE.request	ESROP_invokeReq()	Invoker user
ESROS-INVOKE-P.confirm	Ret Val of ESROP_invokeReq()	Provider
ESROS-INVOKE.indication	(*ESROP_invokeInd)()	Provider
ESROS-RESULT.request	ESROP_resultReq()	Performer user
ESROS-RESULT.indication	(*ESROP_resultInd)()	Provider
ESROS-RESULT.confirm	(*ESROP_resultCnf)()	Provider
ESROS-ERROR.request	ESROP_errorReq()	Performer user
ESROS-ERROR.indication	(*ESROP_errorInd)()	Provider
ESROS-ERROR.confirm	(*ESROP_errorCnf)()	Provider
ESROS-FAILURE.indication	(*ESROP_failureInd)()	Provider

Table 3: Service Primitives and corresponding functions

An ESROS-INVOKE.indication event primitive is generated at the performer ESROP entity's ESROP-SAP through the invocation of the (*ESROP_invokeInd)() function associated with the performer ESROP-SAP.

The performer ESROP-User can accept the operation and communicate the results by generating an ESROS-RESULT.request at the ESROP layer interface by invoking the function ESROP_resultReq. The performer ESROP-User can issue an ESROS-ERROR.request by invoking the function ESROP_errorReq.

An ESROS-RESULT.confirm or ESROS-ERROR.confirm event primitive is generated at the performer ESROP entity ESROP-SAP through the invocation of the (*ESROP_resultCnf)() or (*ESROP_errorCnf)() function associated with the performer ESROP-SAP.

An ESROS-RESULT.indication or ESROS-ERROR.indication event primitive is generated at the invoker ESROP entity ESROP-SAP through the invocation of the (*ESROP_resultInd)() or (*ESROP_errorInd)() function associated with the invoker ESROP-SAP.

A summary of all operation primitives appears below in Table 3:

The OPERATION may fail due to either the inability of the ESROS provider to transmit the INVOKE PDU or the unwillingness of the ESROS performer user to accept an ESROS-INVOKE.indication. These cases are described later in this chapter. The OPERATION may also fail as a result of the failure in delivery of RESULT or ERROR PDU. In such cases an ESROS-FAILURE.indication event primitive is issued at the invoker or performer ESROP-SAP through the invocation of the (*ESROP_failureInd)() function.

1.4 ESROS With Function Call API

This section provides information about the Function Call API.

The services provided by the ESROS are defined in the ESROS Protocol Specification. The requests and responses are communicated via non-blocking function calls. Remote operation requests, and error and failure indications are communicated to the ESROS user via a call to the ESRO_getEvent function, which may be a blocking call in some implementations.

Remote operation requests, result, error and failure indications are delivered to the ESROS user in an event structure. The reader should consult the following chapters for information about the parameters which make up the structures.

The following subsections describe the ESROS library functions.

1.4.1 Initialize the Parameters

```
PUBLIC ESRO_RetVal
ESRO_init (String configFileName)
```

The argument is defined as follows:

```
configFileName  Config file name
```

configFileName specifies the config file name that contains ESROS initialization values.

1.4.2 Activate ESROS Service Access Point

The ESRO_sapBind function binds an ESRO Service Access Point (ESRO_SAP) to the current user process. It has the following syntax:

```
PUBLIC ESRO_RetVal
ESRO_sapBind(ESRO_SapDesc*sapDesc, /* out */
ESRO_SapSelsapSel
ESRO_FunctionalUnitfunctionalUnit)
```

The arguments are defined as follows:

```
sapDesc      Return value: the SAP descriptor
sapSel       SAP selector
functionalUnitHandshaking   type
```

sapDesc is a pointer to an ESRO_SapDesc structure that is created for the current user.

sapSel identifies the ESROS SAP. If the SAP is in use by another user the function returns an error value.

functionalUnit specifies the type of handshaking that is in effect for the SAP. ESRO_2Way specifies two-way handshaking. ESRO_3Way specifies three-way handshaking. In order for ESROS user processes to interact with one another over a network, they must specify local SAPs that use the same type of handshaking. Furthermore, once a SAP is created the handshaking type stays in effect until the SAP is released. Once an ESRO-SAP has been activated, the user process can use the services provided by ESROS sublayer.

The function returns zero if successful, otherwise it returns a nonzero error value.

1.4.3 Deactivate ESROS Service Access Point

The ESRO_sapUnbind function deactivates the ESROs service access point which is currently in use. It has the following syntax:

```
PUBLIC ESRO_RetVal
ESRO_sapUnbind(ESRO_SapSel sapSel)
```

The argument is defined as follows:

```
sapSel  SAP selector
```

sapSel identifies the ESROS SAP which is already in use.
The function would return 0 if successful, and a nonzero error value otherwise.

1.4.4 ESROS Invoke Service Request

The ESRO_invokeReq function requests a remote operation. It has the following syntax:

```
PUBLIC ESRO_RetVal
ESRO_invokeReq( ESRO_InvokeId*invokeId,/* out */
ESRO_UserInvokeRefuserInvokeRef,
ESRO_SapDesclocSapDesc,
ESRO_SapSelremESROSap,
T_SapSel*remTsap,
N_SapAddr*remNsap,
ESRO_OperationValueopValue,
ESRO_EncodingTypeencodingType,
IntparameterLen,
Byte*parameter)
```

The input arguments are defined as follows:

```
invokeId               Return value: invocation identifier
userInvokeRef          User's invocation reference
locSapDesc             The local SAP descriptor
remESROSap             Remote network SAP address
remTsap                Remote Transport SAP.
remNsap                The remote SAP selector
opValue                Operation value
encodingType           Encoding type
parameterLen           The length of the parameter
parameter              The address of the parameter buffer.
```

invokeId is assigned by ESROS sublayer. It is returned by ESROS sublayer. This identifier is used in future communications between ESROS sublayer and service user to identify the invocation for ESROS sublayer.

userInvokeRef is assigned by ESROS user. It is passed to ESROS sublayer by the user of service. This identifier is used in future communications between ESROS sublayer and service user to identify the invocation for the user of ESROS.

locSapDesc is the local SAP descriptor which is provided by ESROS sublayer at the time of SAP bind.

If ESROS can serve the invoker, the function returns 0 and the invocation identifier is returned through the invokeId parameter. If ESROS cannot serve the invoker, the function returns a nonzero failure reason value.

1.4.5 ESROS Result Service Request

The ESRO_resultReq function is issued by the performer of the operation. It has the following syntax:

```
PUBLIC ESRO_RetVal
ESRO_resultReq( ESRO_InvokeIdinvokeId,
ESRO_UserInvokeRefuserInvokeRef,
ESRO_EncodingTypeencodingType,
IntparameterLen,
Byte*parameter)
```

The input arguments are defined as follows:

```
invokeId        Invocation Identifier.
userInvokeRef   User's invocation reference
encodingType    Encoding type
parameterLen    Length of the parameter
parameter       Address of the parameter buffer.
```

This primitive should be issued after an ESRO_INVOKEIND event. If ESROS cannot serve the requestor, the function returns a nonzero reason value which is the failure value.

1.4.6 ESROS Error Service Request

The ESRO_errorReq function is issued by the performer of the operation in case of error in performing the operation. It has the following syntax:

```
PUBLIC ESRO_RetVal
ESRO_errorReq(  ESRO_InvokeIdinvokeId,
ESRO_UserInvokeRefuserInvokeRef,
ESRO_EncodingTypeencodingType,
ESRO_ErrorValueerrorValue,
IntparameterLen,
Byte*parameter)
```

The input arguments are defined as follows:

```
invokeId        The Invocation Identifier.
userInvokeRef   User's invocation reference
encodingType    Encoding type
errorValue      Identifies the nature of the error.
parameterLen    The length of the parameter
parameter       String describing the error.
```

This primitive should be issued after a INVOKEIND event. If esros cannot serve the requestor, the function returns a negative value which is the failure value.

1.4.7 Get an event

If any event has occurred in ESROS sublayer, the *ESRO_getEvent* function gets the event(s). Based on the value of *wait*, it either waits for and event (if no event available) or immediately returns.

```
PUBLIC ESRO_RetVal ESRO_getEvent( ESRO_SapDesc sapDesc,
ESRO_Event *event, Bool wait)
```

The input arguments are defined as follows:

```
sapDesc Return value: the SAP descriptor
event   ESROS event
wait    Blocking/non-blocking flag
```

The function returns any of the following event codes as the corresponding events are detected:

```
ESRO_INVOKEIND  Remote user is requesting an operation
ESRO_FAILUREIND Operation has failed
ESRO_RESULTIND  ESRO-RESULT-PDU recieved
ESRO_ERRORIND   ESRO-ERROR-PDU received
ESRO_RESULTCNF  ESROS RESULT confirm
ESRO_ERRORCNF   ESROS ERROR confirm
```

The function returns negative error number if unsuccessful, or the number of events (0 or greater than 0).

The data structures of ESROS events and the corresponding event codes are listed below:

1.4.8 Sample Code

The code fragments described in the following sections illustrate the steps required to create a ESRO service access point, and invoke and perform an operation. They are patterned after the primitives of the time sequence in , Example of time sequence diagram for ESROS Services. The code fragments themselves are listed in , ESRO API Example Usage. The code sample "invoker.c" implements the left side, and the code sample "performer.c" implements the right side.

Figure 3: Example of time sequence diagram for ESROS Services

invoker.c

invoker.c first establishes a SAP, then issues an ESRO_invokeReq of a shell command operation. In this example, the command operation is ”date”. It receives a confirmation (ESROESRO_ResultInd) indicating that the operation was performed. It then retrieves the results which are communicated through the ESRO_ResultInd.

performer.c

performer.c receives the ESRO_InvokeInd of a ”date” command operation in the struct ESRO_InvokeInd. The result of the command is the system date which is returned to invoker.c through ESRO_resultReq. performer.c then waits for the next request from invoker.c.

1.5 ESROS With Callback API

This section provides information about the callback API functions.

The services provided by the ESROS are defined in the ESROS Protocol Specification,”RFC-2188” [1]. The requests are issued through function calls. Callback functions associated with ESROS events are passed to ESROS at the time of sapBind function call.

The following subsections describe the ESROS library functions

1.5.1 Initialize the Parameters

```
PUBLIC ESRO_RetVal
ESRO_CB_init (String configFileName)
```

The argument is defined as follows:

```
configFileName  Config file name
```

configFileName specifies the config file name that contains ESROS initialization parameters.

1.5.2 Activate ESROS Service Access Point

The ESRO_CB_sapBind function binds an ESRO Service Access Point (ESRO_SAP) for the current user process. It has the following syntax:

```
PUBLIC ESRO_RetVal
ESRO_CB_sapBind(
ESRO_SapDesc    *sapDesc,
ESRO_SapSel     sapSel,
```

```
ESRO_FunctionalUnit     functionalUnit,
int (*invokeInd)(       ESRO_SapDesc    locSapDesc,
ESRO_SapSel     remESROSap,
T_SapSel        *remTsap,
N_SapAddr       *remNsap,
ESRO_InvokeId   invokeId,
ESRO_OperationValue     opValue,
ESRO_EncodingType       encodingType,
DU_View parameter),
int (*resultInd) (      ESRO_InvokeId   invokeId,
ESRO_UserInvokeRef      userInvokeRef,
ESRO_EncodingType       encodingType,
DU_View parameter),
int (*errorInd) (       ESRO_InvokeId   invokeId,
ESRO_UserInvokeRef      userInvokeRef,
ESRO_EncodingType       encodingType,
ESRO_ErrorValue errorValue,
DU_View parameter),
int (*resultCnf)(       ESRO_InvokeId   invokeId,
ESRO_UserInvokeRef      userInvokeRef),
int (*errorCnf) (       ESRO_InvokeId   invokeId,
ESRO_UserInvokeRef      userInvokeRef),
int (*failureInd)(      ESRO_InvokeId   invokeId,
ESRO_UserInvokeRef      userInvokeRef,
ESRO_FailureValue       failureValue))
```

The input arguments are defined as follows:

```
sapDesc Local SAP descriptor (outgoing param)
sapSel  Local SAP selector
functionalUnit  Handshaking type
locSapDesc      Local SAP descriptor
remESROSap      Remote network SAP address
remTsap Rmote Transport SAP.
remNsap The remote SAP selector
invokeId        Invocation identifier
userInvokeRef   User's invocation reference
opValue Operation value
encodingType    Encoding type
errorValue      Error value
failureValue    Failure value
parameter       parameter.

(*invokeInd)()  Invoke indication function
```

```
(*resultInd)()  Result indication function
(*errorInd)()   Error indication function
(*resultCnf)()  Result confirmation function
(*errorCnf)()   Error confirmation function
(*failureInd)() Failure indication function
```

sapDesc is a pointer to an ESRO_SapDesc structure that is created for the current user.

sapSel identifies the ESROS SAP. If the SAP is in use by another user the function returns an error value.

functionalUnit specifies the type of handshaking that is in effect for the SAP. ESRO_2Way specifies two-way handshaking. ESRO_3Way specifies three-way handshaking. In order for ESROS user processes to interact with one another over a network, they must specify local SAPs that use the same type of handshaking. Furthermore, once a SAP is created the handshaking type stays in effect until the SAP is released. Once an ESRO-SAP has been activated, the user process can use the services provided by ESROS.

After its ESRO-SAP has been activated, the user process can use the services provided by ESROS.

The function returns zero if successful, otherwise it returns a nonzero error value.

1.5.3 Deactivate ESROS Service Access Point

The ESRO_CB_sapUnbind function deactivates the ESROs service access point which is currently in use. It has the following syntax:

```
PUBLIC ESRO_RetVal
ESRO_sapUnbind( ESRO_SapSel     sapSel)
```

The argument is defined as follows:

```
sapSel  SAP selector
```

sapSel identifies the ESROS SAP which is already in use.

The function would return 0 if successful, and a nonzero error value otherwise.

1.5.4 ESROS Invoke Service Request

The ESRO_CB_invokeReq function requests a remote operation. It has the following syntax:

```
PUBLIC ESRO_RetVal
ESRO_CB_invokeReq(      ESRO_InvokeId   *invokeId,/* out */
ESRO_UserInvokeRef      userInvokeRef,
ESRO_SapDesc    locSapDesc,
ESRO_SapSel     remESROSap,
T_SapSel        *remTsap,
N_SapAddr       *remNsap,
ESRO_OperationValue     opValue,
ESRO_EncodingType       encodingType,
DU_View parameter)
```

The input arguments are defined as follows:

```
invokeId        Return value: invocation identifier
userInvokeRef   User's invocation reference
locSapDesc      The local SAP descriptor
remESROSap      Remote network SAP address
remTsap Rmote Transport SAP
remNsap The remote SAP selector
opValue Operation value
encodingType    Encoding type
parameter       user data
```

invokeId is assigned by ESROS sublayer. It is returned by ESROS sublayer and identifies an invocation for ESROS sublayer. This identifier is used in future communications between ESROS sublayer and service user to identify the invocation for ESROS sublayer.

userInvokeRef is assigned by ESROS user. It is passed to ESROS sublayer by the user of service. This identifier is used in future communications between ESROS sublayer and service user to identify the invocation for the user of ESROS.

locSapDesc is the local SAP descriptor which is provided by ESROS sublayer at the time of SAP bind.

parameter is a pointer to a DU_view data structure into which user data was previously copied. Refer to the Open C Platform document [2] for a discussion of the DU_ module.

If ESROS can serve the invoker, the function returns 0 and the invocation identifier is returned through the invokeId parameter. If ESROS cannot serve the invoker, the function returns a nonzero failure reason value.

1.5.5 ESROS Result Service Request

The ESRO_CB_resultReq function is issued by the performer of the operation. It has the following syntax:

```
PUBLIC ESRO_RetVal
ESRO_CB_resultReq(      ESRO_InvokeId   invokeId,
ESRO_UserInvokeRef      userInvokeRef,
ESRO_EncodingType       encodingType,
DU_View parameter)
```

The input arguments are defined as follows:

```
invokeId        invocation Identifier
userInvokeRef   User's invocation reference
encodingType    Encoding type
parameter       Parameter.
```

This primitive should be issued after invokeInd function is called. If ESROS cannot serve the requestor, the function returns a nonzero reason value which is the failure value.

1.5.6 ESROS Error Service Request

The ESRO_CB_errorReq function is issued by the performer of the operation in case of error in performing the operation. It has the following syntax:

```
PUBLIC ESRO_RetVal
ESRO_CB_errorReq(       ESRO_InvokeId    invokeId,
ESRO_UserInvokeRef      userInvokeRef,
ESRO_EncodingType       encodingType,
ESRO_ErrorValue errorValue,
DU_View parameter)
```

The input arguments are defined as follows:

```
invokeId        The Invocation Identifier
userInvokeRef   User's invocation reference
encodingType    Encoding type
errorValue      Error value
parameter       Parameter.
```

This primitive should be issued after invokeInd function is called. If ESROS cannot serve the requestor, the function returns a negative value which is the failure value.

1.5.7 Sample Code

The code fragments described in the following sections illustrate the steps required to create a ESRO service access point, and invoke and perform an operation. They are patterned after the primitives of the time sequence in , Example of time sequence diagram for ESROS CB Services. The code fragments themselves are listed in , ESRO API Example Usage. The code sample ”invoksch.c” implements the left side, and the code sample ”perfsch.c” implements the right side.

invoksch.c

invoksch.c first establishes a SAP, then issues an ESRO_invokeReq of a shell command operation. In this example, the command operation is ”date”. The resultInd function is called indicating that the operation was performed and the result is passed to it through data parameter.

perfsch.c

perfsch.c establishes a SAP and waits for a request from invoksch.c. The invokeInd function is called when the request for a command operation arrives. The result of the ”date” command is the system date. perfsch.c then returns the data to invoksch.c through ESRO_resultReq. perfsch.c then waits for the next request from invoksch.c.

Figure 4: Example of time sequence diagram for ESROS CB Services

A Acronyms

ASN.1	Abstract Syntax Notation One (ASN.1)
FSM	ESROS Finite State Machine.
IP-Message	InterPersonal Message
ESROS	Efficient Short Remote Operation Services
ESROP	ESROS Protocol Engine
ESRO-SAP	ESROS Service Access Point.
MD	Management Domain
MH	Message Handling
MHS	Message Handling System
MS	Message Store
MT	Message Transfer
MTA	Message Transfer Agent
MTS	Message Transfer Service
SEQ_	Sequence Module
TMR_	Timer Management Module
TM_	Trace Module
DU_	Data Unit Management Module

B ESRO API Example Usage

B.1 invoker.c

B.2 invoksch.c

B.3 performer.c

B.4 perfsch.c

References

[1] M. Banan, M. Taylor, and J. Cheng. AT&T/Neda's Efficient Short Remote Operations (ESRO) Protocol Specification Version 1.2. RFC 2188 (Informational), September 1997.

[2] Neda Public Document. *Open C Platform.* Neda Published Document 103-103-01, Neda Communications Inc, Bellevue, WA, October 1996. Online document is available at http://www.mailmeanywhere.org/sw.free/neda/foundations/ocp/OCP-MulPub/accessPage.html.