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

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Introduction</td>
<td>5</td>
</tr>
<tr>
<td>1.1 About This API</td>
<td>5</td>
</tr>
<tr>
<td>1.2 Architecture</td>
<td>5</td>
</tr>
<tr>
<td>1.3 ESRO Service Primitives</td>
<td>5</td>
</tr>
<tr>
<td>1.3.1 SAP Management</td>
<td>6</td>
</tr>
<tr>
<td>1.3.2 Operation Invocation</td>
<td>6</td>
</tr>
<tr>
<td>1.4 ESROS With Function Call API</td>
<td>8</td>
</tr>
<tr>
<td>1.4.1 Initialize the Parameters</td>
<td>8</td>
</tr>
<tr>
<td>1.4.2 Activate ESROS Service Access Point</td>
<td>8</td>
</tr>
<tr>
<td>1.4.3 Deactivate ESROS Service Access Point</td>
<td>9</td>
</tr>
<tr>
<td>1.4.4 ESROS Invoke Service Request</td>
<td>9</td>
</tr>
<tr>
<td>1.4.5 ESROS Result Service Request</td>
<td>10</td>
</tr>
<tr>
<td>1.4.6 ESROS Error Service Request</td>
<td>10</td>
</tr>
<tr>
<td>1.4.7 Get an event</td>
<td>11</td>
</tr>
<tr>
<td>1.4.8 Sample Code</td>
<td>12</td>
</tr>
<tr>
<td>1.5 ESROS With Callback API</td>
<td>12</td>
</tr>
<tr>
<td>1.5.1 Initialize the Parameters</td>
<td>12</td>
</tr>
<tr>
<td>1.5.2 Activate ESROS Service Access Point</td>
<td>13</td>
</tr>
<tr>
<td>1.5.3 Deactivate ESROS Service Access Point</td>
<td>14</td>
</tr>
<tr>
<td>1.5.4 ESROS Invoke Service Request</td>
<td>14</td>
</tr>
<tr>
<td>1.5.5 ESROS Result Service Request</td>
<td>15</td>
</tr>
<tr>
<td>1.5.6 ESROS Error Service Request</td>
<td>16</td>
</tr>
<tr>
<td>1.5.7 Sample Code</td>
<td>16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A Acronyms</td>
<td>18</td>
</tr>
<tr>
<td>B ESRO API Example Usage</td>
<td>19</td>
</tr>
<tr>
<td>B.1 invoker.c</td>
<td>19</td>
</tr>
<tr>
<td>B.2 invoksch.c</td>
<td>19</td>
</tr>
<tr>
<td>B.3 performer.c</td>
<td>19</td>
</tr>
<tr>
<td>B.4 perfsch.c</td>
<td>19</td>
</tr>
</tbody>
</table>

List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 ESRO Service Primitives</td>
<td>6</td>
</tr>
<tr>
<td>2 ESROS-SAP Management</td>
<td>6</td>
</tr>
<tr>
<td>3 Service Primitives and corresponding functions</td>
<td>7</td>
</tr>
</tbody>
</table>
List of Figures

1 Implementation Architecture .................................................. 5
2 Time sequence diagram for ESRO Services .......................... 7
3 Example of time sequence diagram for ESROS Services ............ 12
4 Example of time sequence diagram for ESROS CB Services ......... 17
IBM is a registered trademark of International Business Machines Corporation. Unix is a trademark of AT&T and Unix System Laboratories. Sun is a registered trademark and Sun Workstation is a trademark of Sun Microsystems, Inc. Windows is a registered trademark of Microsoft Corporation.

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

Permission is granted to make copy and distribute verbatim copies of this manual provided the copyright notice and this permission notice are preserved on all copies.

Permission is granted to copy and distribute translations of this manual into another language provided the copyright notice and this permission notice are preserved on all copies.
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
ESRO Service Primitives

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESROS-INVOKE.request</td>
<td>Bind an ESRO-SAP and register an ESRO-User.</td>
</tr>
<tr>
<td>ESROS-INVOKE-P.confirm</td>
<td></td>
</tr>
<tr>
<td>ESROS-INVOKE.indication</td>
<td></td>
</tr>
<tr>
<td>ESROS-RESULT.request</td>
<td>Delete an ESRO-SAP by binding the ESRO-SAP selector address and registering the ESRO-User.</td>
</tr>
<tr>
<td>ESROS-RESULT.indication</td>
<td></td>
</tr>
<tr>
<td>ESROS-RESULT.confirm</td>
<td></td>
</tr>
<tr>
<td>ESROS-ERROR.request</td>
<td></td>
</tr>
<tr>
<td>ESROS-ERROR.indication</td>
<td></td>
</tr>
<tr>
<td>ESROS-ERROR.confirm</td>
<td></td>
</tr>
<tr>
<td>ESROS-FAILURE.indication</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: ESRO Service Primitives

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESROP_sapBind</td>
<td>Bind an ESROP-SAP and register an ESROP-User.</td>
</tr>
<tr>
<td>ESROP_sapUnbind</td>
<td>Unbind an ESROP-SAP and deregister an ESROP-User.</td>
</tr>
</tbody>
</table>

Table 2: ESROS-SAP Management

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:

- 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.

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.

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

Table 3: Service Primitives and corresponding functions

<table>
<thead>
<tr>
<th>Service Primitive Name</th>
<th>Neda Function Name</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESROS-INVOKE.request</td>
<td>ESROP_invokeReq()</td>
<td>Invoker user</td>
</tr>
<tr>
<td>ESROS-INVOKE-P.confirm</td>
<td>Ret Val of ESROP_invokeReq()</td>
<td>Provider</td>
</tr>
<tr>
<td>ESROS-INVOKE.indication</td>
<td>(*ESROP_invokeInd)()</td>
<td>Provider</td>
</tr>
<tr>
<td>ESROS-RESULT.request</td>
<td>ESROP_resultReq()</td>
<td>Performer user</td>
</tr>
<tr>
<td>ESROS-RESULT.indication</td>
<td>(*ESROP_resultInd)()</td>
<td>Provider</td>
</tr>
<tr>
<td>ESROS-RESULT.confirm</td>
<td>(*ESROP_resultCnf)()</td>
<td>Provider</td>
</tr>
<tr>
<td>ESROS-ERROR.request</td>
<td>ESROP_errorReq()</td>
<td>Performer user</td>
</tr>
<tr>
<td>ESROS-ERROR.indication</td>
<td>(*ESROP_errorInd)()</td>
<td>Provider</td>
</tr>
<tr>
<td>ESROS-ERROR.confirm</td>
<td>(*ESROP_errorCnf)()</td>
<td>Provider</td>
</tr>
<tr>
<td>ESROS-FAILURE.indication</td>
<td>(*ESROP_failureInd)()</td>
<td>Provider</td>
</tr>
</tbody>
</table>

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_FunctionalUnit functionalUnit)

The arguments are defined as follows:

sapDesc Return value: the SAP descriptor
sapSel SAP selector
functionalUnit Handshaking 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:

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>invokeId</td>
<td>Return value: invocation identifier</td>
</tr>
<tr>
<td>userInvokeRef</td>
<td>User's invocation reference</td>
</tr>
<tr>
<td>locSapDesc</td>
<td>The local SAP descriptor</td>
</tr>
<tr>
<td>remESROSap</td>
<td>Remote network SAP address</td>
</tr>
<tr>
<td>remTsap</td>
<td>Remote Transport SAP.</td>
</tr>
<tr>
<td>remNsap</td>
<td>The remote SAP selector</td>
</tr>
<tr>
<td>opValue</td>
<td>Operation value</td>
</tr>
</tbody>
</table>
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:

```c
PUBLIC ESRO_RetVal
ESRO_resultReq( ESRO_InvokeId invokeId,
                 ESRO_UserInvokeRef userInvokeRef,
                 ESRO_EncodingType encodingType,
                 Int parameterLen,
                 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:

```c
PUBLIC ESRO_RetVal
ESRO_errorReq( ESRO_InvokeId invokeId,
                ESRO_UserInvokeRef userInvokeRef,
                ESRO_EncodingType encodingType,
                Int parameterLen,
                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 an event (if no event available) or immediately returns.

```c
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 received
- **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 Figure 3, 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.

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:

```c
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` Remote Transport SAP
- `remNsap` The remote SAP selector
- `invokeId` Invocation identifier
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:

```c
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:
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
- **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:
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:

```c
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.
## A  Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASN.1</td>
<td>Abstract Syntax Notation One (ASN.1)</td>
</tr>
<tr>
<td>FSM</td>
<td>ESROS Finite State Machine.</td>
</tr>
<tr>
<td>IP-Message</td>
<td>InterPersonal Message</td>
</tr>
<tr>
<td>ESROS</td>
<td>Efficient Short Remote Operation Services</td>
</tr>
<tr>
<td>ESROP</td>
<td>ESROS Protocol Engine</td>
</tr>
<tr>
<td>ESRO-SAP</td>
<td>ESROS Service Access Point.</td>
</tr>
<tr>
<td>MD</td>
<td>Management Domain</td>
</tr>
<tr>
<td>MH</td>
<td>Message Handling</td>
</tr>
<tr>
<td>MHS</td>
<td>Message Handling System</td>
</tr>
<tr>
<td>MS</td>
<td>Message Store</td>
</tr>
<tr>
<td>MT</td>
<td>Message Transfer</td>
</tr>
<tr>
<td>MTA</td>
<td>Message Transfer Agent</td>
</tr>
<tr>
<td>MTS</td>
<td>Message Transfer Service</td>
</tr>
<tr>
<td>SEQ</td>
<td>Sequence Module</td>
</tr>
<tr>
<td>TMR</td>
<td>Timer Management Module</td>
</tr>
<tr>
<td>TM</td>
<td>Trace Module</td>
</tr>
<tr>
<td>DU</td>
<td>Data Unit Management Module</td>
</tr>
</tbody>
</table>
B  ESRO API Example Usage

B.1  invoker.c
B.2  invoksch.c
B.3  performer.c
B.4  perfsch.c
References
