Network Working Group
Request for Comments: 2188
Category: Informational

M. Banan, Neda
M. Taylor, AWS
J. Cheng, AWS

September 1997

AT&T/Neda's Efficient Short Remote Operations (ESRO)
Protocol Specification Version 1.2

This Is A Copy

The Postscript, PDF and HTML formats of RFC-2188 are not the authoritative RFC. The formal, authoritative publication of RFC-2188 is the text format published at the online RFC library.

The author has made every effort to ensure that the Postscript, PDF and HTML formats contain the exact same text as the original. If there are any discrepancies between the Postscript, PDF, HTML formats and the original text version, the original text version is always authoritative.

Status of this Memo:

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

IESG Note

This protocol has not had the benefit of IETF Working Group review, but a cursory examination reveals several issues which may be significant issues for scalability. A site considering deployment should conduct a careful analysis to ensure they understand the potential impacts.

Abstract:

This document specifies the service model, the notation and protocol for Efficient Short Remote Operations (ESRO). The ESRO service is similar to and is consistent with other Remote Procedure Call services. The emphasis of ESRO service definition and the ESRO protocol is on efficiency. ESRO is designed specifically with wireless network (e.g., CDPD) usage in mind.

ESRO protocol provides reliable connectionless remote operation services on top of UDP (or any other non-reliable connectionless transport service) with minimum overhead. ESRO protocol supports segmentation and reassembly, concatenation and separation as well as multiplexing for service users (applications).

ESRO allows for trade-offs between efficiency and reliability by specifying both 2-way hand-shake and 3-way hand-shake based protocols.

Encoding mechanisms for presentation of the parameters of remote operations are outside the scope of this document. But, identification (tagging) of the encoding mechanism in use (e.g., XDR, BER, PER) is supported by ESRO protocol.

A variety of applications can use the ESRO protocol. Some early applications using ESRO include efficient short message submission and delivery, credit card authorization and white pages lookup.

Contents

1 INTRODUCTION

Efficient Short Remote Operations (ESRO) provide an efficent mechanism for realization of Remote Procedure Call. This document specifies many aspects of ESRO including:

- Service Model
- Service Primitives
- A Notation for user of the Service
- Confirmed Connectionless Protocol (based on a 3-way hand-shake)
- Unconfirmed Connectionless Protocol (based on a 2-way hand-shake)

1.1 Relationship To Existing Remote Operation Services

The overall model of ESRO is similar to and consistent with many existing protocols. ESRO's distinguishing characteristic is efficiency.

A brief comparison of ESRO and Remote Procedure Calls [7] and Remote Operation Service Elements [1] follows.

1.1.1 ESRO and RPC

Remote Procedure Call (RPC) is specified in [7] (RFC-1831) and [6] (RFC-1833).

RPC specifications define a remote procedure model that is essentially same as ESRO. RPC's notation uses a syntax quite different from that of ESRO. RPC can rely on a connection oriented or connectionless transport mechanism. When using the connectionless mechanism, the retransmission and reliability issues are considered beyond the scope of the RPC specification. RPC is usually used in combination with External Data Representation, XDR [8] (RFC-1832).

1.1.2 ESRO and ROSE

ROSE is specified in [1] and [2]. The service definition for ESRO Service (ESROS) specified in this document is similar ROSE's Notation. The Notation specified in this document for ESROS is similar ROSE's Notation. The ESRO protocol specified in this document is very different from the ROSE protocol [2].

The operation model for ESRO Service (ESROS) is based on Remote Operations Services Element (ROSE) in [1]. In ESROS model both entities can invoke operations.

ESRO protocols can accomplish short operations with much less overhead than ROSE.

1.2 Overview of ESROS

ESROS provides a service which supports interaction of applications based on a remote operation model. A Remote Operation is invoked by one entity; the other entity attempts to perform the Remote Operation and then reports the outcome of the attempt. The ESROS protocol is designed such that it could support many applications.

1.3 The Remote Operation Model

ESROS provides for performance of operations between two peer sublayers. Users of the ESROS assume the roles of invoker and performer which invoke and perform the operations respectively. An ESROS-User can assume both roles and be an invoker for some operations and be a performer for other operations. The performer is expected to report either the result of the operation or an error. A result reply is sent to the invoker if the operation is successful, and an error reply is sent if the operation is unsuccessful. If the performer is unreachable, the ESROS sends a failure indication primitive to the invoker.

Operations are asynchronous and the invoker may continue to invoke further operations without waiting for a reply. Synchronous or serialized operations are also supported as a subset and a special case of asynchronous service. By default the ESRO service provider on both invoker and performer sides supports the asynchronous operation invocation. However, if one side is to support only serialized (synchronous) mode, it should be in agreement with the peer side.

ESROS has no authentication mechanism. Authentication is the responsibility of the performer (which is outside of the scope of ESROS) and the performer is not expected to honor the invoker when it is not authenticated.

The ESROS operation model is represented in Figure 1. In this example, the ESROS User on the left is the Invoker and the ESROS User on the right is the Performer. The Provider is the entity providing a service to the layer above it.

2 ESRO SERVICE DEFINITIONS

ESRO service primitives are illustrated in Figure 2, Table 1 and Table 2. The description of services and primitives comes in the following sections.

ESROS-User accesses ESRO services through Efficient Short Remote Operations Service Access Point (ESRO-SAP) as shown in Figure 2.

The RESULT.request, ERROR.request and FAILURE.indication service primitives can be implemented in two different modes:

1. Acknowledged Result, and
2. Non-Acknowledged Result

```
ESROS      -------------------    -------------------     ESROS
User      | Layer above ESROS |  | Layer above ESROS |    User
(Invoker) |                   |  |                   |  (Performer)
           -------------------    -------------------
   ^               |                      |                ^
   |               |                      |                |
   v               |                      |                v
ESROS     -------------------    -------------------      ESROS
Provider |       ESROS       |  |       ESROS       |    Provider
          -------------------    -------------------
                   |                      |
                   |                      |
                   |                      |
          -------------------    -------------------
         |    UDP            |  |   UDP             |
          -------------------    -------------------
                 \_                     _/
                   \_                 _/
                     \_     .       _/
                       \_ . .* . _/
                         . * .* .
                           * . *
```

Figure 1: ES Remote Operation Model

ESRO Service	Type
ESROS-INVOKE	Non-confirmed
ESROS-INVOKE-P	Provider-initiated
ESROS-RESULT	Confirmed / Non-confirmed
ESROS-ERROR	Confirmed / Non-confirmed
ESROS-FAILURE	Provider initiated

Table 1: ESRO Services

```
                Invoker        Performer
               ESRO SAP         ESRO SAP
                     |               |
                     |               |
ESROS-INVOKE.req.    |               | ESROS-INVOKE.ind.
--------->-----------|               |-------->---------
                     |               |
ESROS-INVOKE-P.conf.|                |
---------<-----------|               |
                     |               |
                     |               |
                     |               |
ESROS-RESULT.ind.    |               | ESROS-RESULT.req.
---------<-----------|               |--------<---------
                     |               |
                     |               | ESROS-RESULT.conf.
                     |               |-------->---------
                     |               |
                     |               |
ESROS-ERROR.ind.     |               | ESROS-ERROR.req.
---------<-----------|               |--------<---------
                     |               |
                     |               | ESROS-ERROR.conf.
                     |               |-------->---------
                     |               |
                     |               |
                     |               |
                     |               |
ESROS-FAILURE.ind.   |               | ESROS-FAILURE.ind.
---------<-----------|               |-------->---------
                     |               |
```

Figure 2: Time sequence diagram for ESRO services

Primitive	Parameters
ESROS-INVOKE.request	Operation-value Performer-address Invoke-argument-encoding-type Invoke-argument
ESROS-INVOKE.indication	Operation-value Invoker-address Invoke-argument-encoding-type Invoke-argument Invoke-ID
ESROS-INVOKE-P.confirm	Invoke-ID
ESROS-RESULT.request	Result-argument-encoding-type Result-argument Invoke-ID
ESROS-RESULT.indication	Result-argument-encoding-type Result-argument Invoke-ID
ESROS-RESULT.confirm	Invoke-ID
ESROS-ERROR.request	Error-value Error-argument-encoding-type Error-argument
ESROS-ERROR.indication	Error-value Error-argument-encoding-type Error-argument Invoke-ID
ESROS-ERROR.confirm	Invoke-ID
ESROS-FAILURE.indication	Failure-value Invoke-ID

Table 2: ESRO service primitives and associated parameters

Service Mode	Performer	Invoker
Acknowledged Result	RESULT.confirm	RESULT.indication
	FAILURE.indication (protocol)	RESULT.indication
	FAILURE.indication (protocol)	FAILURE.indication (protocol)

Table 3: Success and Failure in Acknowledged Result Mode

as described below. The difference between different modes is in their reliability of service and efficiency. Reliability of service is defined based on the understanding of invoker and performer about the success or failure of the operation on the peer side. Table 3 and Table 4 summarize understanding of performer about success or failure on invoker side in different situations. In these tables the FAILURE.indication refers to the primitive generated by protocol and not the failure of local provider.

2.1 Acknowledged Result Service Mode

In this service mode, the result is acknowledged by invoker, but the mechanism by which the acknowledgment is accomplished may not be reliable. Table 3 summarizes the relationship between performer and invoker in success and failure cases.

2.1.1 Performer side

In this type of service, the RESULT.confirm and ERROR.confirm primitives on performer side are generated if the result/error is acknowledged by invoker.

The FAILURE.indication on performer side is generated if result/error is not acknowledged by invoker or if there is a local failure on performer side.

From the protocol point of view, the FAILURE.indication might be because either the result/error PDU or the ack PDU is lost. The outcome of this is that a FAILURE.indication is not robust as the operation may have been successful from the invoker's perspective. One method of compensating for this shortcoming is having the performer verify the FAILURE.indication in a separate operation.

2.1.2 Invoker side

When invoker receives failure indication, the performer has the failure indication too.

This type of service can be implemented by protocols based on 3-Way handshaking.

Service Mode	Performer	Invoker
Non-acknowledged Result	RESULT.confirm	RESULT.indication
	RESULT.confirm	FAILURE.indication (protocol)
	FAILURE.indication (protocol) does not exist	---

Table 4: Success and Failure in Non-acknowledged Result Mode

2.2 Non-acknowledged Result

In this service mode the result is not acknowledged. Table 4 summarizes the relationship between performer and invoker in success and failure cases.

2.2.1 Performer side

In this type of service, the RESULT.confirm and ERROR.confirm primitives on performer side are generated without receiving additional information from the invoker peer. In other words, these Primitives have no protocol-related meaning and convey no information, other than end-of-operation.

The FAILURE.indication on performer side is not generated by protocol. The only case that can generate FAILURE.indication on performer side is local failure in service provider on performer side.

2.2.2 Invoker side

The FAILURE.indication on invoker side can be the resultof not receiving result/error/failure from peer performer or it can result from failure in local service provider.

This type of service can be implemented by protocols based on 2-Way handshaking.

2.3 Serialized Use of ESRO Services

Although the ESRO Services are defined to support asynchronous operation invocation in general, they can be used in the special case of synchronous (serialized) mode too. The serialized use of ESRO Services is implementation specific. However, one of the possible scenarios is as follows:

```
               Invoker        Performer
              ESROS AP         ESROS AP
                    |               |
                    |               |
ESROS-INVOKE.req.   |               | ESROS-INVOKE.ind.
-------->-----------|               |-------->---------
                    |               |
ESROS-INVOKE-P.conf.|               |
--------<-----------|               |
                    |               |
ESROS-FAILURE.ind.  |               |
--------<-----------|               |
                    |               |
```

Figure 3: Time sequence diagram for ESROS-INVOKE service

2.3.1 Invoker

Invokes an operation after it receives either RESULT.indication, ERROR.indication, or FAILURE.indication for the previous operation.

2.3.2 Performer

Considers an operation to be complete and accepts the next operation after it receives RESULT.confirm, ERROR.confirm, or FAILURE.indication.

2.4 ESROS-INVOKE Service

The ESROS-INVOKE service is used by an ESROS-User (the invoker) to cause the invocation of an OPERATION to be performed by the other ESROS-User (the performer).

ESROS Invoker User issues ESROS-INVOKE.request primitive to invoke an operation.

ESROS-INVOKE.indication primitive provides the ESROS Performer User with the parameters of the invoked operation.

ESRO Service Provider issues the ESROS-INVOKE-P.confirm primitive to provide the ESROS Invoker User with Invoke-ID of the invoked operation.

The related service structure consists of three service primitives as illustrated in Figure 3 and Table 5.

Primitive	Parameters
ESROS-INVOKE.request	Operation-value Performer-address Invoke-argument-encoding-type Invoke-argument
ESROS-INVOKE.indication	Operation-value Invoker-address Invoke-argument-encoding-type Invoke-argument Invoke-ID
ESROS-INVOKE-P.confirm	Invoke-ID Failure-value
ESROS-FAILURE.indication	Invoke-ID

Table 5: ESROS-INVOKE service primitives and associated parameters

2.4.1 Operation-value

This value is the identifier of the operation to be invoked. The value is agreed upon between the ESROS Users. This parameter has to be supplied by the invoker of the service.

ESROS Invoker User provides the Operation-value parameter for the ESROS-INVOKE.request primitive. The Operation-value parameter of ESROS-INVOKE.indication is provided to the ESROS Performer User.

2.4.2 Performer-address

This parameter is the address of the ESROS Performer User which consists of ESRO Service Access Point (SAP) Selector, Transport Service Access Point (TSAP) Selector (e.g., port number), and Network Service Access Point (NSAP) address (e.g., IP address). This parameter has to be supplied by the invoker of the service.

ESROS Invoker User provides the Performer-address parameter for the ESROS-INVOKE.request primitive.

2.4.3 Invoker-address

This parameter is the address of the ESROS Invoker User which consists of ESRO Service Access Point (SAP) Selector, Transport Service Access Point (TSAP) Selector (e.g. port number), and Network Service Access Point (NSAP) address (e.g. IP address).

The Invoker-address parameter of ESROS-INVOKE.indication is provided to the ESROS Performer User.

2.4.4 Invoke-argument-encoding-type

This parameter identifies the encoding type of the Invoke-argument (see next subsection). The encoding type has to be agreed upon between ESROS Users. This parameter has to be supplied by the invoker of the service.

ESROS Invoker User provides the Invoke-argument-encoding-type parameter for the ESROS-INVOKE.request primitive. The Invoke-argument-encoding-type parameter of ESROS-INVOKE.indication is provided to the ESROS Performer User.

2.4.5 Invoke-argument

This parameter is the argument of the invoked operation. The type has to be agreed between the ESROS Users. This parameter has to be supplied by the invoker of the service. Encoding type of the Invoke-argument is specified through the Invoke-argument-encoding-type parameter (see previous subsection).

ESROS Invoker User provides the Invoke-argument parameter for the ESROS-INVOKE.request primitive. The Invoke-argument parameter of ESROS-INVOKE.indication is provided to the ESROS Performer User.

2.4.6 Invoke-ID

This parameter identifies the invocation of an ESROS-INVOKE service and is used to correlate this invocation with the corresponding replies (ESROS-RESULT, ESROS-ERROR, and ESROS-FAILURE services.) This parameter has to be supplied by the ESROS provider.

This parameter distinguishes several invocations of the service in progress (asynchronous operations). The ESROS provider may begin to reuse Invoke-ID values whenever it chooses, subject to the constraint that it may not reuse an Invoke-ID value that was previously assigned to an invocation of the service for which it expects, but has not yet received a reply. In other words, the provider does not reuse a previously used Invoke-ID unless the corresponding service is fully completed.

2.4.7 Failure-value

This parameter identifies the failure that occurred during the processing or transmission of any of the service primitives of ESROS. This parameter has to be supplied by the ESROS provider (see also Section 2.7).

2.5 ESROS-RESULT Service

The ESROS-RESULT service is used by an ESROS User to reply to a previous ESROS-INVOKE.indication in the case of a successfully performed operation. This service is either confirmed or non-confirmed based on the service mode (see Section 2).

```
                  Invoker        Performer
                 ESROS AP         ESROS AP
                       |              |
                       |              |
ESROS-RESULT.ind.      |              | ESROS-RESULT.req.
--------<-----------|              |--------<---------
                       |              |
                       |              | ESROS-RESULT.conf.
                       |              |-------->---------
                       |              |
                       |              | ESROS-FAILURE.ind.
                       |              |-------->---------
                       |              |
```

Figure 4: Time sequence diagram for ESROS-RESULT service

The related service structure consists of three service primitives as illustrated in Figure 4 and Table 6.

2.5.1 Result-argument-encoding-type

This parameter identifies the encoding type of the Result-argument (see next subsection). The encoding type has to be agreed upon between the ESROS Users. This parameter has to be supplied by the ESROS Performer User.

ESROS Performer User provides the Result-argument-encoding-type parameter for the ESROS-RESULT.request primitive. The Result-argument-encoding-type parameter of ESROS-RESULT.indication is provided to the ESROS Invoker User.

2.5.2 Result-argument

This parameter is the result of an invoked and successfully performed operation. The type has to be agreed between the ESROS Users. This parameter has to be supplied by the invoker of the service. Encoding type of the Result-argument is specified through the Result-argument-encoding-type parameter (see previous subsection).

ESROS Performer User provides the Result-argument parameter for the ESROS-RESULT.request primitive. The Result-argument parameter of ESROS-RESULT.indication is provided to the ESROS Invoker User.

2.5.3 Invoke-ID

This parameter identifies the corresponding invocation. This Invoke-ID, which is originally generated by the ESROS provider at the time of ESROS-INVOKE indication, is extracted from the Invoke ID that has to be supplied by the ESROS performer User. The value is that of the corresponding ESROS-INVOKE.indication primitive.

Primitive	Parameters
ESROS-RESULT.request	Result-argument-encoding-type Result-argument Invoke-ID
ESROS-RESULT.indication	Result-argument-encoding-type Result-argument Invoke-ID
ESROS-RESULT.confirm	Invoke-ID Failure-value
ESROS-FAILURE.indication	Invoke-ID

Table 6: ESROS-RESULT service primitives and associated parameters

2.5.4 Failure-value

This parameter identifies the failure that occurred during the processing or transmission of any of the service primitives of ESROS. This parameter has to be supplied by the ESROS provider (see also Section 2.7).

2.6 ESROS-ERROR Service

The ESROS-ERROR service is used by an ESROS User to reply to a previous ESROS-INVOKE.indication in the case of an unsuccessfully performed operation. This service is either confirmed or non-confirmed based on the service mode (see Section 2).

The related service structure consists of three service primitives as illustrated in Figure 5 and Table 7.

2.6.1 Error-value

This parameter identifies the error in reply to a previous ESROS-INVOKE.indication in the case of an unsuccessfully performed operation. The value has to be agreed between the ESROS-Users. This parameter has to be supplied by the ESROS Performer User.

ESROS Performer User provides the Error-argument parameter for the ESROS-ERROR.request primitive. The Error-argument parameter of ESROS-ERROR.indication is provided to the ESROS Invoker User.

```
               Invoker        Performer
              ESROS AP         ESROS AP
                    |               |
                    |               |
ESROS-ERROR.ind.    |               | ESROS-ERROR.req.
--------<-----------|               |--------<---------
                    |               |
                    |               | ESROS-ERROR.conf.
                    |               |-------->---------
                    |               |
                    |               | ESROS-FAILURE.ind.
                    |               |-------->---------
```

Figure 5: Time sequence diagram for ESROS-ERROR service

Primitive	Parameters
ESROS-ERROR.request	Error-value Error-argument-encoding-type Error-argument
ESROS-ERROR.indication	Error-value Error-argument-encoding-type Error-argument Invoke-ID
ESROS-ERROR.confirm	Invoke-ID Failure-value
ESROS-FAILURE.indication	Invoke-ID

Table 7: ESROS-ERROR service primitives and associated parameters

2.6.2 Error-argument-encoding-type

This parameter identifies the encoding type of the Error-argument (see next subsection). The encoding type has to be agreed upon between the ESROS Users. This parameter has to be supplied by the ESROS Performer User.

ESROS Performer User provides the Error-argument-encoding-type parameter for the ESROS-ERROR.request primitive. The Error-argument-encoding-type parameter of ESROS-ERROR.indication is provided to the ESROS Invoker User.

2.6.3 Error-argument

This parameter provides additional information about the error in reply to a previous ESROS-INVOKE.indication in the case of an unsuccessfully performed operation. The type (if any) has to be agreed between the ESROS users. This parameter has to be supplied by the ESROS Performer User. Encoding type of the Error-argument is specified through the Error-argument-encoding-type parameter (see previous subsection).

ESROS Performer User provides the Error-argument parameter for the ESROS-ERROR.request primitive. The Error-argument parameter of ESROS-ERROR.indication is provided to the ESROS Invoker User.

2.6.4 Invoke-ID

This parameter identifies the corresponding invocation. This Invoke-ID, which is originally generated by the ESROS provider at the time of the ESROS-INVOKE.indication, is extracted from the Invoke ID which has to be supplied by the ESROS performer User. The value is that of the corresponding ESROS-INVOKE.indication primitive.

2.6.5 Failure-value

This parameter identifies the failure that occurred during the processing or transmission of any of the service primitives of ESROS. This parameter has to be supplied by the ESROS provider (see also Section 2.7).

2.7 ESROS-FAILURE Service

The ESROS-FAILURE service is used by ESROS provider to indicate the failure in providing an ESROS-INVOKE, ESROS-RESULT, or ESROS-ERROR service.

The related service structure consists of one service primitive as illustrated in Figure 6 and Table 8.

```
              Invoker        Performer
             ESROS AP         ESROS AP
                   |              |
                   |              |
ESROS-FAILURE.ind. |              |
--------<----------|              |
                   |              |
                   |              | ESROS-FAILURE.ind.
                   |              |--------->---------
                   |              |
```

Figure 6: Time sequence diagram for ESROS-FAILURE service

Primitive	Parameters
ESROS-FAILURE.indication	Failure-value Invoke-ID

Table 8: ESROS-FAILURE service primitives and associated parameters

2.7.1 Failure-value

This parameter identifies the failure that occurred during the processing or transmission of any of the service primitives of ESROS. This parameter has to be supplied by the ESROS provider.

The values for encoding of Failure-value are presented in Table 9.

2.7.2 Invoke-ID

This parameter identifies the corresponding invocation. This Invoke-ID, which is originally generated by ESROS provider at the time of the ESROS-INVOKE.indication, is extracted from the Invoke ID which has to be supplied by ESROS performer User. The value is that of the corresponding ESROS-INVOKE.indication primitive.

3 ESRO SERVICE NOTATION

Users of ESRO services (invoker and performer) need to agree on a well defined set of parameters which are enumerated below.

1. The operation's Argument data type.

Failure Value	Meaning
0	Transmission failure
1	Out of local resources
2	User not responding
3	Out of remote resources
4	Reassembly failure

Table 9: Encoding of Failure-value

2. The operation's Result data type.
3. The operation's Error data type.
4. The operation's value. A specific tag which uniquely identifies the operation.

The invoker and the performer can specify these parameters using a variety of mechanisms. The notation specified in this section is one such mechanism. It is not the only machanism and ESRO protocol can be used independent of this notation.

3.1 ES-OPERATION Notation

The Remote Operations and Operation Errors are specified in this section. The notation is defined by means of the macro facility defined in [3].

The macros enabling the specification of operations and errors are listed in Figure 7.

Note that this notation is very similar to the abstract operation defined in [1]. The value form of ES-OPERATION is always an integer.

3.2 Mapping of ESROS Notation

3.2.1 Invocation of an Operation

An operation is mapped onto the ESRO Services.

The invocation of an operation is mapped on the ESRO-INVOKE service.

The value assigned to the operation is mapped on the Operation-value parameter of that service. The value of the Named-Type in the ARGUMENT clause of the OPERATION Macro is mapped on the Argument parameter of that service.

```
DEFINITIONS ::=
BEGIN
  ES-OPERATION, ERROR;
  -- macro definition for operations
  ES-OPERATION MACRO ::=
  BEGIN
        TYPE NOTATION   ::=     Argument Result Errors
        VALUE NOTATION  ::=     value (localValue INTEGER)
        Argument        ::=     "ARGUMENT" NamedType | empty
        Result          ::=     "RESULT" ResultType | empty
        ResultType      ::=     NamedType | empty
        Errors          ::=     "ERRORS" "{"ErrorNames"}" | empty
        ErrorNames      ::=     ErrorList | empty
        ErrorList       ::=     Error | ErrorList "," Error
        Error           ::=     value (ERROR) | type
        NamedType       ::=     identifier type | type
  END
  -- macro definition for operations errors
  ERROR MACRO   ::=
  BEGIN
        TYPE NOTATION   ::=     Parameter
        VALUE NOTATION  ::=     value (localValue INTEGER)
        Parameter       ::=     "PARAMETER" NamedType | empty
        NamedType       ::=     identifier type | type
  END
END
```

Figure 7: ES Remote Operation Notation

3.2.2 Reply of an Operation

If an operation was successfully performed, the reply is mapped on the ESRO-RESULT service.

The value of the Named-Type in the RESULT clause of the OPERATION macro is mapped on the Result parameter of that service.

If an operation was not successfully performed, the reply is mapped on the ESRO-ERROR service.

In this case one of the errors in the Identifier List of Error Names in the ERROR clause of the OPERATION macro may be applied. The value assigned to the applied error is mapped onto the Error parameter of that service. The value of the Named-Type in the PARAMETER clause of the ERROR macro of the applied error is mapped on the Error- parameter of that service.

4 REMOTE OPERATIONS PROTOCOL

4.1 Overview of the Protocol

The ESROS protocol realizes the services defined in the section entitled ESROS Service Definitions. Short operations are performed in a highly efficient manner. The protocol operation is summarized below and is described in detail in the following sections.

Two Functional Units are defined which realize the services with 2-Way handshake and 3-Way handshake, called 2-Way Handshake Functional Unit and 3-Way Handshake Functional Unit respectively.

The procedures specified in this section refer to Protocol Data Units (PDUs) which are defined in Section 4.4.

4.1.1 Service Provision (Invoker User)

- An ESROS user binds to an ESRO Service Access Point (SAP) and specifies whether 3-Way or 2-Way handshake Functional Unit is to be associated with the SAP.
- An ESROS user initiates the transfer of a PDU using the INVOKE service.
- On receipt of an ESROS-INVOKE.request service primitive from the ESROS user:
  - The ESROS provider generates an Invoke ID,
  - Communicates the Invoke-ID to the invoker of the service through the ESROS-INVOKE-P.confirm primitive,

4.1.2 Service Provision (Performer User)

- An ESROS user binds to an ESRO Service Access Point (SAP) and specifies whether 3-Way or 2-Way handshake Functional Unit is to be associated with the SAP.

- On receipt of an ESRO-INVOKE-PDU, the ESROS provider issues an ESROS-INVOKE.indication to the ESROS performer user.
- On receipt of ESROS-RESULT.request or ESROS-ERROR.request from the performer, the provider creates the ESRO-RESULT-PDU or ESRO-ERROR-PDU.
- In the case that the provider receives an ESRO-ACK-PDU for the transmitted ESRO-RESULT-PDU or ESRO-ERROR-PDU, if the corresponding SAP is associated with the 3-Way Handshake Functional Unit, it passes an ESROS-RESULT.confirm or ESROS-ERROR.confirm to the performer user. If the corresponding SAP is associated with the 2-Way handshake Functional Unit, the ESRO-ACK-PDU is dropped as an invalid PDU.
- In the case that the provider is not able to deliver the ESRO-RESULT-PDU or ESRO-ERROR-PDU, it issues an ESROS-FAILURE.indication to the performer user. In the case that the performer's SAP is associated with the 3-Way handshake Functional Unit and provider doesn't receive the ESRO-ACK-PDU for a transmitted ESRO-RESULT-PDU or an ESRO-ERROR-PDU, it passes an ESROS- FAILURE.indication to the performer user.
- In the case that the performer's SAP is associated with the 3-Way handshake Functional Unit and provider receives an ESRO-ACK-PDU for the operation, it passes an ESROS-RESULT.confirm or ESROS-ERROR.confirm. In the case that the performer's SAP is associated with a 2-Way handshake Functional Unit and provider doesn't receive duplicate ESROS-INVOKE-PDUs from the invoker, it passes an ESROS-RESULT.confirm or ESROS-ERROR.confirm.
- On receipt of an ESRO-FAILURE-PDU, the ESROS provider issues an ESROS-FAILURE.indication to the ESROS performer user.

4.2 Protocol Procedures

4.2.1 Service Access Point (SAP) Bind Procedure

To access the ESRO Services, an ESROS user binds to an ESRO Service Access Point and specifies the SAP to be associated with 3-Way handshake Functional Unit or 2-Way handshake Functional Unit. ESROS provider generates a SAP descriptor which is passed to the user. The handshaking for all Invoke.requests addressed to that SAP and all PDUs addressed to that SAP will be either 3-Way or 2-Way based on the Functional Unit associated with SAP and specified by user at SAP bind time.

It is the responsibility of the ESROS peer users (invoker and performer) to address their operations to the appropriate SAP (3-Way or 2-Way) based on the agreement between users.

4.2.2 Invoke Service Procedure

An ESROS user initiates the transfer of a PDU using the INVOKE service.

On receipt of an ESRO-INVOKE-PDU, the ESROS provider sends an ESROS-INVOKE.indication primitive to the ESROS performer user.

4.2.3 Invoke ID Assignment Procedure

On receipt of an ESROS-INVOKE.request primitive from the ESROS user, the ESROS provider generates two invoke identifiers:

- Invoke-Reference-Number: Uniquely identifies the invocation between the two peers. This is a PDU field with a length of 8 bits (see section 4.4).
- Invoke-ID-Parameter: Uniquely identifies the invocation to the service user. This Invoke-ID-Parameter is a combination of the Invoke-Reference-Number described above and the invoker address, performer address, and the SAP Selector.

The provider communicates the Invoke-ID-Parameter to the invoker of the INVOKE service through the ESROS-INVOKE-P.confirm primitive.

The Invoke-Reference-Number distinguishes several invocations of the service in progress (asynchronous operations). It is also used as segment identifier when a Service Data Unit (SDU) is transferred using segmentation and reassembly. The ESROS provider may begin to reuse the Invoke-Reference-Number values whenever it chooses, subject to the constraint that it may not reuse an Invoke-Reference-Number value that was previously assigned to an invocation of the service for which it expects, but has not yet received, a reply. In other words the provider does not reuse a previously used Invoke-Reference-Number unless the corresponding service is fully completed. The same value of the Invoke-Reference-Number can be reused to identify the invocation between different peer entities. In that case, the combination of the peer entity's address and the Invoke-Reference-Number guarantees unique identification of each invocation.

4.2.4 Functional Unit Selection Procedure

When an ESRO Services user binds to an ESRO SAP, it associates its SAP descriptor to 3-Way Handshake Functional Unit or 2-Way Handshake Functional Unit.

Based on the Functional Unit associated with SAP, provider selects the corresponding Functional Unit for all Invoke Requests or PDUs addressed to that SAP.

4.3 Connectionless PDU Transfer For Small PDUs

4.3.1 Overview

PDUs sent by UDP use port ESRO_CL_PORT. PDUs carried by UDP are restricted to CLRO_SMALL_PDU_MAX_SIZE bytes (see 4.6.1)

Each PDU is encapsulated in a single UDP datagram.

For PDUs larger than CLRO_SMALL_PDU_MAX_SIZE but smaller than CLRO_SEGMENTED_PDU_MAX_SIZE bytes (see 4.6.1), segmentation and reassembly is used and each segment is transmitted in a UDP datagram.

From Idle to:	Event
CL-Invoker Transition Diagram 2-way Handshake (Connectionless)	ESRO-INVOKE.req
CL-Invoker Transition Diagram 3-way Handshake (Connectionless)	ESRO-INVOKE.req
CL-Performer Transition Diagram 3-way Handshake (Connectionless)	INVOKE-PDU
CL-Performer Transition Diagram 2-way Handshake (Connectionless)	INVOKE-PDU

Table 10: ESROS Finite State Machine

PDUs sent using UDP may be lost, and hence a retransmission strategy is defined. When a PDU is segmented, the retransmission strategy is not applied to individual segments (i.e., loss of one segment results in retransmission of the whole SDU).

The optimal UDP retransmission policy will vary with the performance of the network and the needs of the transmitter, but the following are considered:

The retransmission interval should be based on prior statistics if possible. Too aggressive retransmission can easily slow response time of the network at large. Depending on how well connected the invoker is to its performer, the minimum retransmission interval should be RETRANSMISSION_INTERVAL (see 4.6.2) seconds.

Delivery of PDUs is asynchronous which means the ESROS does not wait for the result of a transmitted PDU and continues delivering the next PDUs.

This section describes the ESROS protocols in terms of state diagrams. The ESROS Finite State Machine is expressed as four separate transition diagrams. This is illustrated in Table 10.

Details of each of the two transition diagrams for connectionless transmission and different handshakings are described in the following sections. The state diagrams show the state, the events, the actions taken and the resultant state.The ESROS state transition diagrams for connectionless data transmission are presented in Table 11, Table 12, Table 13, and Table 14.

Transitions are identified by numbers on the state diagrams. The corresponding actions are listed next to each table.

4.3.2 3-Way Handshake Functional Unit

This unit implements the Acknowledged Result model of ESRO Services. 3-Way handshaking is used in this unit.

State / Event	STA01 CL Invoker Start	STA02 Invoke PDU Send	STA03 ACK-PDU Send	STA04 Invoker RefNu Wait
U: INVOKE.request	(1) STA02			
T: INVOKE PDU Retransmit		(2) STA02		
T: Last Timer		(3) STA04		
P: Result-PDU				(9) STA04
P: Failure-PDU		(5) STA04		
P: ACK-PDU (Hold On)		(6) STA02		
P: Duplicate Result-PDU			(7) STA03	
T: RefNu Timer				(8) STA01
P: Result-PDU		(4) STA03		
T: Inactivity Timer			(10) STA04	

Table 11: ESROS State Transition Diagram-Connectionless Transmission, 3-Way HS. P = Protocol, T = Timer, U = User, I = Internal.

The RESULT.confirm and ERROR.confirm primitives on performer are generated when ESRO-ACK-PDU is received.

The FAILURE.indication on performer side is resulted from remote or local failures. Not receiving ESRO-ACK-PDU or local failure can generate FAILURE.indication primitive.

The FAILURE.indication on invoker side is generated if a local failure happens or a ESRO-FAILURE-PDU is received.

The transmission of INVOKE, RESULT, and ERROR SDUs can be in a single PDU (when it fits in one UDP) or a sequence of segment PDUs.

3-Way Handshake Connectionless Transmission: Invoker

For each transition number in the state diagram Table 11, the corresponding actions are listed below:

1. INVOKE.request:
   - Assign Invoke-ID.
   - Issue ESROS-INVOKE-P.confirm primitive.
   - Assign invoke reference number.
   - Send operation in one ESRO-INVOKE-PDU or in segmented INVOKE- PDUs depending on the size of the operation.
   - Initialize retransmission counter.
   - Initialize retransmission timer.
2. Invoke PDU Retransmit:
   - Retransmit operation in one ESRO-INVOKE-PDU or segmented PDUs while number of retransmissions is less than MAX_RETRANSMISSIONS.
   - Increment the retransmission counter. When MAX_RETRANSMISSIONS reached, start LAST_TIMER, otherwise initialize retransmission timer.
3. Last Timer:
   - Issue ESROS-FAILURE.indication primitive.
   - Initialize reference number timer.
4. ESRO-RESULT-PDU or ESRO-ERROR-PDU (or reassembled ESRO- RESULT-SEGMENTED-PDU or ESRO-ERROR-SEGMENTED-PDU when the PDU is received in segmented format):
   - Send ESRO-ACK-PDU.
   - Issue ESROS-RESULT.indication or ESROS-ERROR.indication primitive.
   - Initialize inactivity timer.
5. ESRO-FAILURE-PDU:
   - Issue ESROS-FAILURE.indication primitive with User not Responding failure cause.
   - Initialize reference number timer.
6. ESRO-ACK-PDU (Hold on):
   - For future use (no action).
7. Duplicate ESRO-RESULT-PDU or ESRO-ERROR-PDU:
   - Initialize inactivity timer (Ignore PDU).
   - Send ESRO-ACK-PDU.
8. Invoke reference number timer:
   - Release the invoke reference number.
9. ESRO-RESULT-PDU or ESRO-ERROR-PDU:

- Reset Invoke reference number timer.

10. Inactivity timer:
    - Initialize reference number timer.

On receipt of an ESROS-INVOKE.request, ESROS provider generates an Invoke- Reference-Number and an Invoke-ID (see Section 4.2.3). The provider issues an ESROS-INVOKE-P.confirm primitive and passes the Invoke-ID to the invoker.

The ESROS provider initiates the timer for the Invoke-ID and transmits the PDU. Based on the size of SDU, if segmentation is required, the SDU is segmented and transmitted in a sequence of segmented PDUs. If the ESRO-RESULT-PDU or ESRO-ERROR-PDU associated with the invoke ID is not received within the INVOKE_PDU_RETRANSMISSION_INTERVAL (see 4.6.2) period, the SDU is retransmitted (in one PDU or segmented and transmitted in a sequence of segment PDUs). The retransmission is repeated for a maximum of MAX_RETRANSMISSIONS unless an ESRO-RESULT-PDU or ESRO-ERROR-PDU is received.

If the ESRO-RESULT-PDU or ESRO-ERROR-PDU is received in a segmented format, the reassembly process reassembles the sequence of segment PDUs.

In the case that the Hold-on ESRO-ACK-PDU is received from the performer, the provider stops retransmitting the ESRO-INVOKE-PDU and waits for the ESRO- RESULT-PDU or ESRO-ERROR-PDU for a period equal to the multiplication of INVOKE_PDU_RETRANSMISSION_INTERVAL (see 4.6.2) and MAX_RETRANSMISSIONS (see 4.6.2, for future use).

In the case that the ESRO-INVOKE-PDU is sent MAX_RETRANSMISSIONS (see 4.6.2) times and no ESRO-RESULT-PDU or ESRO-ERROR-PDU is received, the ESROS provider sends an ESROS-FAILURE.indication primitive, with the Invoke-ID of the failed PDU and the Failure-value as parameters, to the invoker.

When an ESRO-RESULT-PDU or ESRO-ERROR-PDU is received (whether in one PDU or reassembled from a sequence of segmented PDUs), the provider issues an ESROS-RESULT.indication or ESROS-ERROR.indication to the invoker user, sends an ESRO-ACK-PDU and initializes the inactivity timer. In the case that duplicate ESRO- RESULT-PDU or ESRO-ERROR-PDU are received, they are ignored, the inactivity timer is reset, and an ESRO-ACK-PDU is retransmitted.

When no duplicate ESRO-RESULT-PDU or ESRO-ERROR-PDU is received for a period equal to INACTIVITY_TIME (see 4.6.2), or in the case of ESRO-INVOKE- PDU retransmission time-out, or in the case of internal failure, the provider initializes the reference number timer. After REFERENCE_NUMBER_TIME (see 4.6.2), the reference number is released.

3-Way Handshake Connectionless Transmission: Performer

For each transition number in the state diagram above, the corresponding actions are listed below:

1. ESRO-INVOKE-PDU (as a single PDU or a sequence of segment PDUs):
    - Issue ESROS-INVOKE.indication primitive.
2. ESROS-RESULT.request or ESROS-ERROR.request:

State / Event	STA01 CL Performer Start	STA02 Invoke PDU Received	STA03 ACK-PDU Wait	STA04 Performer RefNu Wait
P: Invoke-PDU	(1) STA02			
U: RESULT.req.		(2) STA03		
P: ACK-PDU			(3) STA04	
P: Invoke-PDU Duplicate		(4) STA02	(6) STA03	(7) STA04
T: Result-PDU Retransmission Timer			(5) STA03	
I: Failure		(8) STA01		
T: Last Time			(9) STA04	
T: RefNu Timer				(10) STA01
P: ACK-PDU Duplicate				(11) STA04
U/P: Hold On ACK		(12) STA02		

Table 12: ESROS State Transition Diagram-Connectionless Transmission, 3-Way HS: Performer. P = Protocol, T = Timer, U = User, I = Internal.

- Add invoke reference number to the active list.
- Transmit ESRO-RESULT-PDU or ESRO-ERROR-PDU (in a single PDU or a sequence of segment PDUs).
- Set ESRO-RESULT-PDU or ESRO-ERROR-PDU retransmission timer.

3. ESRO-ACK-PDU:
   - Initialize invoke reference number timer.
   - Issue ESROS-RESULT.confirm or ESROS-ERROR.confirm.
4. Duplicate ESRO-INVOKE-PDU:
   - No action (ignore the duplicate ESRO-INVOKE-PDU).
5. ESRO-RESULT-PDU or ESRO-ERROR-PDU retransmission timer:
   - Retransmit ESRO-RESULT-PDU or ESRO-ERROR-PDU (in a single PDU or in a segmented format) while number of retransmissions is less than MAX_RETRANSMISSIONS.
   - Increment the transmission counter.
6. Duplicate ESRO-INVOKE-PDU:
   - Retransmit ESRO-RESULT-PDU or ESRO-ERROR-PDU.
   - Reset ESRO-RESULT-PDU or ESRO-ERROR-PDU retransmission timer.
   - Re-initialize the number of retransmissions counter to 1.
7. Duplicate ESRO-INVOKE-PDU:
   - Reset invoke reference number timer.
8. Internal failure:
   - Send ESRO-FAILURE-PDU.
   - Release the invoke reference number.
9. Last time:
   - Issue ESROS-FAILURE.indication.
   - Initialize invoke reference number timer.
10. Invoke reference number timer:
    - Release the invoke reference number.
11. Duplicate ESRO-ACK-PDU:
    - Reset invoke reference number timer.
12. Hold-on ACK request:
    - Send hold-on ESRO-ACK-PDU (for future use).

On receipt of an ESRO-INVOKE-PDU, the ESROS provider issues an ESROS-INVOKE.indication to the ESROS performer user. The provider ignores the duplicate ESRO-INVOKE-PDUs.

In the case of internal failure or no response from performer user, the provider sends an ESRO-FAILURE-PDU and releases the invoke reference number.

On receipt of a Hold-on request from the performer user, or based on other information, provider sends a Hold-on ESRO-ACK-PDU (future use).

On receipt of either ESROS-RESULT.request or ESROS-ERROR.request from the ESROS performer user, the ESROS provider initiates the retransmission timer for the ESRO-RESULT-PDU or ESRO-ERROR-PDU and transmits the ESRO-RESULT-PDU or ESRO-ERROR-PDU in a single PDU or in a sequence of segment PDUs. If the ESRO-ACK-PDU associated with the Invoke-ID is not received within RESULT_ERROR_PDU_RETRANSMISSION_INT (see 4.6.2), the PDU is retransmitted.

When provider is waiting for ESRO-ACK-PDU and a duplicate ESRO-INVOKE-PDU arrives, ESRO-RESULT-PDU or ESRO-ERROR-PDU is retransmitted (in a single PDU or in a sequence of segment PDUs), the retransmission timer is reset and counter for number of retransmissions is re-initialized to 1.

If after MAX_TRANSMISSIONS (see 4.6.2) no ESRO-ACK-PDU is received, the provider issues an ESROS-FAILURE.indication primitive, with the Invoke-ID of the failed PDU and the Failure-value as parameters, to the performer user. Then the provider sets the reference number timer and releases the reference number after REFERENCE_NUMBER_TIME (see 4.6.2).

On receipt of ESRO-ACK-PDU associated with the Invoke-ID before MAX_TRANSMISSIONS (see 4.6.2), the provider issues a ESROS-RESULT.confirm or ESROS-ERROR.confirm primitive and sets the reference number timer and releases the reference number after REFERENCE_NUMBER_TIME (see 4.6.2).

The duplicate ESRO-ACK-PDU and duplicate ESRO-INVOKE-PDUs are ignored while provider waits for the reference number timer to expire.

4.3.3 2-Way Handshake Functional Unit

This Functional Unit implements the Not-Acknowledged Result model of ESRO Services. 2-Way handshaking is used in this unit.

The RESULT.confirm and ERROR.confirm primitives on performer side are generated based on time-out, i.e. when no duplicate ESRO-INVOKE-PDU is received in a specified period of time, provider issues RESULT.confirm or ERROR.confirm primitive.

The FAILURE.indication on performer side is generated as a result of local failure or after time-out of retransmission of ESRO-RESULT-PDU or ESRO-ERROR-PDU.

The FAILURE.indication on invoker side is generated if a local failure happens or a ESRO-FAILURE-PDU is received.

The transmission of INVOKE, RESULT, and ERROR PDUs can be in a single PDU (when it fits in one PDU) or a sequence of segmented PDUs.

State / Event	STA01 2-Way HS CL Invoker Start	STA02 Invoke PDU Send	STA03 Invoker RefNu Wait
U: INVOKE.req.	(1) STA02		
T: Invoke PDU Retransmit		(2) STA02	
T: Last Timer		(3) STA03	
P: Result/Error PDU		(4) STA03	
P: Failure-PDU		(5) STA03	
P: Duplicate Result PDU			(6) STA03
T: RefNu Timer			(7) STA01

Table 13: ESROS State Transition Diagram-Connectionless Transmission, 2-Way HS: Invoker p = Protocol, T = Timer, U = User, I = Internal.

2-Way Handshake Connectionless Transmission: Invoker

For each transition number in the state diagram above, the corresponding actions are listed below:

1. INVOKE.request:
   - Assign Invoke-ID.
   - Issue ESROS-INVOKE-P.confirm primitive.
   - Assign invoke reference number.
   - Send ESRO-INVOKE-PDU in a single PDU or as a sequence of segment PDUs.
   - Initialize retransmission counter.
2. Invoke PDU Retransmit:
   - Retransmit ESRO-INVOKE-PDU (in a single PDU or in a sequence of segment PDUs) while number of retransmissions is less than MAX_RETRANSMISSIONS.
   - Increment the transmission counter. When MAX_RETRANSMISSIONS reached, start LAST_TIMER.
3. Last Timer:
   - Issue ESROS-FAILURE.indication primitive.
   - Initialize reference number timer.
4. ESRO-RESULT-PDU or ESRO-ERROR-PDU:

- Issue ESROS-RESULT.indication or ESROS-ERROR.indication primitive.
- Initialize reference number timer.

5. ESRO-FAILURE-PDU:
   - Issue ESROS-FAILURE.indication primitive with User not Responding failure cause.
   - Initialize reference number timer.
6. Duplicate ESRO-RESULT-PDU or ESRO-ERROR-PDU:
   - Reset Invoke reference number timer.
7. Invoke reference number timer:
   - Release the invoke reference number.

On receipt of an ESROS-INVOKE.request, ESROS provider generates an Invoke- Reference-Number and an Invoke-ID (see 4.2.3). The provider issues an ESROS-INVOKE-P.confirm primitive and passes the Invoke-ID to the invoker.

The ESROS provider initiates the timer for the Invoke-ID and transmits the PDU. The PDU is transmitted as a single PDU or a sequence of segment PDUs. If the ESRO- RESULT-PDU or ESRO-ERROR-PDU associated with the invoke ID is not received within the INVOKE_PDU_RETRANSMISSION_INTERVAL (see 4.6.2) period, the PDU is retransmitted. The retransmission is repeated for a maximum of MAX_RETRANSMISSIONS unless an ESRO-RESULT-PDU or ESRO-ERROR-PDU is received.

In the case that the ESRO-INVOKE-PDU is sent MAX_RETRANSMISSIONS (see 4.6.2) times and no ESRO-RESULT-PDU or ESRO-ERROR-PDU is received, the ESROS provider sends an ESROS-FAILURE.indication primitive, with the Invoke-ID of the failed PDU and the Failure-value as parameters, to the invoker. If ESRO-FAILURE-PDU is received, the ESROS provider sends and ESROS-FAILURE.indication primitive, with the Invoke-Id of the failed PDU and the Failure- value as parameters to the invoker.

When an ESRO-RESULT-PDU or ESRO-ERROR-PDU is received, the provider issues an ESROS-RESULT.indication or ESROS-ERROR.indication to the invoker user, and initializes the Reference-Number timer. In the case that duplicate ESRO-RESULT-PDU or ESRO-ERROR-PDU are received, they are ignored. In the case of internal failure, the provider initializes the reference number timer. After REFERENCE_NUMBER_TIME (see 4.6.2), the reference number is released.

2-Way Handshake Connectionless Transmission: Performer

For each transition number in the state diagram above, the corresponding actions are listed below:

1. ESRO-INVOKE-PDU (received in a single PDU or reassembled from a sequence of segment PDUs):
   - Issue ESROS-INVOKE.indication primitive.
2. Duplicate ESRO-INVOKE-PDU:
   - No action (ignore the duplicate ESRO-INVOKE-PDU).

State / Event	STA01 2-Way HS CL Performer Start	STA02 Invoke PDU Received	STA03 Result PDU Retransmit	STA04 Performer RefNu Wait
P: Invoke-PDU	(1) STA02			
P: Invoke-PDU Duplicate		(2) STA02	(5) STA03	(7) STA04
U: RESULT.req.		(3) STA03		
I: Failure		(4) STA01		
T: Inactivity Timer			(6) STA04	
T: RefNu Timer				(8) STA01

Table 14: ESROS State Transition Diagram-Connectionless Transmission, 2-Way HS: Performer. P = Protocol, T = Timer, U = User, I = Internal.

3. ESROS-RESULT.request or ESROS-ERROR.request:
   - Add invoke reference number to the active list.
   - Transmit ESRO-RESULT-PDU or ESRO-ERROR-PDU (as a single PDU or as a sequence of segment PDUs.)
   - Set Inactivity timer.
4. Internal failure:
   - Send ESRO-FAILURE-PDU.
   - Release the invoke reference number.
5. Duplicate ESRO-INVOKE-PDU:
   - Retransmit ESRO-RESULT-PDU or ESRO-ERROR-PDU (as a single PDU or as a sequence of segment PDUs.)
   - Set Inactivity timer.
6. Inactivity Timer:
   - Issue ESROS-RESULT.confirm.
   - Initialize invoke reference number timer.
7. Duplicate ESRO-INVOKE-PDU:
   - Reset invoke reference number timer.
8. Invoke reference number timer:

- Release the invoke reference number.

On receipt of an ESRO-INVOKE-PDU (as a single PDU or reassembled from a sequence of segment PDUs), the ESROS provider issues an ESROS-INVOKE.indication to the ESROS performer user. The provider ignores the duplicate ESRO-INVOKE-PDUs.

In the case of internal failure or no response from performer user, the provider sends an ESRO-FAILURE-PDU and releases the invoke reference number.

On receipt of either ESROS-RESULT.request or ESROS-ERROR.request from the ESROS performer user, the ESROS provider initiates the inactivity timer for the ESRO- RESULT-PDU or ESRO-ERROR-PDU and transmits the ESRO-RESULT-PDU or ESRO-ERROR-PDU (in a single PDU or as a sequence of segment PDUs.) If a duplicate ESRO-INVOKE-PDU associated with the Invoke-ID is received within INACTIVITY_TIME interval (see 4.6.2), the PDU is retransmitted.

If no duplicate ESRO-INVOKE-PDU is received within the INACTIVITY_TIME interval (see 4.6.2), provider issues a ESROS-RESULT.confirm or ESROS-ERROR.confirm primitive and sets the reference number timer and releases the reference number after REFERENCE_NUMBER_TIME (see 4.6.2).

The duplicate ESRO-INVOKE-PDUs are ignored while provider waits for the reference number timer to expire.

4.3.4 Segmentation and Reassembly

Small ESRO Service Data Units (ESRO-SDUs) can benefit from the efficiencies of connectionless feature of ESROS (See Section 4.3.1).

When an ESRO-SDU is too large to fit in a single connectionless PDU it is segmented and reassembled. There might be similar mechanisms in the upper layers with different levels of efficiency. When in addition to the ESROS segmentation/reassembly, the upper layers are capable of segmentation/reassembly services, then the ESROS user can decide whether to use ESROS segmenting/reassembly mechanism depending on the factors such as reliability of the underlying network.

In the case of segmentation/reassembly in ESROS layer, transmission of operation segments is not acknowledged. This results in an efficient transmission over a reliable underlying network. However failure of one segment results in retransmission of all segments.

When acknowledged segments are desired, the ESROS user should implement it using the acknowledged result service of ESROS.

The ESROS segmentation/reassembly is accommodated by:

- Use of two additional PDU codes for segmented INVOKE PDU.
- Use of one byte segmentation information, which contains First/Other flag and segment number.
- Use of unused bits of RESULT and ERROR PDUs to identify a segmented RESULT or ERROR PDU.

Segmentation and Assembly applies to INVOKE, RESULT, and ERROR SDUs.

The sender of the message is responsible for segmenting the ESRO-SDU into segments that fit in CL PDUs. The segmented ESRO-SDU is sent in a sequence of segments each carrying a segment of the SDU. The Invoke-Reference-Number is a unique identifier that is used as the segment identifier which relates all segments of an ESRO-SDU. In addition to this identifier, the first segment specifies the total number of segments (number-of-segments). Other segments have a segment sequence number (segment- number). The receiver is responsible for sequencing (based on segment-number) and reassembling the entire ESRO-SDU.

Segmenting/Reassembling over the Connectionless ESRO Service

The sender maps the original ESRO-SDU into an ordered sequence of segments. Several ESRO-SDU segment sequences can exist over the same ESROS association, distinguished by their Invoke-Reference-Number (used as segment identifier.)

All segments in the sequence have the same Invoke-Reference-Number assigned by sender.

The first segment specifies the total number of segments. All segments in the sequence except the first one shall be sequentially numbered, starting at 1 (first segment has an implicit segment number of 0).

Each segment is transmitted in one UDP PDU and is sent by sender. All segments of a segmented ESRO-SDU are identified by the same Invoke-Reference-Number. For a given operation, the receiver should not impose any restrictions on the order of arrival of segments.

There is no requirement that any segment content be of CLRO_SMALL_PDU_MAX_SIZE for connectionless transmission; however, no more than CLRO_MAX_PDU_SEGMENTS segments can be derived from a single ESRO-SDU.

The receiver reassembles a sequence of segments into a single ESRO-SDU. An ESRO-SDU shall not be further processed unless all segments of the ESRO-SDU are received. Failure to receive the SDU shall be determined by the following event:

- Expiration of Reassembly Timer (see Section 4.3.4).

In the event of the above mentioned failure, the receiver shall discard a partially assembled sequence.

The reassembly is done as described below:

- In the case of segmented Invoke ESRO-SDU, the encoding type and operation-value fields are carried in the first segment used for the whole operation. These three fields are ignored in the segments other than the first one.
- In the case of segmented Result ESRO-SDU, the encoding type of the first segment is used for all segments. The encoding type field of segments other than the first one are ignored.
- In the case of segmented Error ESRO-SDU, the encoding type and Error-value field of the first segment are used for all segments. These two fields are ignored in segments other than the first one.

PDU Name	PDU Type Code
ESRO-INVOKE	0
ESRO-RESULT	1
ESRO-ERROR	2
ESRO-ACK	3
ESRO-FAILURE	4
ESRO-SEGMENTED-INVOKE	5

Table 15: PDU Coding

Sender sends all segments of a segmented ESRO-SDU one after the other. There is no mechanism for re-transmission of a single segment. In the case that the sender receives a failure indication for a segment, it means that receiver has failed in reassembly process, and the sender retransmits the whole ESRO-SDU (all segments).

Reassembly Timer

The Reassembly Timer is a local timer maintained by the receiver of the segments that assists in performing the reassembly function. This timer determines how long a receiver waits to receive all segments of a segment sequence.

The Reassembly Timer shall be started on receipt of a segment with different sequence identifier (Invoke-Reference-Number). On receipt of all segments composing a sequence, the corresponding reassembly timer shall be stopped.

The value of the Reassembly Timer is defined based on the network characteristics and the number of segments. This requires that the transmission of all segments of a single ESRO-SDU must be completed within this time limit.

4.4 Structure and Encoding of ESROS PDUs

Five PDU types are used in the ESRO protocol which are described in the following sections. PDU type coding is presented in Table 15.

The octets are numbered in increasing order, starting from 1. The bits of an octet are numbered from 1 to 8, where 1 is the low-order bit.

4.4.1 ESRO-INVOKE-PDU Format

Bit string format of the ESRO-INVOKE-PDU is represented in Table 16 and Table 17.

```
|Bit       |8 |          7           |6 |5 |4 |3 |2 |1 |
|----------+--+----------------------+--+--+--+--+--+--|
|Octet 1   | Performer SAP                |0 |0 |0 |0 |
|----------+--+----------------------+--+--+--+--+--+--|
|Octet 2   | Invoke Reference Number                    |
|----------+--+----------------------+--+--+--+--+--+--|
|Octet 3   |Parameter Encoding Type  |Operation Value   |
|----------+--+----------------------+--+--+--+--+--+--|
|Octet 4   |                                            |
| ...      | Operation Information                      |
|Octet N   |                                            |
|__________|____________________________________________|
```

Table 16: ESRO-INVOKE-PDU format. ESRO-INVOKE-PDU Type Code = 0. Note: Invoker SAP = Performer SAP - 1.

Value	Meaning
0	BER [5]
1	PER [4]
2	XDR [8]
3	Reserved

Table 17: Parameter Encoding Type for ESRO-INVOKE-PDU

```
 ---------------------------------------------------------
|Bit     |  8  |  7  |  6  |  5  |  4  |  3  |  2  |  1  |
|--------|-----------|-----|-----|-----|-----|-----|-----|
|        | Parameter |     |     |     |     |     |     |
|        | Encoding  |     |     |     |     |     |     |
|Octet 1 | Type      |  0  |  0  |  0  |  0  | 0   |  1  |
|        |           |     |     |     |     |     |     |
|        |           |     |     |     |     |     |     |
|        |           |     |     |     |     |     |     |
|--------|-----------------------------------------------|
|Octet 2 |           Invoke Reference Number             |
|--------|-----------------------------------------------|
|Octet 3 |                                               |
|...     |             Result-parameter                  |
|Octet N |                                               |
 ---------------------------------------------------------
ESRO-RESULT-PDU Type Code = 1.
```

Table 18: ESRO-RESULT-PDU format

Value	Meaning
0	Basic
1	Packed
2	XDR
3	Reserved

Table 19: Parameter Encoding Type for ESRO-RESULT-PDU

4.4.2 ESRO-RESULT-PDU Format

Bit string format of the ESRO-RESULT-PDU is represented in Table 18 and Table 19.

4.4.3 ESRO-ERROR-PDU Format

Bit string format of the ESRO-ERROR-PDU is represented in Table 20 and Table 21.

4.4.4 ESRO-ACK-PDU Format

Bit string format of the ESRO-ACK-PDU is represented in Table 22 and Table 23.

```
 ----------------------------------------------------------
|Bit     |  8  |  7  |  6  |  5  |  4  |  3  |  2  |  1  |
|--------|-----------|-----|-----|-----|-----|-----|-----|
|        | Parameter |     |     |     |     |     |     |
|        | Encoding  |     |     |     |     |     |     |
|Octet 1 | Type      |  0  |  0  |  0  |  0  | 1   |  0  |
|        |           |     |     |     |     |     |     |
|        |           |     |     |     |     |     |     |
|        |           |     |     |     |     |     |     |
|--------|-----------------------------------------------|
|Octet 2 |           Invoke Reference Number             |
|--------|-----------------------------------------------|
|Octet 3 |               Error Value                     |
|--------|-----------------------------------------------|
|Octet 4 |                                               |
|...     |             Error parameter                   |
|Octet N |                                               |
 ----------------------------------------------------------
ESRO-ERROR-PDU Type Code = 2.
```

Table 20: ESRO-ERROR-PDU format

Value	Meaning
0	Basic
1	Packed
2	XDR
3	Reserved

Table 21: Parameter Encoding Type for ESRO-ERROR-PDU

```
 ----------------------------------------------------------
|Bit     |  8  |  7  |  6  |  5  |  4  |  3  |  2  |  1  |
|--------|-----------------------|-----|-----|-----|-----|
|        | ESRO-ACK-PDU Type     |     |     |     |     |
|Octet 1 |                       |  0  |  0  | 1   |  1  |
|--------|-----------------------------------------------|
|Octet 2 |           Invoke Reference Number             |
 ----------------------------------------------------------
 ESRO-ACK-PDU Type Code = 3.
```

Table 22: Fields of ESRO-ACK-PDU

ESRO-ACK-PDU Type	Meaning
0	Complete 3-way handshake
1	Hold on

Table 23: Encoding of ESRO-ACK-PDU Type

Bit	8	7	6	5	4	3	2	1
Octet 1	Not used				0	1	0	0
Octet 2	Invoke Reference Number							
Octet 3	Failure Value							

ESRO-FAILURE-PDU Type Code = 4.

Table 24: ESRO-FAILURE-PDU format

4.4.5 ESRO-FAILURE-PDU Format

Bit string format of the ESROS-FAILURE-PDU is represented in Table 24 and Table 25.

The first nibble of the first octet of ESRO-FAILURE-PDU shall be set to zero.

Failure Value	Meaning
0	Transmission failure
1	Out of local resources
2	User not responding
3	Out of remote resources

Table 25: Encoding of failure value

```
 ---------------------------------------------------------------
|Bit     |  8   |  7   |  6   |  5   |  4   |  3   |  2   |  1   |
|--------|---------------------------|------|------|------|------|
|Octet 1 | Performer Service         |  0   |  1   |  0   |  1   |
|        | Access Point Selector     |      |      |      |      |
|--------|------------------------------------------------------|
|Octet 2 |           Invoke Reference Number                    |
|--------|------------------------------------------------------|
|        | Parameter    |                                       |
|        | Encoding     |                                       |
|Octet 3 | Type         |        Operation Value                |
|        |              |                                       |
|        |              |                                       |
|        |              |                                       |
|--------|------------------------------------------------------|
|Octet 4 |First/|                                               |
|        |Other |                Segment Number                 |
|--------|------------------------------------------------------|
|Octet 5 |                                                      |
|...     |            Operation Information                     |
|Octet N |                                                      |
 ---------------------------------------------------------------|
 ESRO-INVOKE-PDU Type Code = 5.
```

Table 26: ESRO-INVOKE-SEGMENTED-PDU format

4.4.6 ESRO-INVOKE-SEGMENTED-PDU Format

Bit string format of the ESRO-INVOKE-SEGMENTED-PDU is represented in Table 25 and Table 26.

Note: Invoker SAP = Performer SAP - 1.

- For the first segment, the first/other bit is set to one, and the segment number field contains the total number of segments.
- For segments other than the first one, the first/other bit is set to zero, and the segment number field has the sequence number of the segment.

The values of the three fields Performer-SAP, Parameter-Encoding-Type, and Operation-Value of the first segment are used by performer and these fields are ignored in the segments other than the first one.

4.4.7 ESRO-RESULT-SEGMENTED-PDU Format

Bit string format of the ESRO-RESULT-SEGMENTED-PDU is represented in Table 28 and Table 29.

- For the first segment, the first/other bit is set to one, and the segment number field contains the total number of segments.

Value	Meaning
0	Basic
1	Packed
2	XDR
3	Reserved

Table 27: Parameter Encoding Type for ESRO-INVOKE-SEGMENTED-PDU

```
 -----------------------------------------------------------------
|Bit     |  8   |  7   |  6   |  5   |  4   |  3   |  2   |  1   |
|--------|-------------|------|------|------|------|------|------|
|        | Parameter   |      |      |      |      |      |      |
|        | Encoding    |      |      |      |      |      |      |
|Octet 1 | Type        |  0   |  1   |  0   |  0   |  0   |  1   |
|        |             |      |      |      |      |      |      |
|        |             |      |      |      |      |      |      |
|        |             |      |      |      |      |      |      |
|--------|------------------------------------------------------|
|Octet 2 |            Invoke Reference Number                   |
|--------|------------------------------------------------------|
|Octet 4 |First/|                                               |
|        |Other |                 Segment Number                |
|--------|------------------------------------------------------|
|Octet 5 |                                                      |
|...     |                 Result Parameter                     |
|Octet N |                                                      |
 ---------------------------------------------------------------|
ESRO-RESULT-SEGMENTED-PDU Type Code = 1.
```

Table 28: ESRO-RESULT-SEGMENTED-PDU format

Value	Meaning
0	Basic
1	Packed
2	XDR
3	Reserved

Table 29: Parameter Encoding Type for ESRO-RESULT-SEGMENTED-PDU

- For segments other than the first one, the first/other bit is set to zero, and the segment number field has the sequence number of the segment.

The values of the Parameter-Encoding-Type field of the first segment is used by invoker and this field is ignored in the segments other than the first one.

4.4.8 ESRO-ERROR-SEGMENTED-PDU Format

Bit string format of the ESRO-ERROR-PDU is represented in Table 30 and Table 31.

- For the first segment, the first/other bit is set to one, and the segment number field contains the total number of segments.
- For segments other than the first one, the first/other bit is set to zero, and the segment number field has the sequence number of the segment.

The values of the Parameter-Encoding-Type field of the first segment is used by invoker and this field is ignored in the segments other than the first one.

4.5 Concatenation and Separation

The procedure for concatenation and separation conveys multiple ESRO-PDUs in one TSDU. This is accomplished by ESRO-CONCATENATED-PDU.

An ESRO-CONCATENATED-PDU can contain one or more of the following PDUs: INVOKE, RESULT, ERROR, FAILURE, and ACK.

The ESRO-PDUs within a concatenated set may be distinguished by means of the length indicator. A one byte length indicator comes before each ESRO-PDU.

The number of ESRO-PDUs in an ESRO-CONCATENATED-PDU is bounded by the maximum length of TSDU.

Bit	8	7	6	5	4	3	2	1
Octet 1	Parameter Encoding Type		0	1	0	0	1	0
Octet 2	Invoke Reference Number							
Octet 3	First/ Other	Segment Number						
Octet 4	Error Value							
Octet 5 ... Octet N	Error Parameter							

ESRO-ERROR-SEGMENTED-PDU Type Code = 2.

Table 30: ESRO-ERROR-SEGMENTED-PDU

Value	Meaning
0	Basic
1	Packed
2	XDR
3	Reserved

Table 31: Parameter Encoding Type for ESRO-SEGMENTED-ERROR-PDU

4.5.1 Procedures

Concatenation

The ESROS provider concatenates PDUs as follows:

- PDU type code 8 is used.
- The length indicator which is the total length of first ESRO-PDU (header and data) in octets is placed after PDU type code in length indicator field of ESRO- CONCATENATED-PDU (see Section 4.5.2).
- The first PDU (header and data) is placed after the length indicator field and in the ESRO-PDU field of ESRO-CONCATENATED-PDU (see Section 4.5.2).
- For any additional ESRO-PDU, the length indicator and PDUs are concatenated.

Separation

When the ESRO service provider receives a PDU with PDU type code 8, it separates the concatenated PDUs as described below:

- Length indicator field coming after type code field (see Section 4.5.2) specifies the total length of the first PDU in octets.
- The first PDU is in the ESRO-PDU field after the length indicator field (see Section 4.5.2).
- Any additional PDU has its length indicator field specifying the total length of PDU, followed by PDU itself (see Section 4.5.2).
- PDUs are separated until the end of the ESRO-CONCATENATED-PDU is reached.

4.5.2 ESRO-CONCATENATED-PDU format

Bit string format of the ESRO-CONCATENATED-PDU containing multiple concatenated ESRO-PDUs is represented in Table 32.

Length Indicator field

This field is contained in one octet and comes before each ESROS-PDU in the concatenated PDU. The length indicated is total length of the ESRO-PDU (including header and data) coming after it in octets.

```
 _________________________________________________________________
|Bit     |  8  |  7  |  6  |  5  |  4  |  3  |  2  |  1  |
|--------|---------------------------|------|------|------|------|
|Octet 1 |         Not used          |  1   |  0   |  0   |  0   |
|--------|--------------------------------------------------------|
|Octet 2 |                  Length Indicator                      |
|--------|--------------------------------------------------------|
|Octet 3 |                                                        |
|...     |                     ESRO-PDU                           |
|Octet N |                                                        |
|--------|--------------------------------------------------------|
|Octet   |                                                        |
| N+1    |                  Length Indicator                      |
|--------|--------------------------------------------------------|
|Octet   |                                                        |
| N+2    |                     ESRO-PDU                           |
| ...    |                                                        |
|--------|--------------------------------------------------------|
| ...    |                       ...                              |
|________|________________________________________________________|
ESRO-CONCATENATED-PDU Type Code = 8.
```

Table 32: ESRO-CONCATENATED-PDU format

ESRO-PDU field

This field contains an ESRO-INVOKE-PDU, ESRO-RESULT-PDU, ESRO-ERROR- PDU, ESRO-FAILURE-PDU, or ESRO-ACK-PDU.

The length of this field is specified by the length indicator field coming before it.

4.6 ES Remote Operations Protocol Parameters

4.6.1 PDU size

- CLRO_SMALL_PDU_MAX_SIZE:

  The value of this parameter should be chosen based on the specifics of the subnetwork in use. For example, in CDPD the maximum size of SN-Userdata size can be up to 2048 bytes (see part 404-2.b of CDPD Specification V1.1). Based on this value and IP and UDP protocol information fields, the value of CLRO_SMALL_PDU_MAX_SIZE may be determined for CDPD. Again based on the specifics of the subnetwork, the optimum value of CLRO_SMALL_PDU_MAX_SIZE may best be determined based on field experience and may be smaller than the maximum size that the subnetwork supports.

- CLRO_SEGMENTED_PDU_MAX_SIZE

  The value of this parameter should be chosen based on the specifics of the subnetwork in use. The optimum value of CLRO_SEGMENTED_PDU_MAX_SIZE may best be determined based on field experience.

- CLRO_MAX_PDU_SEGMENTS

  The value of this parameter should be chosen based on the specifics of the subnetwork in use. The optimum value of CLRO_MAX_PDU_SEGMENTS may best be determined based on field experience. In any case, this value should be smaller than 127.

4.6.2 Timers

- INVOKE_PDU_RETRANSMISSION_INTERVAL:

  The INVOKE_PDU retransmission interval should be specified and optimized based on the characteristics of the network in use.

- RESULT_ERROR_PDU_RETRANSMISSION_INTERVAL:

  The RESULT and ERROR-PDU retransmission interval should be specified and optimized based on the characteristics of the network in use.

- MAX_RETRANSMISSIONS:

  The maximum number of retransmissions should be specified and optimized based on the characteristics of the network in use.

- INACTIVITY_TIME:

  The minimum waiting time during which no duplicate PDU is received should be specified and optimized based on the characteristics of the network in use.

- REFERENCE_NUMBER_TIME: The reference number lifetime timer should be specified and optimized based on the characteristics of the network in use.

4.6.3 Use of lower layers

ESRO protocol uses UDP port number 259.

5 ACKNOWLEDGMENTS

Development of this specification was funded by AT&T Wireless Services (AWS). This protocol specification has been derived from AT&T Wireless Services' document titled: "Limited Size Remote Operation Services (LSROS)", Revision 0.8, dated April 20, 1995.

This specification is technically consistent with CDPD Forum's Implementor's Guidelines Part 1028, Release 1.03, June 21, 1996.

6 SECURITY CONSIDERATIONS

ESROS has no authentication mechanism. Authentication is the responsibility of the performer (which is outside of the scope of ESROS) and the performer is not expected to honor the invoker when it is not authenticated.

7 AUTHORS' ADDRESSES

Mohsen Banan
Neda Communications, Inc.
17005 SE 31st Place
Bellevue, WA 98008
email: http://mohsen.banan.1.byname.net/ContactMe

Mark S. Taylor
Director of Strategic Engineering
AT&T Wireless Services
Wireless Data Division
10230 NE Points Drive
Kirkland, WA 98033-7869 USA
email: mark.taylor@airdata.com

Jia-bing Cheng
AT&T Wireless Services
Wireless Data Division
10230 NE Points Drive
Kirkland, WA 98033-7869 USA
email: jcheng@airdata.com

References

[1] Remote Operations: Model, Notation and Service Definition. The International Telegraph and Telephone Consultative Committee, March 1988. Recommendation X.219.

[2] Remote Operations: Protocol Specification. The International Telegraph and Telephone Consultative Committee, March 1988. Recommendation X.229.

[3] Specification of Abstract Syntax Notation One. The International Telegraph and Telephone Consultative Committee, 1988. Recommendation X.208.

[4] Information Processing — Open Systems Interconnection — Specification of Packed Encoding Rules for Abstract Syntax Notation One (ASN.1). International Organization for Standardization and International Electrotechnical Committee. International Standard 8825-2.

[5] Information Processing — Open Systems Interconnection — Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1). International Organization for Standardization and International Electrotechnical Committee, 1987. International Standard 8825.

[6] R. Srinivasan. Binding Protocols for ONC RPC Version 2. RFC 1833 (Proposed Standard), August 1995.

[7] R. Srinivasan. RPC: Remote Procedure Call Protocol Specification Version 2. RFC 1831 (Proposed Standard), August 1995. Obsoleted by RFC 5531.

[8] R. Srinivasan. XDR: External Data Representation Standard. RFC 1832 (Draft Standard), August 1995. Obsoleted by RFC 4506.