# Functional Specification: Intra-Cluster Communications

*ICC System Services*

*Rev 0.9 (Review copy, not yet reviewed outside team)*

Last Update: 05-Aug-1996 23:06

Author: John Covert

**Revision History**

| Rev | Date | Author | Description |
|---|---|---|---|
| 0.9 | 5-Aug-1996 | John Covert | Document for group review, after team internal review |
| 1.0 | TBS | John Covert | Update after group review (not yet done) |

# Table of Contents

## 1. Problem Statement

The most significant distinguishing technology available in OpenVMS is the VMScluster capability. Developing this capability and enhancing the ability of VMSclusters to support the high end of the three-tier client server model is key to the success of OpenVMS and Digital Equipment Corporation.

The ability to provide an environment in which middleware providers can implement highly reliable and highly available servers within a VMScluster will further enhance the distinguishing characteristics of OpenVMS. To do this, we must provide reliable and efficient Intracluster Communications to allow applications within a cluster but distributed over several nodes to communicate with each other.

There are currently no supported Intracluster Communications methods other than those provided by networking products such as DECnet and TCP/IP. Independence from these products, which vary in availability from one system to another, is seen as a critical enhancement to OpenVMS. There are numerous Intracluster Communications needs today which cannot be met because of the lack of knowledge about which network product might be installed. We know of internal requirements and we believe there are also external requirements.

We currently believe that the needs in this space are primarily by OpenVMS components, middleware providers, and database vendors. There are also some customers who would write to a VMScluster specific API to gain access to high speed interconnects such as Memory Channel.

## 1.1. Goals

1. Provide an easy to use conventional system service interface for inter-process communications within a cluster.
2. Provide an interface callable from all modes and from execlets as well as from images.
3. Provide an easy-to-use interface giving access to high-speed interconnects such as Memory Channel.
4. Provide an interface independent of the installation of any networking product.
5. Provide an interface usable for communications between processes within a single, non-clustered node.
6. Provide an interface usable for non-privileged clients and authorized but not necessarily privileged servers.
7. Provide interoperability with any existing users of the existing IPC interface.
8. Allow migration to the new interface without concern for mixed-version operation.

## 1.2. Non-goals

1. Neither guarantee nor break operation over transports other than local or SCS.

# 2. Terminology

## 2.1. Intra-Cluster Communications

Intra-Cluster Communications refers to communications between two cooperating processes within the same VMScluster.

## 2.2. Networking Products

Networking Products refer to layered products purchased under a separate license from the base operating system, whether packaged with the base operating system or separately distributed. There is no guarantee that a system will have any Networking Products installed.

## 2.3. Systems Communications Services

Systems Communications Services (SCS) is a term referring to the integrated communications services provided as part of the base operating system but available for use if and only if a VMScluster license is activated. It is usable only between nodes of the same VMScluster.

## 2.4. VMScluster Interconnect

A VMScluster Interconnect is any device provided with associated drivers and other necessary routines to allow the use of Systems Communications Services for communications over the device. Examples are the CI, the DSSI, the NI, FDDI, and Memory Channel. A SCSI bus is not a VMScluster Interconnect.

## 2.5. IPC

Within this specification, the term IPC refers to the existing interprocess communications services running primarily over SCS. It refers to both of its two forms: (1) a kernel mode only interface and (2) a user-written system service. It is not to be confused with the $IPC service supplied with DECnet. The ICC System Services are an interface to IPC.

## 2.6. ICC System Services

ICC System Services are a new interface to IPC described in this document. As a “real” system service, these services may be called from sufficiently authorized user programs as well as from within the executive.

## 2.7. Association

An Association is the relationship between a process using IPC and IPC itself. The name of an Association can be used by other processes to establish a Connection to a Server using IPC. A process may establish any number of associations, up to a current systemwide limit of 256.

### 2.8. Connection

A Connection is a logical communication channel between two cooperating processes. Each connection using IPC uses one of the associations for its cooperating process. There is no requirement that the assocations have the same name in order for processes to establish a connection.

### 2.9. Client

A client program, in general, establishes a connection to request a service from a server. A client may, however, operate as a server to other clients while acting as a client to other servers.

### 2.10. Server

A server program, in general, advertises a named service and waits for connections from clients wishing to use this service. A server may, however, act as a client to other servers while acting as a server to other clients.

### 2.11.Authentication

Authentication refers to a capability of allowing two cooperating processes to reliably learn each other's identity. Within a single operating system, such as VMS, the login process provides authentication of a user, and non-privileged processes are constrained to retain that initial identity. When a process communicates outside its original system, other mechanisms for Authentication are necessary.

### 2.12.Authorization

Authorization refers to a capability of allowing an appropriately authenticated subject access to a particular service or object.

## 3. New Features

### 3.1. Restrictions being lifted

The existing IPC interface has been implemented in a manner which requires two completely different programming paradigms depending on whether the interface is being used from within the executive (such as from a hypothetical QIO server implemented as an execlet) or from within a normal image. The ICC System Services eliminate this restriction by providing a new interface, which is much easier to use than either of the two existing interfaces.

### 3.2. Syntax and Semantics Changes

A completely new set of services with new syntax and semantics are provided as part of this project, described later in this document.

# 4. Data structure changes

## 4.1. Existing IPC Data Structures

To implement the first phase of this project, no changes at all are being made to any existing IPC Data Structures. When 64-bit support is added early in 1997, it is expected that there will be changes necessary. We plan to manage these changes to ensure that existing users of the old IPC interfaces are not affected.

## 4.2. New data structures

There are new data structures to manage the use of IPC from users of the ICC System Services. These are not visible to users, and as such are described in the design spec.

# 5. ICC System Services

## 5.1. Open Association: $ICC_OPEN_ASSOC

The Open Association service declares an application association with IPC. Servers must make this call in order to declare/register their name and indicate their readiness to receive incoming connections. Although a client is permitted to call this routine, it is not necessary for simple applications not needing to handle incoming connections or disconnect event ASTs. A client may open a connection without specifying an open association; this will automatically create a default association of ICC$PID_nnnnnnnn (where nnnnnnnn is the character representation of their Process ID).

NETMBX privilege is required to open any association.

The Association Namespace is a controlled resource, managed as described in the section entitled ***The Association Namespace*** (q.v.). An attempt to open an association with a name not authorized as described in this section will fail with the error SS$_NOSYSNAM returned to the caller. In addition to making entries in the system's local association namespace, a call to $ICC_OPEN_ASSOCIATION may also make an entry in a simple clusterwide registry of active assocations.

An association opened in an inner mode is not visible to any outer mode. Opening the default association in an inner mode will prevent any outer modes from using the default association.

An application may open any number of associations. Currently, there is a systemwide limit of 256 open associations. There is no limit imposed clusterwide.

```
$ICC_OPEN_ASSOC             assoc_handle, assoc_name, [logical_name],
                            [logical_table], [event_rtn], [recv_rtn],
                            [maxbuf], [maxrec], [maxxmt]

    assoc_handle    Type:       longword (unsigned)
                    Access:     write only
                    Mechanism:  by 32-bit or 64-bit reference (Alpha)
```

```
                by 32-bit reference (VAX)

                The 32-bit or 64-bit address (on Alpha systems) or
                the 32-bit address (on VAX systems) of a longword
                into which $ICC_OPEN_ASSOC writes the handle
                assigned to the opened association.

assoc_name      Type:        character-coded text string
                Access:      read only
                Mechanism:   by 32-bit or 64-bit descriptor (Alpha)
                             by 32-bit descriptor (VAX)

                An ASCII character string of up to 32 characters
                in length specifying the name of the application
                opening the association.  Null (0 length) and
                empty/blank association names are not allowed.
                If this argument is omitted (i.e., a zero is
                passed in by value), it signifies that the user
                wants to open the default association.

logical_name    Type:        character-coded text string
                Access:      read only
                Mechanism:   by 32-bit or 64-bit descriptor (Alpha)
                             by 32-bit descriptor (VAX)

                A logical name in a clusterwide logical name table
                used to maintain the simple association registry.

logical_table   Type:        character-coded text string
                Access:      read only
                Mechanism:   by 32-bit or 64-bit descriptor (Alpha)
                             by 32-bit descriptor (VAX)

                The table containing the logical name 'logical_name'.

event_rtn       Type:        procedure_entry_mask
                Access:      call without stack unwinding
                Mechanism:   by 32-bit or 64-bit linkage reference
                             (Alpha)
                             by 32-bit reference (VAX)

                The address of the AST routine to be called for
                incoming connect/disconnect events.  This routine
                will be called in the mode-of-the-caller (no
                mechanism is provided for the routine to be called
                at a different mode)

recv_rtn        Type:        procedure_entry_mask
                Access:      call without stack unwinding
                Mechanism:   by 32-bit or 64-bit linkage reference
                             (Alpha)
                             by 32-bit reference (VAX)

                The address of the AST routine to be called for
                incoming new data events.

                If the user provides this routine it indicates that
                buffers greater than the size specified in 'maxrec'
```

```
                    will be supplied by the user any time they are
                    requested.  If the user supplies this routine,
                    receive calls should only be issued after receive
                    events arrive and sufficient buffer space has been
                    specified to handle the incoming data.

                    This routine will be called in the mode-of-the-
                    caller (no mechanism is provided for the routine
                    to be called at a different mode)

maxbuf              Type:       longword (unsigned)
                    Access:     read only
                    Mechanism:  by value

                    The maximum number of pending inbound messages
                    (per connection) that IPC will allow the user
                    before initiating flow control.  A message is
                    pending if it is being held within IPC but no
                    receive call(s) are outstanding from the user.

                    Default = 5

maxrec              Type:       longword (unsigned)
                    Access:     read only
                    Mechanism:  by value

                    The maximum size (in bytes) of all receive buffers
                    that will be specified on any/all connection using
                    this association.

                    Default = 256

maxxmt              Type:       longword (unsigned)
                    Access:     read only
                    Mechanism:  by value

                    The maximum size (in bytes) of all transmit buffers
                    that will be specified on any/all connection using
                    this association.

                    Default = 256


Condition Values returned       Type:       VMS cond_code
                                Mechanism:  R0

    Successful completion return codes:
        SS$_NORMAL       - Normal completion

    Error completion return codes:
        SS$_ACCVIO       - Access violation on parameter
        SS$_BADPARAM     - Bad parameter value specified
        SS$_DUPLNAM      - Specified association name is already
                           registered (already exists).
        SS$_EXQUOTA      - One or more process quotas has been exceeded
        SS$_SSFAIL       - The specified association was marked for
                           rundown or unable to allocate an necessary
                           resource.
```

```
        SS$_INSFARG      - Too few arguments were supplied
        SS$_INSFMEM      - Not enough system resources or process
                           virtual memory available.
        SS$_INSFWSL      - Process working set limit too small
        SS$_NONETMBX     - Request requires NETMBX privilege
        SS$_NOSYSNAM     - Request requires SYSNAM privilege
        SS$_TOO_MANY_ARGS- Too many arguments were specified
        and any failures from the system services:
            $ENQ, $DEQ, $CRELNM, $TRNLNM
```

### 5.2. Close Association: $ICC_CLOSE_ASSOC

The Close Association System Service closes the application's association with IPC. If multiple associations are open, only the specified association is closed. When an association is closed, any active connections on that association are disconnected. All associations opened in user mode will be closed at image exit; associations opened in inner modes will be closed at process termination.

When an association is closed, the entry (if any) in the simple clusterwide association registry is removed.

```
$ICC_CLOSE_ASSOC            assoc_handle

    assoc_handle            Type:        longword (unsigned)
                            Access:      read only
                            Mechanism:   by value

                            The handle of the association to be closed.


    Condition Values returned       Type:        VMS cond_code
                                    Mechanism:   R0

        Successful completion return codes:
            SS$_NORMAL       - Normal completion

        Error completion return codes:

            SS$_INSFARG      - The assoc_handle was not supplied
            SS$_IVCHAN       - Invalid association handle
            SS$_TOO_MANY_ARGS- Too many arguments were specified
```

### 5.3. Connect: $ICC_CONNECT and $ICC_CONNECTW

The Connect System Service establishes a connection to a remote application over an open association. Connections must be opened in the same mode as their association. If the user provides an association handle of zero, the default association will be used; it will be opened if it is not already open. Any number of connections are possible over a single association.

At image exit, as a result of closing any open user mode associations, all user mode connections will be disconnected. Inner mode connections are the responsibility of the inner mode code, but will be disconnected at process termination when inner mode associations are closed. Connections opened in an inner mode are not visible to outer modes.

A client opens connections with the Connect service; a server opens connections with the Accept service (q.v.).

```
$ICC_CONNECT              [efn], iosb, [astadr], [astprm],
                          [assoc_handle], conn_handle, remote_assoc,
                          [remote_node], [alias], [conn_buf], [conn_len],
                          [return_buf], [return_len], [retlength]

    efn              Type:       longword (unsigned)
                     Access:     read only
                     Mechanism:  by value

                     The event flag number to wait on.

    iosb             Type:       quadword (unsigned)
                     Access:     write only
                     Mechanism:  by 32-bit or 64-bit reference (Alpha)
                                 by 32-bit reference (VAX)
                                           +2                      0
                     -------------------------------------------------
                     |       undefined        |   completion status   |
                     -------------------------------------------------
                     |        unused          |        unused         |
                     -------------------------------------------------
                                           +6                     +4

                     I/O status block

                     Successful completion status values:
                         SS$_NORMAL
                     Failure completion status values:
                         SS$_BUFFEROVF, SS$_DISCONNECT, SS$_EXQUOTA,
                         SS$_INSFMEM, SS$_IVBUFLEN,
                         SS$_LINKABORT, SS$_LINKDISCON,
                         SS$_NOLOGNAM, SS$_NOSUCHID, SS$_NOSUCHNODE,
                         SS$_PATHLOST, SS$_REJECT, SS$_SSFAIL,
                         SS$_UNREACHABLE, SS$_WRONGSTATE

    astadr           Type:       procedure_entry_mask
                     Access:     call without stack unwinding
                     Mechanism:  by 32-bit or 64-bit linkage reference
                                 (Alpha)
                                 by 32-bit reference (VAX)

                     The AST routine to be executed when the operation
                     completes.

    astprm           Type:       quadword (unsigned) (Alpha)
                                 longword (unsigned) (VAX)
                     Access:     read only
                     Mechanism:  by 64-bit value (Alpha)
                                 by 32-bit value (VAX)

                     The parameter to be passed to the AST routine.

    assoc_handle     Type:       longword (unsigned)
                     Access:     read only
                     Mechanism:  by value
```

The handle of the association on which the connection is to be opened. If zero or not specified the default association is to be used (and opened if necessary).

| | | |
|---|---|---|
| conn_handle | Type: | longword (unsigned) |
| | Access: | write only |
| | Mechanism: | by 32-bit or 64-bit reference (Alpha)<br>by 32-bit reference (VAX) |

The 32-bit or 64-bit address (on Alpha systems) or the 32-bit address (on VAX systems) of a longword into which $ICC_CONNECT writes the connection handle of the created connection On a successful call.

| | | |
|---|---|---|
| remote_assoc | Type: | character-coded text string |
| | Access: | read only |
| | Mechanism: | by 32-bit or 64-bit descriptor (Alpha)<br>by 32-bit descriptor (VAX) |

An ASCII character string (32 characters max.) specifying the name of the target application to connect to. This input is used to qualify the 'remote_node' argument below.

| | | |
|---|---|---|
| remote_node | Type: | character-coded text string |
| | Access: | read only |
| | Mechanism: | by 32-bit or 64-bit descriptor (Alpha)<br>by 32-bit descriptor (VAX) |

The name of the node where the target association resides. A null or blank string can be used to indicate the local node. If zero or not specified, the simple clusterwide association registry is to be used.

| | | |
|---|---|---|
| alias | Type: | longword (unsigned) |
| | Access: | read only |
| | Mechanism: | by value |

A user specified Connection Identifier to be subsequently returned on any disconnect or data events on this connection.

| | | |
|---|---|---|
| conn_buf | Type: | quadword (unsigned) (Alpha)<br>longword (unsigned) (VAX) |
| | Access: | read only |
| | Mechanism: | by 32-bit or 64-bit reference (Alpha)<br>by 32-bit reference (VAX) |

A buffer of up to 1024 bytes of connection data to be sent to the target of the connection during the connection process.

Maximum length = 1024 bytes

```
conn_len        Type:       longword (unsigned)
                Access:     read only
                Mechanism:  by value

                The number of bytes in conn_buf to actually be sent.

return_buf      Type:       quadword (unsigned) (Alpha)
                            longword (unsigned) (VAX)
                Access:     read only
                Mechanism:  by 32-bit or 64-bit reference (Alpha)
                            by 32-bit reference (VAX)

                A buffer of up to 1024 bytes in length to receive any
                incoming connection accept or reject data returned.

                Maximum length = 1024 bytes

return_len      Type:       longword (unsigned)
                Access:     read only
                Mechanism:  by value

                The actual length of return_buf.

retlength       Type:       longword (unsigned)
                Access:     write only
                Mechanism:  by 32-bit or 64-bit reference (Alpha)
                            by 32-bit reference (VAX)

                The 32-bit or 64-bit address (on Alpha systems) or
                the 32-bit address (on VAX systems) of a longword
                into which $ICC_CONNECT writes the actual length
                (in bytes) of any user accept/reject data returned
                in the buffer 'return_buf'.


Condition Values returned       Type:       VMS cond_code
                                Mechanism:  R0

    Successful completion return codes:
        SS$_NORMAL

    Error completion return codes:
        SS$_ACCVIO       - Access violation on parameter
        SS$_BADPARAM     - Bad parameter value specified
        SS$_BUFFEROVF    - Overflow on inbound accept/reject data
        SS$_DISCONNECT   - The remote association closed the connection
                           before it was accepted or rejected
        SS$_EXQUOTA      - Not enough AST quota (asynchronous request)
        SS$_INSFARG      - Too few arguments were supplied
        SS$_INSFMEM      - Not enough system resources or process
                           virtual memory available
        SS$_NOSUCHID     - The remote association name and/or node was
                           not found
        SS$_IVBUFLEN     - The length of the context data or accept/
                           reject data buffer is more than 1024 bytes
        SS$_IVMODE       - Attempted to open a connection from a
                           more privileged access mode than the
                           requested association
```

```
SS$_LINKABORT, SS$_LINKDISCON, SS$_PATHLOST
                 - The communications link to the target node
                   was lost.
SS$_NOLOGNAM     - The underlying layers failed to start
                   properly during system initialization
SS$_IVCHAN       - Invalid association handle
SS$_NOSUCHNODE   - The target node is not known
SS$_REJECT       - The remote association or node rejected
                   the connection request
SS$_TOO_MANY_ARGS- Too many arguments were specified
SS$_UNREACHABLE  - Target node currently unreachable
SS$_WRONGACP     - It isn't the local node or a cluster member
                   and the NSP transport isn't initialized yet
SS$_WRONGSTATE   - Connection is in wrong state for request
and any failures from system services:
    $TRNLNM or $ICC_OPEN_ASSOCIATION
```

```
$ICC_CONNECTW              [efn], iosb, [astadr], [astprm],
                           [assoc_handle], conn_handle, remote_assoc,
                           [remote_node], [alias], [conn_buf], [conn_len],
                           [return_buf], [return_len], [retlength]
```

Arguments and returns are the same for $ICC_CONNECTW as for $ICC_CONNECT.

## 5.4. Accept: $ICC_ACCEPT

The Accept System Service is used by a server to respond to an incoming connection request notified in a connect/disconnect AST (q.v.). At the completion of the service, the connection is open and data may be exchanged. Once opened, there is no logical distinction between a connection opened by a client with the Connect service or a server with the Accept service.

A server may reject a request by calling the Reject service (q.v.).

```
$ICC_ACCEPT        conn_handle, [accept_buf], [accept_len], [alias]

    conn_handle    Type:       longword (unsigned)
                   Access:     read only
                   Mechanism:  by value

                   The handle of the requested connection.


    accept_buf     Type:       quadword (unsigned) (Alpha)
                               longword (unsigned) (VAX)
                   Access:     read only
                   Mechanism:  by 32-bit or 64-bit reference (Alpha)
                               by 32-bit reference (VAX)

                   A buffer of up to 1024 bytes of accept data to
                   be sent to the source of the connection at the
                   completion of the connection process.

                   Maximum length = 1024 bytes

    accept_len     Type:       longword (unsigned)
                   Access:     read only
                   Mechanism:  by value
```

```
                The number of bytes in accept_buf to actually be sent.

    alias       Type:       longword (unsigned)
                Access:     read only
                Mechanism:  by value

                A user specified Connection Identifier to be
                subsequently returned on any disconnect or
                data events on this connection.


    Condition Values returned      Type:       VMS cond_code
                                   Mechanism:  R0

        Successful completion return codes:
            SS$_NORMAL       - Normal completion

        Error completion return codes:
            SS$_ACCVIO       - Access violation on parameter
            SS$_BADPARAM     - Bad parameter value specified
            SS$_CLEARED      - Remote association closed the link before
                               it was accepted.
            SS$_INSFARG      - Too few arguments were supplied
            SS$_INSFMEM      - Not enough system resources or process
                               virtual memory available.
            SS$_IVCHAN       - Connection not found or
                               Invalid association handle
            SS$_TOO_MANY_ARGS- Too many arguments were specified
            SS$_WRONGSTATE   - Connection is in wrong state for request
            any failures from routines:
                ALLOCATE_NONPAGED
```

### 5.5. Reject: $ICC_REJECT

The Reject System Service is used by a server to refuse an incoming connection request notified in a connect / disconnect AST (q.v.). After the completion of the service, the client will be notified that the connection has not been opened.

```
$ICC_REJECT           conn_handle, [reject_buf], [reject_len], [reason]

    conn_handle       Type:       longword (unsigned)
                      Access:     read only
                      Mechanism:  by value

                      The handle of the requested connection.

    reject_buf        Type:       quadword (unsigned) (Alpha)
                                  longword (unsigned) (VAX)
                      Access:     read only
                      Mechanism:  by 32-bit or 64-bit reference (Alpha)
                                  by 32-bit reference (VAX)

                      A buffer of up to 1024 bytes of reject data to
                      be sent to the source of the connection at the
                      completion of the rejection process.
```

```
                Maximum length = 1024 bytes

reject_len      Type:       longword (unsigned)
                Access:     read only
                Mechanism:  by value

                The number of bytes in reject_buf to actually be sent.

reason          Type:       longword (unsigned)
                Access:     read only
                Mechanism:  by value

                User specified reject reason code to be supplied
                to the remote application.

                Default = SS$_REJECT


Condition Values returned       Type:       VMS cond_code
                                Mechanism:  R0

    Successful completion return codes:
        SS$_NORMAL       - Normal completion

    Error completion return codes:
        SS$_ACCVIO       - Access violation on parameter
        SS$_BADPARAM     - Bad parameter value specified
        SS$_CLEARED      - Remote association closed the link before
                           it was accepted.
        SS$_ILLIOFUNC    - Invalid function specified
        SS$_INSFARG      - Too few arguments were supplied
        SS$_INSFMEM      - Not enough system resources or process
                           virtual memory available.
        SS$_IVCHAN       - Connection not found or
                           Invalid association handle
        SS$_TOO_MANY_ARGS- Too many arguments were specified
        SS$_WRONGSTATE   - Connection is in wrong state for request
        any failures from routines:
            ALLOCATE_NONPAGED
```

## 5.6. Disconnect: $ICC_DISCONNECT and $ICC_DISCONNECTW

The Disconnect System Service terminates the specified connection. After this service has been called, no further communication is possible over this connection. All outstanding data transmission and reception functions will be terminated with an error before completion is signalled by setting the event flag (if specified) or calling the AST (if specified).

```
$ICC_DISCONNECT             [efn], conn_handle, iosb, [astadr], [astprm],
                            [disc_buf], [disc_len]

    efn             Type:       longword (unsigned)
                    Access:     read only
                    Mechanism:  by value

                    The event flag number to wait on.

    conn_handle     Type:       longword (unsigned)
                    Access:     read only
```

```
                 Mechanism:  by value

                 The ID of the connection to be disconnected.

iosb             Type:       quadword (unsigned)
                 Access:     write only
                 Mechanism:  by 32-bit or 64-bit reference (Alpha)
                             by 32-bit reference (VAX)
                                        +2                      0
                 -------------------------------------------------
                 |       undefined       |   completion status   |
                 -------------------------------------------------
                 |        unused         |        unused         |
                 -------------------------------------------------
                                        +6                     +4
                 I/O status block

                 Successful completion status values:
                     SS$_NORMAL
                 Failure completion status values:
                     SS$_EXQUOTA, SS$_LINKDISCON,
                      and any failures from the routine:
                         $ICC_REJECT


astadr           Type:       procedure_entry_mask
                 Access:     call without stack unwinding
                 Mechanism:  by 32-bit or 64-bit linkage reference
                             (Alpha)
                             by 32-bit reference (VAX)

                 The AST routine to be executed when the operation
                 completes.

astprm           Type:       quadword (unsigned) (Alpha)
                             longword (unsigned) (VAX)
                 Access:     read only
                 Mechanism:  by 64-bit value (Alpha)
                             by 32-bit value (VAX)

                 The parameter to be passed to the AST routine.

disc_buf         Type:       quadword (unsigned) (Alpha)
                             longword (unsigned) (VAX)
                 Access:     read only
                 Mechanism:  by 32-bit or 64-bit reference (Alpha)
                             by 32-bit reference (VAX)

                 A buffer of up to 1024 bytes of disconnect data to
                 be sent to the partner in the connection when
                 notifying it that disconnection is being initiated.

                 Maximum length = 1024 bytes

disc_len         Type:       longword (unsigned)
                 Access:     read only
                 Mechanism:  by value
```

```
                 The number of bytes in disc_buf to actually be sent.


    Condition Values returned       Type:       VMS cond_code
                                    Mechanism:  R0

        Successful completion return codes:
            SS$_NORMAL       - normal successful completion.

        Error completion return codes:
            SS$_ACCVIO       - Access violation on parameter
            SS$_BADPARAM     - Bad parameter value specified
            SS$_DISCONNECT   - The remote association closed the connection
                               before it was accepted or rejected.
            SS$_EXQUOTA      - Not enough AST quota (asynchronous request)
            SS$_INSFARG      - Too few arguments were supplied
            SS$_IVBUFLEN     - The length of the disconnect data buffer
                               is more than 1024 bytes
            SS$_IVCHAN       - Unknown Association or connection specified
                               or invalid association handle
            SS$_LINKDISCON   - An Incoming disconnect event is already
                               in progress
            SS$_TOO_MANY_ARGS- Too many arguments were specified
            SS$_WRONGSTATE   - Connection is in wrong state for request
            And any failures from the routines:
                $ICC_REJECT

$ICC_DISCONNECTW            [efn], conn_handle, iosb, [astadr], [astprm],
                            [disc_buf], [disc_len]
```

$ICC_DISCONNECTW has the same arguments and return codes as $ICC_DISCONNECT.

## 5.7. Transmit Data: $ICC_TRANSMIT and $ICC_TRANSMITW

The Transmit Data System Service sends a single message over a connection. When completion is signalled by setting the efn (if supplied) or calling the AST (if supplied), the buffer has been delivered to the communications system, but not necessarily to the application at the other end of the connection.

```
$ICC_TRANSMIT[W]            [efn], conn_handle, iosb, [astadr], [astprm],
                            send_buf, send_len

    efn             Type:       longword (unsigned)
                    Access:     read only
                    Mechanism:  by value

                    The event flag number to wait on.

    conn_handle     Type:       longword (unsigned)
                    Access:     read only
                    Mechanism:  by value

                    The handle of the fully established (open) connection
                    to send the data over.

    iosb            Type:       quadword (unsigned)
                    Access:     write only
```

```
Mechanism:  by 32-bit or 64-bit reference (Alpha)
            by 32-bit reference (VAX)
                       +2                    0
-------------------------------------------------
|       undefined       |   completion status   |
-------------------------------------------------
|                     unused                    |
-------------------------------------------------
                       +6                   +4
I/O status block

Successful completion status values:
    SS$_NORMAL
Failure completion status values:
    SS$_EXQUOTA, SS$_INSFMEM,
    SS$_LINKABORT, SS$_LINKDISCON

astadr          Type:       procedure_entry_mask
                Access:     call without stack unwinding
                Mechanism:  by 32-bit or 64-bit linkage reference
                            (Alpha)
                            by 32-bit reference (VAX)

                The AST routine to be executed when the operation
                completes.

astprm          Type:       quadword (unsigned) (Alpha)
                            longword (unsigned) (VAX)
                Access:     read only
                Mechanism:  by 64-bit value (Alpha)
                            by 32-bit value (VAX)

                The parameter to be passed to the AST routine.

send_buf        Type:       quadword (unsigned) (Alpha)
                            longword (unsigned) (VAX)
                Access:     read only
                Mechanism:  by 32-bit or 64-bit reference (Alpha)
                            by 32-bit reference (VAX)

                The 32-bit or 64-bit address (on Alpha systems) or
                the 32-bit address (on VAX systems) of the buffer
                containing the data to be sent.  The length of this
                buffer is specified by the argument 'send_len'.

send_len        Type:       longword (unsigned)
                Access:     read only
                Mechanism:  by value

                The length (in bytes) of the data to be sent over
                the connection.  This value specifies the length
                of the buffer 'send_buf'.  IPC will segment buffers
                over 16384 bytes internally.


Condition Values returned       Type:       VMS cond_code
                                Mechanism:  R0
```

```
Successful completion return codes:
    SS$_NORMAL      - Normal completion

Error completion return codes:
    SS$_ACCVIO       - Access violation on parameter
    SS$_BADPARAM     - Bad parameter value specified
    SS$_EXBYTLM      - Insufficient byte count quota
    SS$_WRONGSTATE   - Connection is in wrong state for request
    SS$_INSFARG      - Too few arguments were supplied
    SS$_IVCHAN       - Unknown Association or connection specified
                       or invalid association handle
    SS$_IVMODE       - Attempted to use a connection from a
                       more privileged access mode than the
                       mode in which it was opened
    SS$_LINKDISCON   - An Incoming disconnect event is already
                       in progress
    SS$_TOO_MANY_ARGS- Too many arguments were specified
```

### 5.8. Receive Data: $ICC_RECEIVE and $ICC_RECEIVEW

The Receive Data System Service receives a single message over a connection. If a Request ID is returned at completion, the partner has used a Transceive System Service and requires data to be returned with a Reply service.

```
$ICC_RECEIVE[W]            [efn], conn_handle, iosb, [astadr], [astprm],
                           recv_buf, recv_len, retlen, requestid,
                           reply_len

    efn             Type:       longword (unsigned)
                    Access:     read only
                    Mechanism:  by value

                    Event flag number to wait on

    conn_handle     Type:       longword (unsigned)
                    Access:     read only
                    Mechanism:  by value

                    The handle of the fully established connection.

    iosb            Type:       quadword (unsigned)
                    Access:     write only
                    Mechanism:  by 32-bit or 64-bit reference (Alpha)
                                by 32-bit reference (VAX)
                                          +2                        0
                    -------------------------------------------------
                    |       undefined        |   completion status   |
                    -------------------------------------------------
                    |                      unused                    |
                    -------------------------------------------------
                                          +6                       +4
                    I/O status block

                    Successful completion status values:
                        SS$_NORMAL
                    Failure completion status values (may still need work):
                        SS$_EXQUOTA, SS$_INSFMEM, SS$_LINKDISCON,
```

```
                SS$_BUFOVL

astadr          Type:       procedure_entry_mask
                Access:     call without stack unwinding
                Mechanism:  by 32-bit or 64-bit linkage reference
                            (Alpha)
                            by 32-bit reference (VAX)

                The AST routine to be executed when the operation
                completes.

astprm          Type:       quadword (unsigned) (Alpha)
                            longword (unsigned) (VAX)
                Access:     read only
                Mechanism:  by 64-bit value (Alpha)
                            by 32-bit value (VAX)

                The parameter to be passed to the AST routine.

recv_buf        Type:       quadword (unsigned) (Alpha)
                            longword (unsigned) (VAX)
                Access:     read only
                Mechanism:  by 32-bit or 64-bit reference (Alpha)
                            by 32-bit reference (VAX)

                The 32-bit or 64-bit address (on Alpha systems) or
                the 32-bit address (on VAX systems) of the buffer
                to receive the incoming data.  The length of this
                buffer is specified by the argument 'recv_len'.

recv_len        Type:       longword (unsigned)
                Access:     read only
                Mechanism:  by value

                The length (in bytes) of the buffer available to
                hold the incoming data.  This value specifies the
                length of the buffer 'recv_buf'.  IPC will segment
                buffers over 16384 bytes internally.

retlen          Type:       longword (unsigned)
                Access:     write only
                Mechanism:  by 32-bit or 64-bit reference (Alpha)
                            by 32-bit reference (VAX)

                The 32-bit or 64-bit address (on Alpha systems) or
                the 32-bit address (on VAX systems) of a longword
                into which $ICC_RECEIVE writes the actual length
                of the incoming data message received from the target
                application (in bytes).

requestid       Type:       longword (unsigned)
                Access:     write only
                Mechanism:  by 32-bit or 64-bit reference (Alpha)
                            by 32-bit reference (VAX)

                The 32-bit or 64-bit address (on Alpha systems) or
                the 32-bit address (on VAX systems) of a longword
                into which $ICC_RECEIVE writes the Request/Response
```

```
                  identifier.  This is non-zero if the application is
                  expected to reply to this message.

reply_len         Type:       longword (unsigned)
                  Access:     write only
                  Mechanism:  by 32-bit or 64-bit reference (Alpha)
                              by 32-bit reference (VAX)

                  The 32-bit or 64-bit address (on Alpha systems) or
                  the 32-bit address (on VAX systems) of a longword
                  into which $ICC_RECEIVE writes the length (in
                  bytes) of the expected Reply message (if 'requestid'
                  is non-zero).


Condition Values returned       Type:       VMS cond_code
                                Mechanism:  R0

    Successful completion return codes:
        SS$_NORMAL      - Normal completion

    Error completion return codes:
        SS$_ACCVIO      - Access violation on parameter
        SS$_BADPARAM    - Bad parameter value specified
        SS$_EXBYTLM     - Insufficient byte count quota
        SS$_WRONGSTATE  - Connection is in wrong state for request
        SS$_INSFARG      - Too few arguments were supplied
        SS$_IVCHAN       - Unknown Association or connection specified
                           or invalid association handle
        SS$_IVMODE       - Attempted to use a connection from a
                           more privileged access mode than the
                           mode in which it was opened
        SS$_LINKDISCON   - An Incoming disconnect event is already
                           in progress
        SS$_TOO_MANY_ARGS- Too many arguments were specified
```

### 5.9. Transceive Data: $ICC_TRANSCEIVE and $ICC_TRANSCEIVEW

The Transceive Data System Service sends a single message over a connection and then waits for a reply. When completion is signalled by setting the efn (if supplied) or calling the AST (if supplied), the buffer has been delivered to the application at the other end of the connection and that application has delivered a reply, now present in the reply buffer.

```
$ICC_TRANSCEIVE[W]          [efn], conn_handle, iosb, [astadr], [astprm],
                            send_buf, send_len, reply_buf, reply_len,
                            retlen


efn               Type:       longword (unsigned)
                  Access:     read only
                  Mechanism:  by value

                  The event flag number to wait on.

conn_handle       Type:       longword (unsigned)
                  Access:     read only
                  Mechanism:  by value
```

```
                  The handle of the fully established (open) connection.

iosb              Type:       quadword (unsigned)
                  Access:     write only
                  Mechanism:  by 32-bit or 64-bit reference (Alpha)
                              by 32-bit reference (VAX)
                                         +2                        0
                  -------------------------------------------------
                  |       undefined        |   completion status   |
                  -------------------------------------------------
                  |                     unused                     |
                  -------------------------------------------------
                                         +6                       +4
                  I/O status block


                  Successful completion status values:
                      SS$_NORMAL
                  Failure completion status values:
                      SS$_EXQUOTA, SS$_INSFMEM, SS$_BUFOVFL,
                      SS$_LINKABORT, SS$_LINKDISCON,

astadr            Type:       procedure_entry_mask
                  Access:     call without stack unwinding
                  Mechanism:  by 32-bit or 64-bit linkage reference
                              (Alpha)
                              by 32-bit reference (VAX)

                  The AST routine to be executed when the operation
                  completes.

astprm            Type:       quadword (unsigned) (Alpha)
                              longword (unsigned) (VAX)
                  Access:     read only
                  Mechanism:  by 64-bit value (Alpha)
                              by 32-bit value (VAX)

                  The parameter to be passed to the AST routine.

send_buf          Type:       quadword (unsigned) (Alpha)
                              longword (unsigned) (VAX)
                  Access:     read only
                  Mechanism:  by 32-bit or 64-bit reference (Alpha)
                              by 32-bit reference (VAX)

                  The 32-bit or 64-bit address (on Alpha systems) or
                  the 32-bit address (on VAX systems) of the buffer
                  containing the data to  be sent.  The length of this
                  buffer is specified by the argument 'send_len'.

send_len          Type:       longword (unsigned)
                  Access:     read only
                  Mechanism:  by value

                  The length (in bytes) of the data to be sent over
                  the connection.  This value specifies the length
                  of the buffer 'send_buf'.  IPC will segment buffers
                  over 16384 bytes internally.
```

```
reply_buf       Type:       quadword (unsigned) (Alpha)
                            longword (unsigned) (VAX)
                Access:     read only
                Mechanism:  by 32-bit or 64-bit reference (Alpha)
                            by 32-bit reference (VAX)

                The 32-bit or 64-bit address (on Alpha systems) or
                the 32-bit address (on VAX systems) of the buffer
                available to receive the incoming reply message.
                The length of this buffer is specified by the
                argument 'reply_len'.

reply_len       Type:       longword (unsigned)
                Access:     read only
                Mechanism:  by value

                The length (in bytes) of the buffer to receive the
                reply message.  This value specifies the length of
                of the buffer 'reply_buf'.  IPC will segment buffers
                over 16384 bytes internally.

retlen          Type:       longword (unsigned)
                Access:     write only
                Mechanism:  by 32-bit or 64-bit reference (Alpha)
                            by 32-bit reference (VAX)

                The 32-bit or 64-bit address (on Alpha systems) or
                the 32-bit address (on VAX systems) of a longword
                into which $ICC_TRANSCEIVE writes the actual length
                (in bytes) of the reply data received.  This value
                represents how much of the data pointed to by
                'reply_buf' was returned by the target application.


Condition Values returned       Type:       VMS cond_code
                                Mechanism:  R0

    Successful completion return codes:
        SS$_NORMAL      - Normal completion

    Error completion return codes:
        SS$_ACCVIO      - Access violation on parameter
        SS$_BADPARAM    - Bad parameter value specified
        SS$_EXBYTLM     - Insufficient byte count quota
        SS$_INSFMEM     - Not enough system resources or process
                          virtual memory available.
        SS$_WRONGSTATE  - Connection is in wrong state for request
        SS$_INSFARG      - Too few arguments were supplied
        SS$_IVCHAN       - Unknown Association or connection specified
                           or invalid association handle
        SS$_IVMODE       - Attempted to use a connection from a
                           more privileged access mode than the
                           mode in which it was opened
        SS$_LINKDISCON   - An Incoming disconnect event is already
                           in progress
        SS$_TOO_MANY_ARGS- Too many arguments were specified
```

## 5.10. Reply: $ICC_REPLY and $ICC_REPLYW

The Reply System service is almost identical to the Transmit Data System Service in that it sends a single message over a connection. The only difference is that it is used in response to the reception of a Request ID in a previous Receive Data System Service. When completion is signalled by setting the efn (if supplied) or calling the AST (if supplied), the buffer has been delivered to the communications system, but not necessarily to the application at the other end of the connection.

```
$ICC_REPLY[W]               [efn], conn_handle, iosb, [astadr], [astprm],
                            reply_buf, reply_len, request_handle

    efn             Type:       longword (unsigned)
                    Access:     read only
                    Mechanism:  by value

                    The event flag number to wait on.

    conn_handle     Type:       longword (unsigned)
                    Access:     read only
                    Mechanism:  by value

                    The handle of the fully established connection.

    iosb            Type:       quadword (unsigned)
                    Access:     write only
                    Mechanism:  by 32-bit or 64-bit reference (Alpha)
                                by 32-bit reference (VAX)
                                           +2                       0
                    -------------------------------------------------
                    |       undefined        |   completion status  |
                    -------------------------------------------------
                    |                     unused                    |
                    -------------------------------------------------
                                           +6                      +4
                    I/O status block

                    Successful completion status values:
                        SS$_NORMAL
                    Failure completion status values:
                        SS$_EXQUOTA, SS$_INSFMEM,
                        SS$_LINKABORT, SS$_LINKDISCON

    astadr          Type:       procedure_entry_mask
                    Access:     call without stack unwinding
                    Mechanism:  by 32-bit or 64-bit linkage reference
                                (Alpha)
                                by 32-bit reference (VAX)

                    The AST routine to be executed when the operation
                    completes.

    astprm          Type:       quadword (unsigned) (Alpha)
                                longword (unsigned) (VAX)
                    Access:     read only
                    Mechanism:  by 64-bit value (Alpha)
                                by 32-bit value (VAX)
```

```
                 The parameter to be passed to the AST routine.

reply_buf        Type:        quadword (unsigned) (Alpha)
                              longword (unsigned) (VAX)
                 Access:      read only
                 Mechanism:   by 32-bit or 64-bit reference (Alpha)
                              by 32-bit reference (VAX)

                 The 32-bit or 64-bit address (on Alpha systems) or
                 the 32-bit address (on VAX systems) of the buffer
                 containing the reply data to be sent.  The length of
                 this buffer is specified by the argument 'reply_len'.

reply_len        Type:        longword (unsigned)
                 Access:      read only
                 Mechanism:   by value

                 The length (in bytes) of the reply data to be sent
                 over the connection.  This value specifies the
                 length of the buffer 'reply_buf'.  IPC will segment
                 buffers over 16384 bytes internally.

request_handle   Type:        longword (unsigned)
                 Access:      read only
                 Mechanism:   by value

                 The Request/Response handle from the received
                 Transceive request.


Condition Values returned       Type:        VMS cond_code
                                Mechanism:   R0

    Successful completion return codes:
        SS$_NORMAL      - Normal completion

    Error completion return codes:
        SS$_ACCVIO       - Access violation on parameter
        SS$_BADPARAM     - Bad parameter value specified
        SS$_EXBYTLM      - Insufficient byte count quota
        SS$_WRONGSTATE   - Connection is in wrong state for request
        SS$_INSFARG      - Too few arguments were supplied
        SS$_IVCHAN       - Unknown Association or connection specified
                           or invalid association handle
        SS$_IVMODE       - Attempted to use a connection from a
                           more privileged access mode than the
                           mode in which it was opened
        SS$_LINKDISCON   - An Incoming disconnect event is already
                           in progress
        SS$_NOSUCHID     - The request_handle is invalid
        SS$_TOO_MANY_ARGS- Too many arguments were specified
```

## 6. User Supplied Routines (ASTs)

When opening an association the user may optionally supply a connect/disconnect AST and/or a Data Event AST. These routines will be used for all connections established over the specified association. In addition, for any of the asynchronous services (those provided with both an immediate return and a "W" form), a completion AST may be supplied by the user.

## 6.1. Connect/Disconnect AST

The user choses the name of this routine and supplies the procedure name as an argument to the Open Association service. Five arguments will be passed to the user. The first argument notifies the user whether this is an incoming new connection or a disconnection of an existing connection. The second identifies the connection. The third and fourth provide access to incoming connect or disconnect data (if any) sent by the cooperating application. The fifth argument provides the number of bytes available for any optional Accept or Reject data (in the case of a connect request) or the disconnect reason supplied by the cooperating application (if any).

```
Event_Type      Type:       longword (unsigned)
                Access:     read only
                Mechanism:  by value

                This field will contain a code describing the type of
                event.  The possible event codes are:
                  ICC$C_EV_CONNECT      - Connection event
                  ICC$C_EV_DISCONNECT   - Disconnection event

conn_handle     Type:       longword (unsigned)
                Access:     read only
                Mechanism:  by value

                The handle of the connection associated with the
                event.

Data_Len        Type:       longword (unsigned)
                Access:     read only
                Mechanism:  by value

                The length (in bytes) of the incoming data.  This
                value specifies the length of the buffer 'Data_Bfr',
                and will be between 0 and 1024, with zero indicating
                no data.

Data_Bfr        Type:       quadword (unsigned) (Alpha)
                            longword (unsigned) (VAX)
                Access:     read only
                Mechanism:  by 32-bit or 64-bit value (Alpha)
                            by 32-bit value (VAX)

                The 32-bit or 64-bit address (on Alpha systems) or
                the 32-bit address (on VAX systems) of the P1-space
                buffer containing the data, or zero if no data is
                available.  The length of this buffer is specified
                by the argument 'Data_Len'.

P5              Type:       longword (unsigned)
                Access:     read only
```

```
                    Mechanism:  by value

                    The usage of this argument is dependent on the
                    specified event type code (Event_Type).

                    For connect events (Event_Type=ICC$C_EV_CONNECT),
                    this argument contains the length (in bytes) of
                    the buffer available for a reply.

                    For disconnect events (Event_Type=ICC$C_EV_DISCONNECT),
                    this argument contains the user defined
                    disconnect reason/status from the remote partner.

    P6              Type:       longword (unsigned)
                    Access:     read only
                    Mechanism:  by value

                    The usage of this argument is dependent on the
                    specified event type code (Event_Type).

                    For connect events (Event_Type=ICC$C_EV_CONNECT),
                    this argument is zero.

                    For disconnect events (Event_Type=ICC$C_EV_DISCONNECT),
                    this argument contains the user defined alias supplied
when
                    the connection was opened.
```

### 6.2. Data Event Routine

This routine, if supplied by the user when opening the association, allows the user to be notified of any pending data events over any connections subsequently opened over that assocation. If the user has supplied this routine, the Receive service must only be called in response to incoming data events signalled by this routine, and must be called with a buffer large enough to handle the message size specified. Use of this routine obligates the user to allocate unlimited sized buffers as requested by the cooperating application. The only recovery provided at this time if a sufficiently large buffer cannot be allocated is to disconnect the connection.

```
    Message_Size    Type:       longword (unsigned)
                    Access:     read only
                    Mechanism:  by value

                    This field will contain the number of bytes in the
                    pending data event.

    conn_handle     Type:       longword (unsigned)
                    Access:     read only
                    Mechanism:  by value

                    The handle of the connection associated with the
                    event.

    alias           Type:       longword (unsigned)
                    Access:     read only
                    Mechanism:  by value
```

```
The user defined alias alias supplied when
the connection was opened.
```

### 6.3. Completion ASTs

Completion ASTs may be supplied to the $ICC_CONNECT[W], $ICC_DISCONNECT[W], $ICC_TRANSMIT[W], $ICC_RECEIVE[W], $ICC_TRANSCEIVE[W], and $ICC_REPLY[W] services. In all cases, they are called at the completion of the requested operation, with the single argument, the AST parameter supplied when the original service was called, passed by value. Completion ASTs are not called if the service returns an error prior to initiating the operation.

## 7. The I/O Status Block (IOSB) and Event Flags

On all routines having an IOSB parameter, the IOSB is cleared to zeroes before any other parameter checking is done. As long as the IOSB itself is accessible, this ensures that the IOSB will never contain a success code unless completion has actually occurred. If the service returns an error prior to initiating the operation, the IOSB remains zero.

On all routines having an event flag parameter, the event flag is cleared immediately after clearing the IOSB. If the service returns an error prior to initiating the operation, the event flag is not set.

## 8. The Association Namespace

Opening an Association by the call to $ICC_OPEN_ASSOCIATION makes an entry in a systemwide (not clusterwide) database called the Association Namespace. ICC will provide an enhancement to the management of this namespace over that provided by IPC, which required all users of the IPC service to have the SYSNAM privilege. In addition, ICC will provide a simple clusterwide registry of services created automatically (but optionally) when an association is opened. This clusterwide registry may optionally be used to provide random load balancing between multiple servers within the cluster.

### 8.1. Authorization to create entries in the Association Namespace

The determination of what association name(s) a process may open is based on the following:

- If the process has SYSNAM privilege, any arbitrary association name may be opened.

- If the process does not have SYSNAM privilege, but the holds the rights identifier ICC$NAME_xxx, it may open the association name "xxx" or any association name beginning with "xxx_".

- If the process does not have SYSNAM privilege, but holds the rights identifier ICC$NAME_xxx$, it may open any association name beginning with "xxx$".

- If process does not have SYSNAM privilege and does not hold any ICC$NAME_ Rights Identifier, it will be allowed to open only the default association name ICC$PID_nnnnnnnn.

### 8.2. Creating entries in the clusterwide registry

Using Clusterwide Logical Names, a simple clusterwide registry describing which associations are open on which nodes within a cluster is maintained automatically (but optionally) when $ICC_OPEN_ASSOCIATION and $ICC_CLOSE_ASSOCIATION are called. When opening an association, a pair of optional parameters may be provided which specify a logical name and a clusterwide logical name table. This logical name will be created if necessary and updated to contain a list of which nodes are running this association and the local name of the association, which may or may not be the same throughout the cluster. When the association is closed, the entry, if one was created upon open, is removed from this simple registry. The format of the equivalence name in this simple registry is not currently an interface open to the user. We reserve the right to change it to improve functionality or performance in the future. It is described in the design specification.

### 8.3. Connecting to an association using the simple registry

When calling $ICC_CONNECT, if the argument “remote_node” is zero or not supplied (no descriptor provided at all), the “remote_assoc” argument is interpreted to be a logical name. This logical name is translated in the logical name table specified by the logical name ICC$NAME_TABLE, which may be a search list as allowed by $TRNLNM. $TRNLNM currently requires such search lists to be in either LNM$PROCESS_DIRECTORY or LNM$SYSTEM_DIRECTORY. If the translation is successful a member of the list in the simple registry for this association is chosen at random. No state is maintained by the random number algorithm. Retries are up to the calling application.

## 9. 64-bit support

### 9.1. On Alpha all address arguments are treated as 64-bits

All address arguments supplied by the user are treated as quadword 64-bit addresses, and any descriptors may be 64-bit descriptors. This includes the linkage address provided as procedure values when specifying routine addresses as well as any buffer provided to ICC by the user or to the user by ICC. In the case of P1 addresses provided to the user, which cannot currently be larger than a longword, the quadword interface will still be used for consistency, and the high order longword will be guaranteed to be zero.

For the initial internal release, the underlying IPC does not support 64-bit buffer addresses. Until this is corrected, planned for January 1997, the ICC System Services will check this and reject buffer addresses located in P2 space. This temporary restriction does not exist for handles or linkage addresses located in P2/S2 space.

When 64-bit support is added, it will be supported only through the ICC System Service. Existing users of either IPC kernel mode interface or the IPC user defined system services will neither be affected by the changes nor able to use 64-bit support.

### 9.2. VAX compatibility

All addresses are 32-bits on VAX. Common code may be written in much the same manner as existing code which has a requirement to pass longwords to system services which return 16-bit values. Such code zeroes the longword prior to calling a system service. A similar technique would be to zero the quadword before calling the system service.

On VAX only 32-bit descriptors are defined and supported.

## 10. Access Modes and Image / Process Rundown

### 10.1.Mode of caller

The ICC System Services may be called from any access mode, however, an Association opened in an any mode may only be used to establish connections in that same mode, and a connection opened in any mode may only be used to transmit and receive data in that same mode. Any routines supplied as arguments to any of the ICC System Services will be called in the mode of the caller supplying these routines as arguments.

### 10.2.Image Rundown

At image rundown, any Associations opened in user mode and all Connections associated with any of these Associations will be closed, terminating all operations over them. Associations and Connections opened in inner mode are the responsibility of the inner mode caller, and will not be meddled with at Image Rundown. The inner mode caller is fully responsible for any misbehaviour which might result from address space going away.

### 10.3.Process Rundown

At process rundown, all Associations and Connections regardless of mode are closed.

# 11. Issues

## 11.1.Open Issues

1. Authentication. The existing IPC assumes that all participants are privileged. It provides no information about the identity of the process at the other end of the connection.

    - How do we prevent users of ICC from connecting to existing IPC services and doing bad things? A possible solution is a further restriction on the namespace.

    - Should we provide authentication (maybe only within a cluster) or require the use of some other mechanism such as Kerberos (maybe only when the connection is from outside the cluster).

2. What happens if Close Association is unable to remove the entry from the simple clusterwide association registry (due to some resource problem). Returning an error to the user seems futile. Future users of the registry may attempt to connect to associations no longer there. The same thing may happen when nodes leave the cluster ungracefully.

3. Do we provide SS$_SYNCH as a return code indicating that the operation is already complete and that the AST will not be activated?

## 11.2.Resolved Issues