EGSE Router

A Message Routing Server providing application to application message parsing, client management and location transparency.

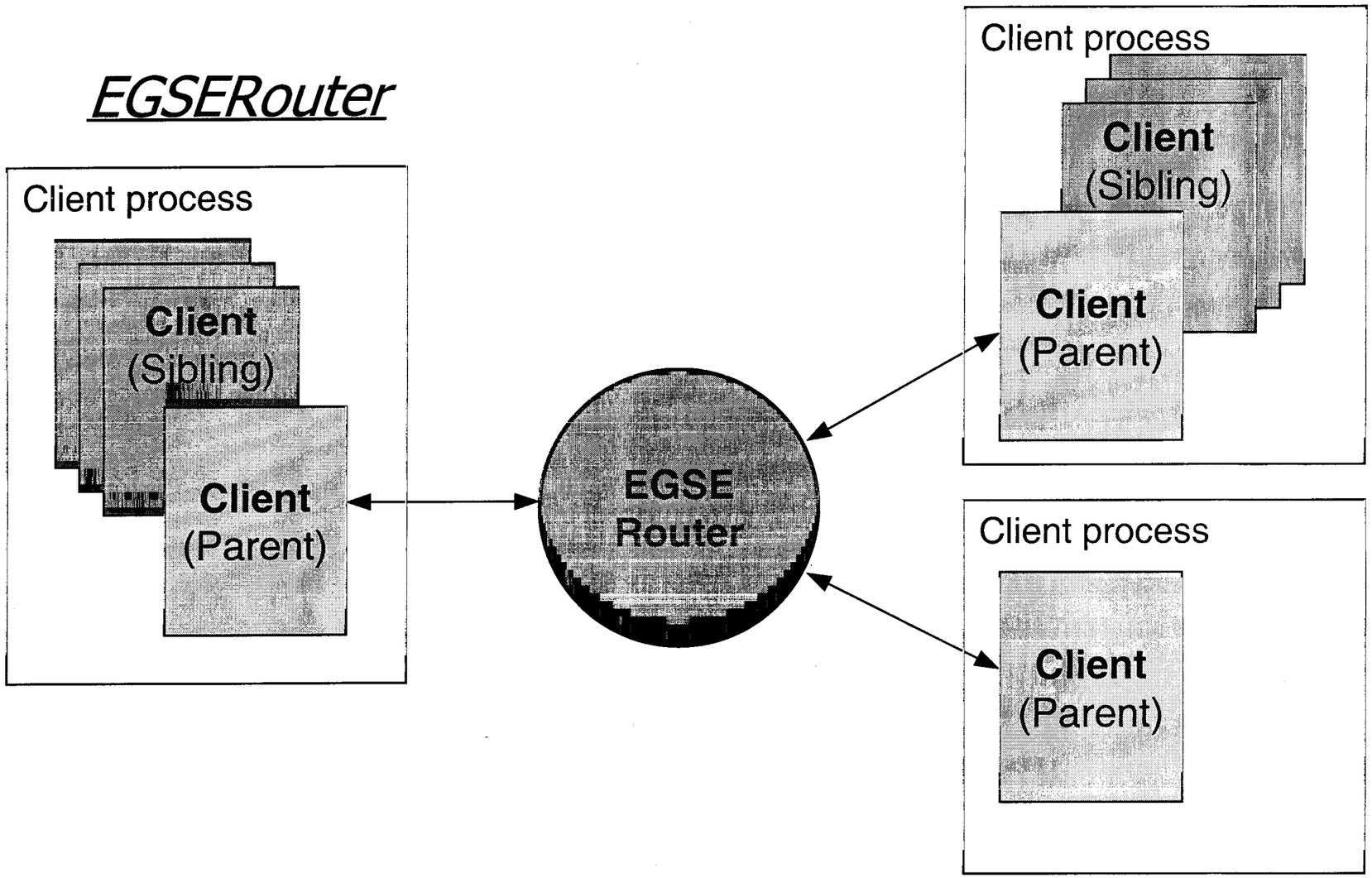
MAIN FEATURES

- Location transparency
- Transparent Client Management
- Multiple Clients per connection
- Data transparency
- Message delivery via:
  - Call-back procedure (asynchronous)
  - Polling for data (synchronous)

IMPLEMENTATION

- Object Oriented
- Using Microsoft Distributed COM
- Out-of-Process/Remote DCOM server
  - ActiveX server
- Muitilanguage support
  - Borland Delphi, C++, LabWiev, Java etc.
    (Any language supporting ActiveX).
- Prepared to use CORBA
  - Corba IDL is available
  - Adds Platform transparency

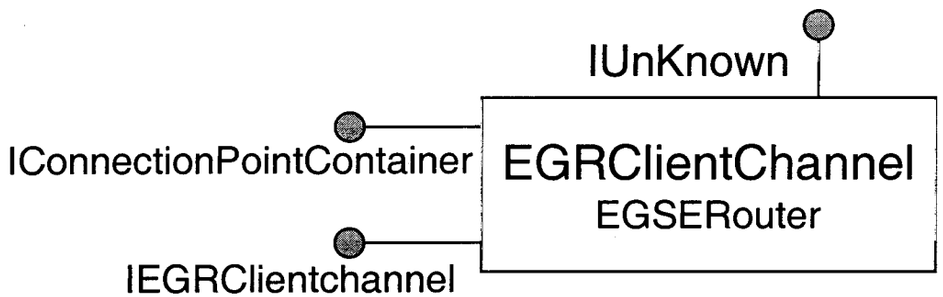
EGSERouter


Client process
Client
(Sibling)
Client
(Parent)
EGSE
Router
Client process
Client
(Sibling)
Client
(Parent)
Client process
Client
(Parent)

EGSERouter Client

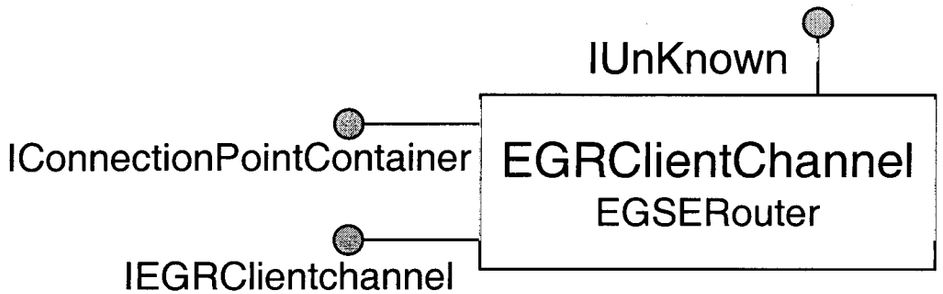
- Clients are Processes.
- Clients are identified by:
  - ClientID (integer) mandatory
  - ClientName (String) optional
    - If ClientName is not explicitly defined by the user, then the system will create a name using the ClientID: #ClientID#.
  - Both the ClientID and ClientName MUST be unique amongst all registered Clients.

CoClass EGRClientChannel



- The *IEGRClientChannel* is the interface through which the Client communicates with the EGSERouter.
- One *Parent Client* must be registered. The Parent Client owns the Channel
- One or more additional *Sibling Clients* may be registered and communicated through the same Channel.

CoClass EGRClientChannel *(continued)*

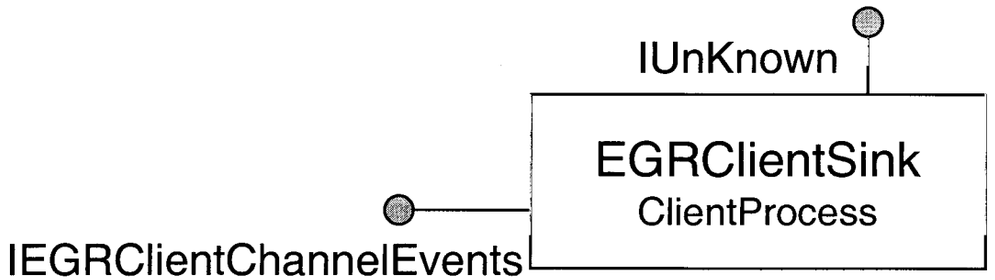


- The *IconnectionPointContainer* is the interface through which the Client connect the Call-back interface IEGRClientChannelEvents with the EGRClientChannel object.

```
FISink: IEGRClientChannelEvents;

FISink:=CoEGRClientSink.Create;
InterfaceConnect(FIChannel,IEGRClientChannelEvents,FISink,FConnections);
```

CoClass EGRClientSink



- The Call-back *IEGRClientChannelEvents* interface must be provided by the Client to the EGSERouter.
  - Messages addressed to the Parent and Sibling Clients are passed through the same Call-back interface.
- Message send to the EGSERouter is immediately passed to a Client which provides the IEGRClientChannelEvents interface.

Supported Data types

- Variant array of Variants
  - Easiest data type for communication between Windows processes.
- Variant array of Bytes
  - Especially included for communication with Unix processes.
  - User is responsible for the format of the application data field (array of bytes). Big endian format.
- EGRMessage Class
  - Support class assisting the user in creating and interpreting Messages.
  - Supports both above Message Data types.
  - Included methods to support easy access to the header fields ansapplication data.

Variant Array of Variants

- Structure:
  - Variant Array of:
    - Destination Client ID (*) VarSmallint
    - Source Client ID (*) VarSmallint
    - Token (*) Varinteger
    - Application data OleVariant

(*) Message Header Fields

NOTE: CURRENTLY NOT SUPPORTED

Variant Array of Bytes

- Structure:
  - Variant array of bytes

Byte offset	0..1	2..3	4..7	8..15	16	17	18..19	20..n
Field	Dest. Client ID	Src. Client ID	Token	Time Stamp	Message type	Spare	Spacecr aft ID	Application Data
	Message Header							

  - All fields MUST be in *Big Endian* format

The Token field

- The *Token* incl. in the Message Header may freely be used by the user to passed data between applications.
- It is generally intended to enable an application to associate a request with the response.
  - The EGSERouter cannot guarantee that the response to a request for data from one application to another will follow directly after the request!
  - Message delivery is chronological but asyncronuous.

The IEGRClientChannel methods

- General:

- All invocations of IEGRChannel methods return a Result value (integer)
  - a successful execution of a method returns the value zero.
  - Any non-zero value is an error code. See TEGRResult.

Note: in the following slides each method will be defined using Borland Pascal notation.

SignOn, SignOf

- Sign On
  - SignOn Parent Client.

function SignOn(ClientID: Word; ClientName: OleVariant): TEGRResult;

Note: ClientName MUST contain a string or an empty string.

- Sign Off
  - Sign Of Parent Client. Sibling registered with the Parent are automatically de-registered.

function SignOff: TEGRResult;

RegSibling, UnRegSibling

- RegSibling
  - Register Sibling Client.

  function RegSibling(ClientID: Word; ClientName: OleVariant): TEGRResult;

  Note: ClientName MUST contain a string or an empty string.

- UnRegSibling
  - De-registered a Sibling. Only Siblings registered thru this channel can be de-registered.

  function UnRegSibling(ClientID: Word): TEGRResult;

SendMsg

- SendMsg
  - Send a Message to a destination client.
  - The message is delivered to the Destination Client
    - directly, if the Destination Client has assigned a call-back procedure;
    - otherwise the message will be placed on the Destination Client's Channel's Message Queue.

```
function SendMsg(Msg: OleVariant): TEGRResult;
```

RecieveMsg

- RecieveMsg
  - Receive the next message from the IEGRChannel message queue.
  - This method is ONLY relevant client using the polling method of receiving messages. I.e. Client who has connected an IEGRClientChannelEvents interface to the IEGRChannel object

```
function RecieveMsg(out Msg: OleVariant): TEGRResult;
```

How to get connected.

- Instantiate the CoClass EGRClientChannel.
  - Delphi

    ```
    FIChannel: IEGRClientChannel; // static or instance var.
    FIChannel:=CoEGRClientChannel.Create;
    ```

    Note: This is automatically performed by the EGRClientStubs
- Sign On (register) Parent Client
  - Delphi

    ```
    RC:=FIChannel.SignOn(ClientID,ClientName);
    ```
- Register Sibling
  - Delphi

    ```
    RC:=FIChannel.RegSibling(ClientID, ClientName);
    ```

RC: =0 success execution of function
#0 Error occurred, RC= error code, see TEGRResult.

PART 2

ESTEC
TOS-GCE

Per Hemso - ESA/ESTEC

EGSEGateway ICD

TOS-GCE/98.015/PSH

Issue 1

Revision 1

10 October 2000

ESTEC
TOS-GCE

Per Hemso - ESA/ESTEC

EGSEGateway ICD

TOS-GCE/98.015/PSH

Issue 1

Revision 1

24 November 1998

CHANGE RECORD SHEET

Date	Issue/ Rev	Page/Para affected	Description	Associated CR No	Approval Authority
24/11/1998	1.0	All pages	Initial issue		
10/10/2000	1.1		Modification due to change of the EGSERouter header		

STATUS SHEET

Page	Status	Page	Status	Page	Status

DOCUMENT APPROVAL

Prepared by	Organisation	Signature	Date
Per S. Hemsoe	ESTEC/TOS-GCE		

Approved by	Organisation	Signature	Date
Serge Valera	ESTEC/TOS-GCE		

DISTRIBUTION LIST

CONTENTS

GLOSSARY OF TERMS

1. Introduction

The purpose of this document is to describe the interface, data messages and protocols, used to exchange of information between components of the SCOS system and the Proba EGSE.
The Proba EGSE components encompass the System SCOE, Unit Test Equipment (Unit SCOEs) and Stimuli Equipment. The SCOS system components (Clients) connect with the EGSE via the EGSEGateway (server) and EGSERouter applications located in the System SCOE PC.

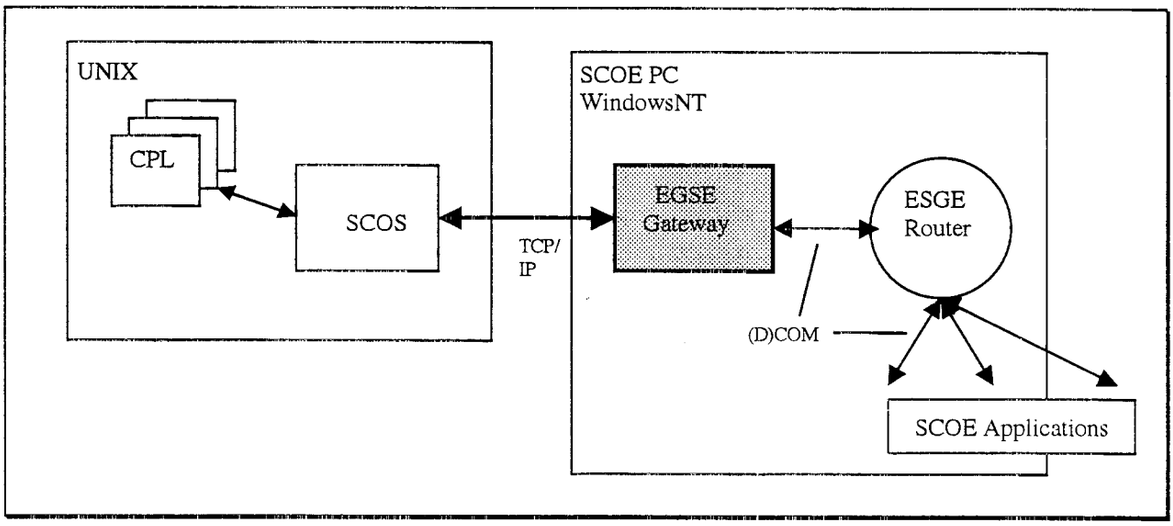


Figure 1 Simplified System Architecture

1.1.1 The EGSEGateway

The EGSEGateway application is on one side the server to which applications (Clients) using communication via TCP/IP sockets with the EGSE Components connects. On the other side the EGSEGateway itself is a client of the EGSERouter. This architecture allows client applications outside the PC Windows NT environment to make use of the services of the EGSERouter Server.

The EGSEGateway is the application implementing the protocol described in this document.

1.1.2 The EGSERouter

The EGSERouter provide services for routing Messages from any application to any application connected with it. The EGSERouter Client interface is implemented using the Microsoft Distributed Common Object model (COM) [1]. The EGSERouter supports full location transparency for its clients within the Windows NT and Windows95/98 environments.
Note: The EGSEGateway expands the services of the EGSERouter to applications using communication via TCP/IP sockets (See *EGSEGateway).*

---

[1] The EGSERouter Client Interface is prepared for using Corba in parallel with DCOM. Currently only DCOM version is released.

2. EGSEGateway protocol

2.1 Message format

A Messages passed between the EGSEGateway and its Client is a byte stream. Fields are send with the MSB first - LSB last ('network order' or big-endian). Ascii strings are passed as null-terminated strings.

The EGSEGateway message expands the EGSERouter Byte Array Message by preceding it with extra header information required to pass information inherently available to a client using the EGSERouter Client Interface procedure calls.

2.1.1 EGSERouter Byte Array Message format

EGSERouter Byte Array Message format has been included to support the communication via TCP/Sockets, especially Unix hosted applications.

Field	Var Name	Type	No octets
Destination ID	DstID	Word	2
Source ID	SrcID	Word	2
Token	Token	Uint	4
Time Stamp	Time	LongInt 64	8
Message Type	Msg_Type	Octet	1
Spare			1
Spacecraft ID	SC_id	Word	2
Application data		Array of byte	0 to 64Kb

Table 1 EGSERouter Byte Array Message format

The field DstID, SrcID and Token together constitute the EGSERouter Message Header.

2.1.2 EGSEGateway Message Format

The EGSEGateway Message consists of a Message header followed by the EGSERouter Byte Array Message.

Field	Var Name	Type	No octets
Message Length	Length	Uint	4
Message ID	MsgID	Octet	1
Result	Result	TEGRResult [2]	4
EGSERouter Array Message		EGSERouter Byte Array Message	See above

Table 2 EGSEGateway Message format

The format described in Table 2 is used for sending messages (data and commands) to and receiving event and data messages from the EGSEGateway. The semantics of each message type and its fields are described in the remainder of this document.

[2] TEGRResult is defined in the EGSERouter.IDL, see appendix A.

2.1.2.1 Command Message

The *Command Messages* contain commands to the EGSEGateway or Client Data to be send to a Destination.

Field	Description	Message Data
Message Length	Number of bytes included in the message excl. the Length field itself.	
Message ID	Identification of the type of message and thereby the contents of the Application Data field. Note: Messages with ID 0 through 4 are the EGSEGateway Commands For details refer to the description of each Message type in the following chapters. 0: Register Client ID/Name (smRegister) 1: Unregister Client (smUnRegister) 2: Send application data (Message) to destination Client (smSendMsg) 3: Request Client ID of named Client (smRequestID) 4: Request Client Name with ID (smRequestName) 5: only applicable to Event Message	 ID, Name Empty User Data Client ID Client Name
Result	Unused	
Message data	EGSERouter Byte Array Message.	

Table 3 Client Output Message Fields

2.1.2.2 Event Message

The *Event Messages* are send to the Client by the EGSEGateway to report error and data events. Event Messages are only send when needed. Event Messages resulting from execution of Command Message are generated by exchanging the SrcID and DstID, leaving the Token unchanged, and inserting the data in the Message Data field when required.

Note: It is the responsibility of the user to ensure correlation between the send Message and the Event by means of a value loaded in the Token field (e.g. the time of sending the Message may be passed in the Token, ms resolution is required).

Field	Description	Event/ Message Data
Message Length	Number of bytes included in the message excl. the Length field itself.	
Message ID	Identification of the type of message and thereby the contents of the Application Data field. Note: Messages with ID 0 through 4 are the EGSEGateway Commands For details refer to the description of each Message type in the following chapters.	
	0: Register Client ID/Name (smRegister)	Yes/ Empty
	1: Unregister Client (smUnRegister)	Yes/ Empty
	2: Send application data (Message) to destination Client (smSendMsg)	No/
	3: Request Client ID of named Client (smRequestID)	Yes/ Client Name
	4: Request Client Name with ID (smRequestName)	Yes/ Client ID
	5: Data Message (smData)	Yes/ Data Send by Source Client
Result	See TEGRResult in the IDL file listing in appendix 1. A value of EGRSuccess indicates successful command execution. The Application Data field the EGSERouter Message contains the Event data, if any. The Application Data field of Event Message type smData contains the data send by the Source Client.	
Message data	EGSERouter Byte Array Message.	

Table 4 EGSEGateway Event Message fields

3. EGSEGateway interface Users Guide.

3.1 Connection/disconnection to/from the EGSEGateway

The EGSEGateway is listening on a TBD socket for incoming connections. It is outside the scope of this document to specify the details of connecting to and disconnecting from the EGSEGateway. The EGSEGateway supports multiple client connections via its Listen socket.
Once connected as many Clients may be registered as required with the constraint that both the Client ID and Client Name MUST be unique. It is the project's responsibility to define a policy, which satisfies this constraint. (If the Client Name is passed as an empty string, then the EGSERouter will create a unique name for its internal use),
Three categories of messages exists depending upon the purpose of the Message:

3.1.1 Command Message

Command Messages are passed by the Client to instruct to the EGSEGateway to perform a specific task. The specifications of the Command Messages are defined in the following chapters.

3.1.1.1 Event Message

Event Messages are passed to the Client whenever the EGSEGateway needs to inform the Client about the result of execution of a command, or when data arrive from a Client. Event Message are generated on a 'need to know basis', e.g. the SendMsg message will only generate an Event Message in case an error in sending the message is detected.

3.1.1.2 Data Event Message

Data Event Message (Result <> EGRSuccess) is used to pass the result of a successful command or when data is available from another Client. The formats of the Data Event Message are defined with the description of each Message type in the following paragraphs.

3.1.1.3 Error Event Message

Error Event Message (Result <> EGRSuccess) is passed when a Command fails execution. A failing command will leave the state of the EGSEGateway and EGSERouter unchanged. (Especially no data has been send if a SendMsg fails). The format of the error Event Message is defined in Table 5.

Field	Value
Length	13
MsgID	MsgID of offending command
Result	TEGRResult [3] <> EGRSuccess
DstID	Client ID of Source of the Message
SrcID	EGSEGateway ClientID ($F000)
Token	User defined
MsgData	empty

Table 5 Error Event Message format

[3] TEGRResult is defined in the EGSERouter.IDL, see appendix A.

4. EGSEGateway Messages

Messages are identified by their MessageID. See Table 6.

MessageID	Command/Message	
0:	Register Client ID/Name	smRegister
1:	Unregister Client	smUnRegister
2:	Send application Data Message to destination Client	smSendMsg
3:	Request Client ID of named Client	smRequestID
4:	Request Client Name with ID	smRequestName
5:	Data Message	smData

Table 6 Message Types

4.1 Register Client ID/Name (smRegister)

Funtion

Before a Client can communicate with the EGSEGateway it must register it's ClientID and ClientName with the EGSEGateway. The ClientID and ClientName MUST be unique amongst all registered Clients. The ClientName may passed as an empty string, in which case the EGSERouter will create a unique name for its internal use.

Data Event

A Data Event message with Result = EGRSuccess is generated to acknowledge a successful registration of the Client.

smRegister Command Message

Field	Value	
Length	13+2+Length(ClientName)	
MsgID	smRegister	
Result	0	
DstID	EGSEGateway ClientID ($F000)	
SrcID	Client ID of Source of the Message	
Token	User defined	
MsgData	ClientID	Word
	ClientName	Null terminated string

smRegister Data Event Message

Field	Value
Length	13
MsgID	smRegister
Result	EGRSuccess
DstID	Client ID of Source of the Message
SrcID	EGSEGateway ClientID ($F000)
Token	User defined
MsgData	empty

4.2 Unregister Client (smUnRegister)

Function
Remove Client with ClientID from the EGSERouter registry.

Data Event
A Data Event message with Result = EGRSuccess is generated to acknowledge a successful de-registration of the Client.

smUnRegister Command Message

Field	Value
Length	13
MsgID	smUnRegister
Result	0
DstID	EGSEGateway ClientID ($F000)
SrcID	Client ID of Source of the Message
Token	User defined
MsgData	ClientID

smUnRegister Data Event Message

Field	Value
Length	13
MsgID	smUnRegister
Result	EGRSuccess
DstID	Client ID of Source of the Message
SrcID	EGSEGateway ClientID ($F000)
Token	User defined
MsgData	Empty

4.3 Send application Data Message to destination Client (smSendMsg)

Function

The Message will be send to the Client with DstID. This message will cause a smData event to be generated in the context of the Destination Client.

Data Event

None

smSendMsg Command Message

Field	Value
Length	13 Length(UserData)
MsgID	smSendMsg
Result	0
DstID	Client ID of Destination of the Message
SrcID	Client ID of Source of the Message
Token	User defined
MsgData	UserData Array of bytes

smSendMsg Data Event Message

None. Generated in the context of the Destination Client, see Data Message (smData).

4.4 Request Client ID of named Client (smRequestID)

Function

This message type allow a Client to obtain the ClientID of another Client when the Name is known if the other Client. It is only possible to request the ClientID of Clients currently registered.

Data Event

MsgData contains the ClientID of the requested Client.

smRequestID Command Message

Field	Value
Length	13+ Length(ClientName)
MsgID	smRequestID
Result	0
DstID	EGSEGateway ClientID ($F000)
SrcID	Client ID of Source of the Message
Token	User defined
MsgData	ClientName Null terminated string

smRequestID Data Event Message

Field	Value
Length	13+2
MsgID	smRequestID
Result	EGRSuccess
DstID	Client ID of Source of the Message
SrcID	EGSEGateway ClientID ($F000)
Token	User defined
MsgData	ClientID Word

4.5 Request Client Name with ID (smRequestName)

Function

This message type allows a Client to obtain the ClientName of another Client when the ID is known of the other Client. It is only possible to request the ClientName of Clients currently registered.

Note: This function is not yet implemented.

Data Event

MsgData contains the ClientName of the requested Client.

smRegister Command Message

Field	Value
Length	13+2
MsgID	smRequestName
Result	0
DstID	EGSEGateway ClientID ($F000)
SrcID	Client ID of Source of the Message
Token	User defined
MsgData	ClientID Word

smRegister Data Event Message

Field	Value
Length	13++ Length(ClientName)
MsgID	smRequestName
Result	EGRSuccess
DstID	Client ID of Source of the Message
SrcID	EGSEGateway ClientID ($F000)
Token	User defined
MsgData	ClientName Null terminated string

4.6 Data Message (smData)

Function

This Data Message is send by the EGSEGateway whenever a Client performs a SendMsg command with DstID being one of the Client registered via the correction.

Data Event

MsgData contains the application data.

smData Command Message

Not applicable, See Send application Data Message to destination Client (smSendMsg).

smData Data Event Message

Field	Value
Length	13+ Length(UserData)
MsgID	smData
Result	EGRSuccess
DstID	Client ID of Destination of the Message
SrcID	Client ID of Source of the Message
Token	User defined
MsgData	UserData Array of bytes

APPENDIX A EGSEROUTER.IDL

The listing included here is the DCOM IDL file of the EGSERouter application.

Obtain current list from the software team.

EGSE ROUTER - trial version

This CD contains the ESA EGSERouter, EGSEGateway, EGSETrafficSPY and sample clients.

NOTE: All code is written in Borland Delphi v 5.0 pascal.

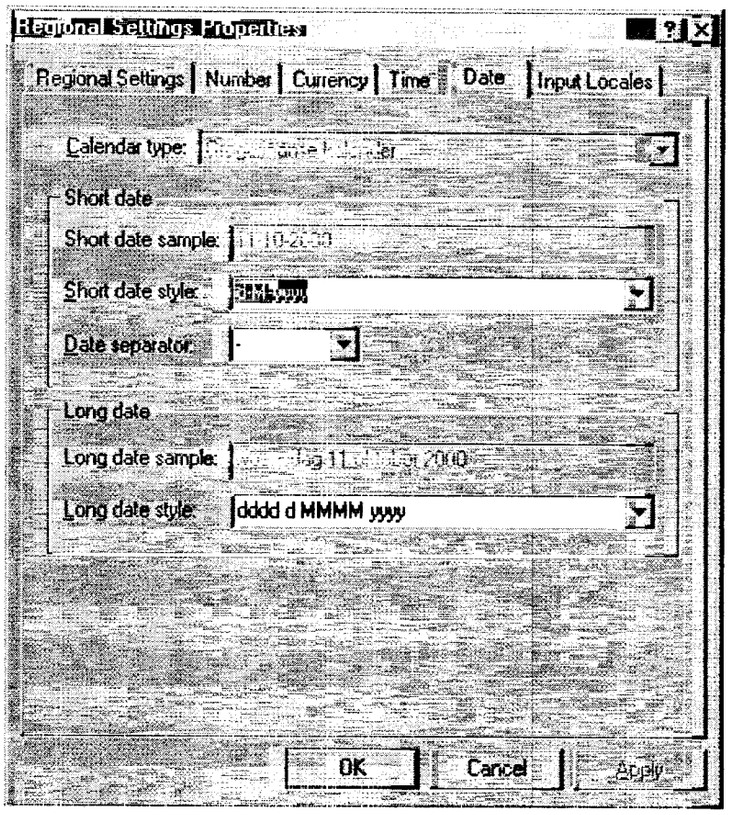
System Requirements:
==========

Windows NT version 6 Service Pack 6 or
Windows 98.

The EGSERouter makes use of the Microsoft DCOM.

Pre-installation:
==========

Prior to installation you must ensure that the Date properties of Regional Settings to European standard as shown below.



Installation:
==========
Installation is performed by InstallShield. Open the Install directory and run Setup.exe from the Install directory.

Executables and Sample sources will be installed in the target directory nominated during the setup procedure. The contents of this directories are listed below.

EgseRouter.exe

EGSERouter Server. It is not necessary to execute the EGSERouter explicitly. It will start automatically whenever a Client requires it's services.

EgseGateway.exe

Gateway allowing TCP/IP Clients to communicate with applications using the EGSERouter services

EgseTrafficSPY.exe

The EgseTrafficSPY will display messages send to the EGSERouter. The monitor must be started manually.
WARNING: The EgseTrafficSPY will impact performance of the EGSERouter When Enabled.

CallBackClient.exe

Sample Client making use of the EGSERouter's capability to perform a Callback to the client when data is available.
Note: - Signing On more than one client will cause the second and subsequent client to be registered as Siblings.
- Both the ClientID and ClientName MUST be unique amongst all Clients of the EGSERouter.
- Monitor the field RESULT. If this is NOT 0 after executing a command (SignOn, SignOff, Send or Read) then the command failed. See EGRResult.

PollingClient.exe

Sample Client using the polling method to obtain messages. The field next to the READ button indicates the no. of message in the clients input queue. Depress the READ button to read the messages.

```
Installation directory contents:
================================
Directory PATH listing
Volume serial number is 0012FC94 A02C:3702
C:\PROGRAM FILES\EGSEROUTER
|   DeIsL1.isu
|   _DEISREG.ISR
|   _ISREG32.DLL
|
+---bin
|       CallBackClient.exe
|       EgseGateway.exe
|       EgseRouter.exe
|       EgseTraficSPY.exe
|       PollingClient.exe
|
\---Samples
    +---Lib
    |       EGRBuffer.pas
    |       EGRClientStubs.pas
    |       EGRGlobals.pas
    |       EGRMessage.pas
    |       EgseClientStub.pas
    |       EgseRouter.tlb
    |       EGSERouter_TLB.dcr
    |       EGSERouter_TLB.pas
    |       E_Time.pas
    |       HexMemo.pas
    |       MTAThread.pas
    |       VerInfo.pas
    |
    \---SampleClients
        |   SampleClients.bpg
        |   SampleClients.dsk
        |
        +---CallBackClient
        |       CallBackClient.cfg
        |       CallBackClient.dof
        |       CallBackClient.dpr
        |       CallBackClient.dsk
        |       CallBackClient.res
        |       CallBacklClientMain.dfm
        |       CallBacklClientMain.pas
        |
        +---EgseTrafficSPY
        |       EgseTrafficSPY.cfg
        |       EgseTrafficSPY.dof
        |       EgseTrafficSPY.dpr
        |       EgseTrafficSPY.drc
        |       EgseTrafficSPY.res
        |       EgseTrafficSpyMain.dcu
        |       EgseTrafficSpyMain.dfm
        |       EgseTrafficSpyMain.pas
        |       TrafficSpyMain.dfm
        |
```

```
+---PollingClient
|       PollingClient.cfg
|       PollingClient.dof
|       PollingClient.dpr
|       PollingClient.dsk
|       PollingClient.res
|       PollingClientMain.dfm
|       PollingClientMain.pas
|
\---TestRouterStub
    |   main.dfm
    |   main.pas
    |   TestRouterStubDll.cfg
    |   TestRouterStubDll.dof
    |   TestRouterStubDll.dpr
    |   TestRouterStubDll.res
    |
    \---LabView
            GatewayClient.vi
            HEX dump.vi
            WriteToGateway.vi
```