API Data Descriptions for the C Wrapper

This chapter describes the client API data structures available for the C Wrapper and covers the following topics:

- Conventions Used for API Data Descriptions
- API Data Descriptions

Conventions Used for API Data Descriptions

The following naming conventions are used to describe the EntireX RPC API data structures:

Naming Convention	Data Type
`ul`*XXXXX*	unsigned long
`us`*XXXXX*	unsigned short
`u`*XXXXX*	unsigned ... (predefined types such as `ptrdiff_t`)
`sz`*XXXXX*	zero terminated string
`p`*XXXXX*	pointer to ...

API Data Descriptions

ERX_CLIENT_IDENTIFICATION

```
typedef  struct                tagERX_CLIENT_IDENTIFICATION
{
   char           szUserId         [ ERX_MAX_USERID_LENGTH + 1 ];
   char           szPassword       [ ERX_MAX_PASSWORD_LENGTH + 1 ];
   char           szToken          [ ERX_MAX_TOKEN_LENGTH + 1 ];
   char           szSecurityToken  [ ERX_MAX_securityToken_LENGTH + 1 ];
   char           szNewPassword    [ ERX_MAX_PASSWORD_LENGTH       + 1 ];
   char           szRpcUserId      [ ERX_MAX_USERID_LENGTH + 1 ];
   char           szRpcPassword    [ ERX_MAX_PASSWORD_LENGTH + 1 ];
   char           cForceLogon;
   unsigned char  uEncryptionLevel;
   char           *pSSLParameter;
   unsigned char  uCompressionLevel;
}  ERX_CLIENT_IDENTIFICATION;
```

Field	Description
szUserId	Mandatory. EntireX Broker user identification. Corresponds to the USER-ID field of the ACI control block. Also used by EntireX Security.
szPassword	Used when EntireX Security is used. Corresponds to the PASSWORD field of the ACI control block.
szNewPassword	Used when EntireX Security is used. Corresponds to the NEWPASSWORD field of the ACI control block.
szToken	Optional. Token used by the EntireX Broker to identify the caller. See TOKEN field of the ACI control block.
szSecurityToken	Security token generated by EntireX Security and EntireX Broker after successful security validation. Do not overwrite or change it. Corresponds to the SECURITY-TOKEN field of the ACI control block.
szRpcUserId	Optional. Additional user identification for EntireX/Natural RPC server. Used by Natural Security, for example. If this field is empty, szUserId is used.
szRpcPassword	Optional. Additional password for EntireX/Natural RPC server. Used by Natural Security, for example. If this field is empty, szPassword is used.
cForceLogon	Mandatory. Determines whether explicit logon or auto-logon is used by the caller. Corresponds to the FORCE-LOGON field of the ACI control block.
uEncryptionLevel	Optional Encryption level used by EntireX Security. Corresponds to the ENCRYPTION-LEVEL field of the ACI control block.
*pSSLParameter	Secure Sockets Layer Settings are provided here as a null-terminated string. See *Running Broker with SSL or TLS Transport* under z/OS | UNIX | Windows for more information.
uCompressionLevel	Optional. Corresponds to the COMPRESSLEVEL field of the ACI control block.

ERX_SERVER_ADDRESS

```
typedef  struct              tagERX_SERVER_ADDRESS
{
   ERXeMedium Medium;
   unsigned long             ulTimeOut;
   union
   {

     ERX_SA_BROKER           BROKER;
     ERX_SA_BROKER_LIBRARY   BROKER_Library;
     ERX_SA_CONNECTION       Connection;

   }  Address;
}  ERX_SERVER_ADDRESS;
```

Field	Description
Medium	Mandatory. Selects an RPC server. The following types are supported: ERX_TM_BROKER (for backward compatibility) ERX_TM_BROKER_LIBRARY ERX_TM_CONNECTION This type of medium is used for the connection-oriented (conversational) RPC. After successful ERXConnect (that is, after opening the conversation), the RPCs are invoked using ERX_TM_CONNECTION. Any open conversation must be closed with ERXDisconnectCommit or aborted with ERXDisconnect.
ulTimeOut	Mandatory. Gives the timeout value for the transport system in seconds. Corresponds to the WAIT field of the ACI control block. **Note:** Zero is not a valid value.

Address

Depending on the Medium field, the Address union holds the necessary information to address a server:

BROKER	
szEtbidName	Mandatory. Broker ID used. Corresponds to the BROKER-ID field of the ACI control block.
szClassName	Mandatory. Class Name of the EntireX/Natural RPC server. Use RPC for Natural RPC Server. Corresponds to the SERVER-CLASS field of the ACI control block.
szServerName	Mandatory. Server Name of the EntireX/Natural RPC server. Corresponds to the SERVER-NAME field of the ACI control block.
szServiceName	Mandatory. Service Name of the EntireX/Natural RPC server. Use CALLNAT for Natural RPC Server. Corresponds to the SERVICE field of the ACI control block.
BROKER_LIBRARY	
szEtbidName	Mandatory. Broker ID used. Corresponds to the BROKER-ID field of the ACI control block.
szClassName	Mandatory. Class Name of the EntireX/Natural RPC server. Use RPC for Natural RPC Server. Corresponds to the SERVER-CLASS field of the ACI control block.
szServerName	Mandatory. Server Name of the EntireX/Natural RPC server. Corresponds to the SERVER-NAME field of the ACI control block.

BROKER	
`szServiceName`	Mandatory. Service Name of the EntireX/Natural RPC server. Use `CALLNAT` for Natural RPC Server. Corresponds to the `SERVICE` field of the ACI control block.
`szLibraryName`	Mandatory. Library sent to the EntireX/Natural RPC server by the client. The library specified here overrides any library information specified in the IDL file, see `library-definition`.
`cCompression`	Mandatory. Switches on compression to transfer EntireX RPC Requests. Valid values: • *blank* - No Compression. Use `ERX_COMPRESSION_NO` to disable compression. • 2 - Transfer `IN` fields from client to server; `OUT` fields from server to client; `INOUT` fields in both directions. Use `ERX_COMPRESSION_YES` to specify compression.
`cNaturalLogon`	• Y - If the server is a Natural RPC Server, a Natural logon will be performed to the library specified above. This flag also causes the `szRpcUserId` and `szRpcPassword` fields to be transferred in encrypted form to the Natural RPC Server and is used by Natural Security. Use `ERX_NATURAL LOGON_YES` to enable a logon to Natural. • If the server is an EntireX RPC server under z/OS (batch), the serving task will be authenticated with the user ID/password supplied with `szUserId`/`szPassword` of `ERX_CLIENT_IDENTIFICATION` structure above. • If the server is an EntireX RPC server under z/OS (CICS), a new serving task will be created (`EXEC CICS START TRAN(esrv) USERID(szUserId)`) to work with the userid which is passed in `szClient` field of `ERX_CLIENT_IDENTIFICATION` structure above. • N - No Natural Logon processing is executed. Use `ERX_NATURAL_LOGON_NO` to disable Natural Logon processing.
Connection	
Contains internal information for EntireX runtime. **Warning:** **Do not modify**	

ERX_CALL_IDENTIFICATION

```
typedef     struct         tagERX_CALL_IDENTIFICATION
{
```

```
   char     szLibraryName  [ ERX_MAX_LIBRARY_NAME_LENGTH + 1 ];
   char     szProgramName  [ ERX_MAX_PROGRAM_NAME_LENGTH + 1 ];
   unsigned long           ulVersion;
}  ERX_CALL_IDENTIFICATION;
```

Field	Description
szLibraryName	The name of the library where the program to be called resides. The format depends on the environment. For a mainframe Natural server, for example, the name of the library must be uppercase.
szProgramName	The name of the program to be called. The format depends on the environment. For a mainframe Natural server, for example, the name of the program must be uppercase.
ulVersion	Reserved for future use, should be set to zero.

ERX_PARAMETER_DEFINITION_V3

```
typedef                    struct             tagERX_PARAMETER_DEFINITION
{
   char                    szParameterName    [ ERX_MAX_PARAMETER_NAME_LENGTH + 1 ];
   ERXeTypeCode            usType;
   ERXeAttributes          usAttributes;
   size_t                  uElementLength;
   ERX_OBJECT_SIZE         uSize
   unsigned long           uParent;
   unsigned long           uOccurrence        [ ERX_MAX_INDICES ];
   ERX_POINTER_DIFFERENCE  uBase;
   ERX_POINTER_DIFFERENCE  uDelta             [ ERX_MAX_INDICES ];
   void                    *pCallInfoBlock;
}  ERX_PARAMETER_DEFINITION_V3;
```

Field	Description
szParameterName	The name of the parameter.

Field	Description
usType	The data type including parameter type direction and index count. See *IDL Data Types*. Bits: FEDCBA9876543210 UUUUUUTTTTTTDDNN U: Unused T: Type Code ERX_TYPE_A, STRING FIXED LENGTH ERX_TYPE_AV, STRING VARIABLE LENGTH ERX_TYPE_K, KANJI FIXED LENGTH ERX_TYPE_KV, KANJI VARIABLE LENGTH ERX_TYPE_L, LOGICAL ERX_TYPE_I, INTEGER ERX_TYPE_N, UNPACKED DECIMAL ERX_TYPE_NU, UNPACKED DECIMAL UNSIGNED ERX_TYPE_P, PACKED DECIMAL ERX_TYPE_PU, PACKED DECIMAL UNSIGNED ERX_TYPE_F, FLOAT ERX_TYPE_T, DATE & TIME ERX_TYPE_D, DATE ERX_TYPE_B, BINARY FIXED LENGTH ERX_TYPE_BV, BINARY VARIABLE LENGTH ERX_TYPE_G, GROUP ERX_TYPE_S, STRUCTURES ERX_TYPE_U, UNICODE FIXED LENGTH ERX_TYPE_UV, UNICODE VARIABLE LENGTH D: Parameter Direction ERX_IN_PARM ERX_OUT_PARM ERX_INOUT_PARM N: Index Count (0 .. 3) All other bits must be zero.
usAttributes	Attributes can be combined by OR when they are not exclusive, such as ERX_ATTR_STRING and ERX_ATTR_MF_ALPHA. For a list of Attributes see the following table.
uElementLength	Information on the logical length of the parameter. For ERX_TYPE_A, for example, A10 in the IDL file has a uElementLength of 11 when mapped to string with ERX_ATTR_STRING. It has a uElementLength of 10 when mapped with ERX_ATTR_MF_ALPHA.
uSize	The physical size of the parameter in bytes.
uOccurrence	The count of elements in each dimension in ascending order. For unbounded arrays: when set presents a possible maximum. Unbounded arrays with value zero have no maximum.
uBase	The address of the base of the parameter.

Field	Description
uDelta	The difference for each dimension between following elements in ascending order.
uCallInfoBlock	Pointer to CallInfoBlock for structures ERX_TYPE_S.

Attributes

Attribute	Description
ERX_ATTR_ARRAY_V1	First dimension of array is unbounded. Attribute evaluated by Software AG IDL Compiler.
ERX_ATTR_ARRAY_V2	Second dimension of array is unbounded. Attribute evaluated by Software AG IDL Compiler.
ERX_ATTR_ARRAY_V3	Third dimension of array is unbounded. Attribute evaluated by Software AG IDL Compiler.
ERX_ATTR_ALIGNED	The parameter is aligned on the server side. The attribute is evaluated by the Software AG IDL Compiler. Used by EntireX RPC server on CICS.
ERX_ATTR_DOUBLE	The parameter is mapped to C data type double. Valid for: • ERX_TYPE_P • ERX_TYPE_PU • ERX_TYPE_N • ERX_TYPE_NU The mapping can be forced by a template by adding 32768 to the %TypeAttribute macro of the Software AG IDL Compiler.
ERX_ATTR_PACKED	The parameter is mapped to C data type char[...] contained in IBM mainframe packed format. Valid for: • ERX_TYPE_P • ERX_TYPE_PU Default mapping for P and PU data types; used when nothing is added to the %TypeAttribute.
ERX_ATTR_UNPACKED	The parameter is mapped to C data type char[...] contained in IBM mainframe packed format. Valid for: • ERX_TYPE_N • ERX_TYPE_NU Default mapping for N and NU data types; used when nothing is added to the %TypeAttribute.

Attribute	Description
ERX_ATTR_STRING	The parameter is mapped to a null terminated string. Valid for • ERX_TYPE_A The mapping can be forced by a template by adding 16384 to the %TypeAttribute macro of the Software AG IDL Compiler.
ERX_ATTR_MF_ALPHA	The parameter is mapped to C data type char[...] - but without a NULL terminator. Valid for: • ERX_TYPE_A Default attribute for A data types; is used when nothing is added to the %TypeAttribute.
ERX_ATTR_NOTHING	Use when none of the mappings described above apply.

ERX_CALL_INFORMATION_BLOCK

```
typedef  struct                         tagERX_CALL_INFORMATION_BLOCK
{
   ERX_CALL_IDENTIFICATION              Callee;
   unsigned short                       uParameterCount;
   ERX_PARAMETER_DEFINITION_V3          *pParmDef;
}  ERX_CALL_INFORMATION_BLOCK;
```

Field	Description
Callee	The identification of the program to be called, see ERX_CALL_IDENTIFICATION
uParameterCount	The total count of the parameters (the number of entries in the parameter definition array).
pParmDef	A pointer to the parameter definition array. See ERX_PARAMETER_DEFINITION_V3_V3

ERX_ERROR_INFORMATION

```
typedef  struct          tagERX_ERROR_INFO
{
   ERXeReturnCode        rc;
   char                  szMessage[ 256 ];
}  ERX_ERROR_INFO;
```

Field	Description
rc	The last return code returned. See *Error Messages and Codes*.
szMessage	Error message. Text associated with the last return code issued.

ERX_IS_SERVING

```
typedef struct          tagERX_IS_SERVING
{
   char                 *pMessage;
   int                  uMessageLength;
}  ERX_IS_SERVING;
```

Field	Description
pMessage	Pointer to the provided buffer for server’s "alive" message.
uMessageLength	Length of the provided buffer.

ERX_TERMINATE_SERVER

```
typedef  enum
{
   ERX_SHUTDOWN_IMMED_ALL = 1,
   ERX_SHUTDOWN_ANYONE = 2
}  ERXeShutdownCommand;

typedef struct tagERX_TERMINATE_SERVER
{
   ERXeShutdownCommand       eShutdownCommand;
   char                      *pMessage;
   int                       uMessageLength;
}  ERX_TERMINATE_SERVER;
```

Field	Description
eShutdownCommand	The shutdown method to be used. • **ERX_SHUTDOWN_IMMED_ALL** Using EntireX Broker Command Service. Using this method forces all instances of RPC servers (Java, Natural, UNIX, Windows, CICS, etc.) to be shut down. • **ERX_SHUTDOWN_ANYONE** Directly to EntireX RPC Server. If multiple instances of RPC servers (Java, Natural, UNIX, Windows, CICS, etc.) are registered under the same class/server/service name at the broker, only one single RPC server instance is shut down. There is no control over which instance is shut down. To shut down all RPC server instances, you have to repeat the function until no more RPC server instances are registered. This method is compatible with the method from versions of EntireX RPC prior to EntireX 5.3.1.
pMessage	Pointer to the provided buffer for server’s "completion" message.
uMessageLength	Length of the provided buffer.

ERX_CONTEXT_BLOCK

```
typedef struct tagERX_CONTEXT_BLOCK
{
   ERXCallId                   ERXCallId;
   ERXeReturnCode              ERXrc;
   ERX_ERROR_INFO              ERXErrorInfo;
   ERX_SERVER_ADDRESS          ERXServer;
   ERX_CLIENT_IDENTIFICATION   ERXClient;
}  ERX_CONTEXT_BLOCK;
```

Field	Description
ERXCallId	The CallId returned by a caller.
ERXrc	EntireX RPC Error Code. See *Error Messages and Codes*.
ERXErrorInfo	EntireX RPC Error information, see ERX_ERROR_INFORMATION.
ERXServer	The server address, see ERX_SERVER_ADDRESS.
ERXClient	The client identification, see ERX_CLIENT_IDENTIFICATION.

ERX_SVM_VERSION_1

```
#define ERX_SVM_VERSION_1 (unsigned long) 1
typedef struct tagERX_SVM_V1
{
   unsigned long  version;
   char          *pProtocol;
   char          *pSM;
   char          *pFA;
   char          *pVA;
   char          * pSA;
}  ERX_SVM_V1;
```

Field	Description
version	Version of this control block. Initialitze with ERX_SVM_VERSION_1.
pProtocol	RPC protocol version from the related server mapping file (EntireX Workbench file with extension .cvm) evaluated by the Software AG IDL Compiler and provided in the output_substitution_sequence %SVMRpcProtocol.
pSM	Pointer to the meta data part of the related server mapping file evaluated by the IDL Compiler and provided in the output_substitution_sequence %SVMMetaData.
pFA	Pointer to the format area of the related server mapping file evaluated by the IDL Compiler and provided in the output_substitution_sequence %SVMFormatArea.
pSA	Pointer to the string area of the related server mapping file evaluated by the IDL Compiler and provided in the output_substitution_sequence %SVMStringArea.