2 IDL File Installation

This chapter includes information for the following topics:

Installing IDL Files

The Interface Definition Language (IDL) files required for running ezTroubleAdmin are copied to your `ezTA/idl` directory during installation. These files describe ezTroubleAdmin attributes and operations (see Table 2-1).

The Service Provider OSS must implement a portion of this IDL to communicate with the TA Manager.

Table 2-1: ezTroubleAdmin Interface Definition Language (IDL) Files

File	Description
ASSOCOPS.IDL	IDL operations used to perform CMIP Association, Release, and Abort commands, as well as operations used to set key IDs accessed by the TPI.
ASSOCSTATS.IDL	IDL operations used to obtain status and statistics information from CMIP messages transmitted by the TPI.
BASE.IDL	Base data types and exceptions used in ezTroubleAdmin.
CMIPAGENT.IDL	IDL operations for sending PT_CREATE, PT_GET and PT_SET from the TA Manager. This is used for internal communication between the TA Manager and Gateway and not required for CLEC OSS-to-TA Manager communication. Hence this file is not described further.
CMIPMANAGER.IDL	IDL operations for the responses, event reports, and errors received from the TA Gateway. This is used for internal communication between the TA Manager and Gateway and not required for CLEC OSS-to-TA Manager communication. Hence this file is not described further.
CMIPTYPES.IDL	IDL data structures that hold information about CMIP message statistics and operations.
MGRTOOSS.IDL	IDL operations for receiving responses, event reports, and errors from the TA Manager. The TA Manager implements this as a CORBA client. The CLEC OSS should implement the CORBA server for these functions. The IDL operations are described in detail later.

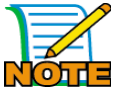
Table 2-1: ezTroubleAdmin Interface Definition Language (IDL) Files

File	Description
OSSTOMGR.IDL	IDL operations for creating, adding/modifying, getting status, providing response for event reports, and canceling trouble tickets. The TA Manager implements this as a CORBA server. The CLEC OSS should implement the CORBA client for this purpose. The IDL operations are described in detail later.
TAATTRIBUTES.IDL	An enumeration that provides a unique ID to each of the attributes. All IDL operations in TA pass a list of attributes back and forth. Each attribute consists of an ID and a CORBA Any type. Hence the attributes from the TATYPES.IDL have to be converted back and forth from CORBA Any. The attribute ID can be used to identify the type of the attribute contained within the CORBA Any.
TATYPES.IDL	All ezTroubleAdmin attribute types derived from the T1.227 and the T1.228 standards and described in the data catalog section.
TIMETYPES.IDL	Data types which hold date/time.

Implementing CORBA

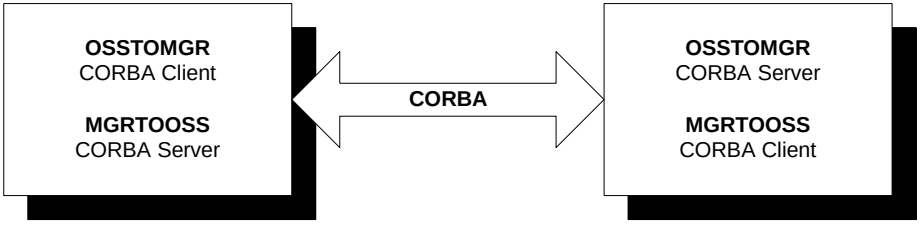
The developers of ezTroubleAdmin 3.0.1 used IONA Orbix and OrbixWeb during the underlying CORBA implementation, but you can choose any Object Request Broker (ORB) for your OSS.

As shown in Figure 2-1, the communication between the OSS and the TA Manager is 2-way. Therefore, the IDL implementation of the OSS requires a CORBA client for the `FromOSSToManager` interface in the `OSSTOMGR.IDL`. It also requires a CORBA server for the `FromMagangerToOSS` interface in the `MGRTOOSS.IDL`.



After creating this implementation, you must update the appropriate configuration parameters in the `TA_MGR_CONFIG` file to point to your OSS.

Figure 2-1: ezTroubleAdmin-CORBA Implementation



Using IORs

The use of Interoperable Object References (IORs) requires certain parameters. The following table provides a list of these parameters and, where applicable, the new values required for your installation.

Table 2-2: Parameters affecting IORs

Parameter	Description	Old Value	New Value
OSS_IDS	OSS ID or account name from the MOI. Separate multiple values with semi-colons.	TEST_OSS	Your assigned OSS Id.
OSS_IOR_FILENAMES	The IOR file names (including the absolute path) for the OSS. This IOR will be for the FromMangerToOSS interface in MGRTOOSS.IDL that you implement. Separate multiple pathnames with semi-colons.	…/ezTA/ior/MGRTOOSS.IOR	Absolute pathname of your IOR file.
IOR_PATH	The IOR pathname indicating the location of the IOR for the FromOSSToManager interface in the OSSTOMGR.IDL	…/ezTA/ior	Do not change this parameter
OSSTOMGR_IOR_ FILENAME	The IOR file name for the FromOSSToManager interface. The OSS should read this IOR file in the above directory and use it to bind to TA Manager	OSSTOMGR.IOR	Do not change this parameter

Using the Orbix daemon

If desired, use the Orbix daemon to communicate between the OSS and the TA Manager. You will need to start the Orbix daemon and register the CORBA servers with the daemon. The following table provides a list of these parameters and, where applicable, the new values required for your installation.

Table 2-3: Parameters affecting the Orbix daemon

Parameter	Description	Old Value	New Value
OSS_IDS	The OSS ID or account name from the MOI. Separate multiple values with semi-colons.	TEST_OSS	Your assigned OSS ID.
OSS_SERVER_NAMES	The CORBA server names for the OSS that will be registered with the daemon. Separate multiple values with semi-colons.	MGRTOOSSSrv	The OSS server name registered with Orbix.
OSS_HOST_NAMES	The host name where the server is registered. Separate multiple values with semi-colons.	Boreal	The host name where the server is running
OSS_MARKER_NAMES	The CORBA server marker names, where applicable.		If you registered your server with a marker name, specify the value here.
USE_IORS	A flag indicating the presence of IORs.	Y	N
OSSTOMGR_SERVER_NAME	The TA Manager CORBA server name registered with the daemon.	OSSTOMGRSrv	Do not change this parameter
OSSTOMGR_SERVER_MARKER_NAME	The TA Manager CORBA server marker name.	OSSTOMGR	Do not change this parameter

Understanding IDL Data Types

This section describes data types found in the following IDL files:

BASE.IDL

`BASE.IDL` has two data types. The first is the type definition for an invoke ID used with all operations.

```
typedef long InvokeId_t;
```

The second data type defines a CORBA exception that has a code and description along with it. All IDL operations in TA raise this kind of exception in case an error occurs.

```
exception ProcessingFailureError {
    string     code;
    string     description;
};
```

TIMETYPES.IDL

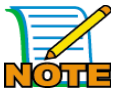
`TIMETYPES.IDL` has the data types that hold the `Date_t`, `Time_t`, `TimeInterval_t` and `Time24_t`. It also has several unions which hold optional values for all the types. These types are self-explanatory.

TATYPES.IDL

TATYPES.IDL has the data types that hold all kinds of TA attributes derived from the T1.227 and the T1.228 standards. CMIP definitions of the attributes in the standards are converted to their equivalent CORBA definitions.

The attribute types found in TATYPES.IDL are generated by converting them to a CORBA format. Some of the types have unions that hold optional values. For example, one of the optional types is defined as follows:

```
union StringOpt_t switch (boolean) {
    case TRUE: string element;
};
```



This definition holds an optional string value. If the discriminator is true, it contains a valid string. If the discriminator is false, the string is absent.

Optional IDL data types are provided for all attribute types. These are to be used to pass NULL values. Their definitions are same as described for the StringOpt_t type.

Table 2-4 describes the mapping between the CMIP and CORBA definitions of the attributes.

Table 2-4: CMIP to CORBA Mapping

CMIP Attribute Name	IDL Definition
Account Name	TATypes::AccountName_t
Called Number	TATypes::CalledNumber_t
Cancel Requested By Manager	TATypes::CancelRequestedByManager_t
Close Out Narrative	TATypes::CloseOutNarr_t
Close Out Verification	TATypes::CloseOutVerification_t
Customer Work Center	TATypes::CustomerWorkCenter_t
Customer Trouble Ticket Number	TATypes::CustTroubleTickNum_t
Initiating Mode	TATypes::InitiatingMode_t
Maintenance Service Charge	TATypes::MaintServiceChange_t

Table 2-4: CMIP to CORBA Mapping

CMIP Attribute Name	IDL Definition
Name Type	TATypes::NameType_t
Perceived Trouble Severity	TATypes::PerceivedTroubleSeverity_t
Preferred Priority	TATypes::PreferredPriority_t
Received Time	TATypes::ReceivedTime_t
Repeat Report	TATypes::RepeatReport_t
Restored Time	TATypes::RestoredTime_t
Trouble Detection Time	TATypes::TroubleDetectionTime_t
Trouble Found	TATypes::TroubleFound_t
Trouble Report ID	TATypes::TroubleReportId_t
Trouble Report State	TATypes::TroubleReportState_t
Trouble Report Status	TATypes::TroubleReportStatus_t
Trouble Report Status Time	TATypes::TroubleReportStatusTime_t
Trouble Type	TATypes::TroubleType_t
Tsp Priority	TATypes::TspPriority_t
A Location Access Person	TATypes::AlocationAccessPerson_t
Agent Contact Person	TATypes::AgentContactPerson_t
Manager Contact Person	TATypes::ManagerContactPerson_t
Trouble Clearance Person	TATypes::TroubleClearancePerson_t
Z Location Access Person	TATypes::ZlocationAccessPerson_t
Activity Duration List	TATypes::ActivityDurationList_t
Additional Trouble Info List	TATypes::AddTrblInfoList_t
Additional Trouble Status Info	TATypes::AddTrblStatusInfo_t
Authorization List	TATypes::AuthorizationList_t
Escalation List	TATypes::EscalationList_t
Commitment Time Request	TATypes::CommitmentTimeRequest_t

Table 2-4: CMIP to CORBA Mapping

CMIP Attribute Name	IDL Definition
Commitment Time	TATypes::CommitmentTime_t
A Location Access Address	TATypes::ALocationAccessAddress_t
Z Location Access Address	TATypes::ZlocationAccessAddress_t
Trouble Report Format Object Pointer	TATypes::TroubleReportFmtObjPtr_t
Managed Object Instance Alias List	TATypes::MOIAliasList_t
A Location Access Hours	TATypes::AlocationAccessHours_t
Managed Object Access Hours	TATypes::ManagedObjectAccessHours_t
Z Location Access Hours	TATypes::ZlocationAccessHours_t
Trouble Report Status Window	TATypes::TroubleReportStatusWindow_t
Outage Duration	TATypes::OutageDuration_t

TAATTRIBUTES.IDL

This section describes data types found in `TAATTRIBUTES.IDL`. The enumeration shown first provides unique IDs to all the attributes.

```
enum TAAttributes_t {
    unknownAttrType,
    aLocationAccessAddress,
    aLocationAccessHours,
    aLocationAccessPerson,
    accountName,
    activityDuration,
    additionalTroubleInfoList,
    additionalTroubleStatusInfo,
    agentContactPerson,
    authorizationList,
    calledNumber,
    cancelRequestedByManager,
    closeOutNarr,
    closeOutVerification,
    commitmentTime,
```

```
    commitmentTimeRequest,
    custTroubleTickNum,
    customerWorkCenter,
    escalationList,
    initiatingMode,
    maintServiceCharge,
    managedObjectAccessHours,
    managedObjectInstance,
    managedObjectInstanceAliasList,
    managerContactPerson,
    networkId,
    outageDuration,
    perceivedTroubleSeverity,
    preferredPriority,
    receivedTime,
    repeatReport,
    restoredTime,
    troubleClearancePerson,
    troubleDetectionTime,
    troubleFound,
    troubleReportFormatObjectPtr,
    troubleReportId,
    troubleReportState,
    troubleReportStatus,
    troubleReportStatusTime,
    troubleReportStatusWindow,
    troubleType,
    tspPriority,
    zLocationAccessAddress,
    zLocationAccessHours,
    zLocationAccessPerson,
    troubleReportCircuitId,
    troubleReportInstance
};
```

The following structure holds an attribute. It has a unique ID along with a value in CORBA Any form. Hence the attributes defined in `TATYPES.IDL` must be inserted and extracted back and forth from CORBA Any's. Use the helper classes or overloaded functions generated by the IDL compiler.

```
struct Attribute_t {
       TAAttributes_t id;
       any value;
};
```

This definition holds a list of attribute IDs.

```
typedef sequence<TAAttributes_t>  AttributeIdList_t;
```