



Table of Contents

1.3 Scope

3 FISSURES Model

3.1 Model Overview

3.4 Channel Model

Channel

4 Why CORBA?

4.1 Introduction to CORBA

- Large commercial support and development (no cost to IRIS)
- Standard services (name, trader, event...)
- Sendor neutral
- Langu.3(n(a)-11.3gu.3(neI)0.8 nu.3(neI)-11.3(u.3(nte)34(ral(CS)7, CS)7++, J)-53(n(a)0.8vu.3(n(a)0.8, t...))] TJ/F2

The sequence of events for the uploading of data for a particular channel occurs as follows:

- 1) The producer calls creat the SeismogramFactory, passing in the basics of the channel, e.g.


```
interface SeismogramFactory : SeismogramComponent
{
    ...
}
```

```
        ) raises (
            IllegalConversion
        );
    };
```

Someone that is not familiar with CORBA might ask "Is that it?" The answer in large part is yes,
ockmenofedoom wdo w-12.1th ssi(o)-52((f)854(t)-1056uh)687rnono no.9do(6g..9gu)0300.44 (

5 Interface Model

The seismogram manager create and managers seismograms. A seismogram manager can be

6 Interface Descriptions

This section describes the CORBA components as defined in 14.4.1-14.4.9.


```

//*****
    valuetype Quantity {
        public double value;
        public Unit the_units;
    };

    typedef Quantity Length;
    typedef Quantity TimeInterval;

public74sac7(U)-7.7(R)9.4(E)1.5(21 8.04 0[ 0 ]-3283tyermsuantity Tim271(0)]TJ /F4 1 Tf6.8(26/)5.8(0271(0)]TJ

```


6.3 Primary Components

6.3.1 Data Center

6.3.1.1 IfDataCenter module

```
//File: IfDataCenter.idl
//Version: 1999.06
//
...

#ifdef _IF_DATACENTER_IDL

//*****
//  Module DataCenter
//*****
module Fissures {
module IfDataCenter {

...

};
};
#endif // _IF_DATACENTER_IDL
```


6.3.2 Seismogram Manager

6.3.2.1 IfSeismogramMgr Module

```
//File: ISeismogramMgr.idl
```


6.3.2.2 Basic Types

```
//*****
//??? Seismogram Display terms - are these needed here or in the Fissures
// idl for the DisplayManager

struct Plottable {
    LongSeq x_coor;
    LongSeq y_coor;
};

//*****
//MirrorFilter elements will allow for wildcard characters
// ? for a single character, and * for 0 or more characters

struct MirrorFilter {
    string network_name;
    string station_name;
    string site_id;
    string 0..T0f_code;
};

typedef sequence<MirrorFilter> MirrorFilterSeq;
```


is_mirroring

6.3.2.5 SeismogramFinder Interface

```
/** *****  
//      SeismogramFinder  
/** *****  
    interface SeismogramFinder :
```

6.3.2.6 SeismogramFactory Interface

//*****

6.3.2.8 FactoryMirror

```
/** *****  
// FactoryMirror  
//  
// Seperate interface to allow a conformation point. DataCenter that  
// did not mirror would return null when an instance of this interface requested. of th21 interface interface  
**
```

//*****

6.3.2.11 Seismogram Interface

```
/** *****  
// Seismogram - provides read/write access to a seismogram  
/** *****  
    interface Seismogram :  
        SeismogramAccess,  
        SeismogramAdmin  
    {  
    };
```

The Seismogram interface provides a complete representation of a Seismogram, providing both access and write operations.

6.3.2.12 Seismogram Mirror

```
/** *****  
// Seismogram Mirror  
/** *****  
    interface SeismogramMirror :  
        SeismogramAdmin  
    {  
  
        void stop();  
  
    };
```

```
        LocatorElementSeq elements;
    };

    typedef string MagType;
    const MagType MB_MAG_TYPE="edu.iris.fissures\MagType\MB";
    const MagType MS_MAG_TYPE="edu.iris.fissures\MagType\MS";
    const MagType MW_MAG_TYPE="edu.iris.fissures\MagType\MW";
    const MagType MO_MAG_TYPE="edu.iris.fissures\MagType\MO";
    const MagType ML_MAG_TYPE="edu.iris.fissures\MagType\ML";

    struct Magnitude {MagType type;float value;

};
```



```
) raises (
```



```
in InformationAudit audit_info
U91i.FissuresExceptionTJ Ef 87.1610R
```


6.4.1.5 AuditSystem

6.4.2 Pick Manager

6.4.2.1 IfPickMgr Module

```
//File: IfPickMgr.idl  
//Version: 2000.02  
//
```

string name;

Retrieve all the picks associated to the given seismogram.

retrieve_pick_group

Retrieve all the collection of picks that are in the pick group.

get_pickgroups_for_dataset

Retrieve the pick groups associated to the given dataset.

6.4.2.5 PickManager Interface

Parameter Manager

ParameterMgr Module

```
//File: IfParameterMgr.idl
//Version: 2000.02
//

#ifdef _IF_PARAMETER_MGR_IDL
#define _IF_PARAMETER_MGR_IDL
```

```
/**
//*****
//      Module IfParameterMgr
//*****
module Fissures {
module IfParameterMgr {

    . . .

};
};

#endif // _IF_PARAMETER_MGR_IDL
```

Basic Types

```
/**
//*****
//      Parameter Terms
//*****
typedef string ParameterId;

struct Parm {
```



```
};    readonly attribute IfAuditSystem::AuditSystemAccess a_audit;
```

The ParameterComponent interface provides a single entry point into the parameter manager and provies navigation to support services.

a_writeable

Returns a writeable version of the manager.

get_value

Returns the value of the parameter identified by the parameter name, creator and parameter id.

get_parm_names

61..4 Time Series

61..4.1 IfTimeSeries Module

```
//File: IfTimeSeries.idl
//Version 1999.06

#ifndef _IF_TIMESERIES_IDL
#define _IF_TIMESERIES_IDL

#include "Fissures.idl"

#pragma prefix "edu.iris"

//*****
//  Module DataCenter
//*****

module Fissures {
module IfTimeSeries {

. . .

};
};

#endif // _IF_TIMESERIES_IDL
```

The TimeSeries Module contains the base data types and abstract interfaces for all time series object.

61..4.2 Basic Types

```
//*****
//  Data Types
//*****
```

```

};
typedef sequence<ComplexNumber> ComplexNumberSeq;

enum TimeSeriesType {TYPE_SHORT, TYPE_LONG, TYPE_FLOAT, TYPE_DOUBLE,
TYPE_ENCODED};

union TimeSeriesDataSel switch (TimeSeriesType) {
    case TYPE_SHORT: sequence<short> sht_values;
    case TYPE_LONG: sequence<long> int_values;
    case TYPE_FLOAT: sequence<float> flt_values;
    case TYPE_DOUBLE: sequence<double> dbl_values;
    case TYPE_ENCODED: EncodedData encoded_values;
};

typedef sequence<TimeSeriesDataSel> TimeSeriesDataSelSeq;

typedef sequence<short> ShortSeq;
typedef sequence<long> LongSeq;
typedef sequence<float> FloatSeq;
typedef sequence<double> DoubleSeq;

```

EncodedData

Contains times series data points that are encoded. The compression member defines the type of

```
) raises (  
    Fissures::FissuresException  
);  
  
ShortSeq get_as_shorts(  
    out CorrectedTime start_time  
) raises (  
    Fissures::FissuresException  
);  
  
FloatSeq get_as_floats(  
    out CorrectedTime start_time  
) raises (  
    Fissures::FissuresException  
);
```



```
};  
  
typedef sequence<VelocityModel> VelocityModelSeq;
```

|

|

|

|

|

|

|

|

6.5 Services

In the FISSURES framework services are defined to perform various task on seismic data. The abstract service interface is defined to provide a common interface to set service parameters. Specific services are free to define interface that set parameters in a more controlled means.

6.5.1 Service

6.5.1.1 Service Module

```
//File: IfService.idl
//Version 1999.06

. . .

#ifndef _IF_SERVICE_IDL
#define _IF_SERVICE_IDL
```

```

typedef string ConfigurationId;

const ConfigurationId DEFAULT_CONFIG = "DEFAULT";

//*****
//  Exceptions
//*****

exception UnknownOption {
    sequence<string> option_name;
};

exception InvalidValue {
    ServiceConfiguration bad_values;
};

exception UnknownConfiguration {
    ConfigurationId config_id;
};

```

ServiceOption

A setting options including name, description and possible values.

ServiceSetting

A instance of a option setting.

ServiceConfiguration

A collection of service settings.

6.5.1.3 Service Interface

6.5.3 Travel Time Calculator

6.5.3.1 IfTravelTimecalculator Module

```
//File: IfTravelTimeCalculator.idl
//Version 1999.06
//
. . .
#endif _IF_TRAVEL_TIME_CALCULATOR_IDL
```



```
//File: Fissures.idl
```

```

        long width;
        long height;
    };

//*****
//      Unit Terms
//*****

enum UnitBase {METER, GRAM, SECOND, AMPERE, KELVIN, MOLE, CANDELA, COUNT };

```

```

//          from UTC.
//*****
    struct Time {
        string date_time;
        long leap_seconds_version;
    };

    typedef sequence<Time> TimeSeq;

    struct CorrectedTime {
        Time best_estimate;
        long time_correction_version;
    };

    struct TimeRange {
        Time start_time;
        Time end_time;
    };

    struct ComplexNumber {
        float real_part;
        float imagicary_part;
    };
    typedef sequence<ComplexNumber> ComplexNumberSeq;

//*****
//      Response/Filter Terms
//*****
    valuetype Sampling {
        public long numPoints;
        public TimeInterval interval;
    };

    enum FilterType {COEFFICIENT, POLEZERO};

    struct CoefficientFilter {
        sequence<float> x_coeff;
        sequence<float> y_coeff;
    };

    struct PoleZeroFilter {
};

```



```
};

typedef sequence<AuditElement> AuditTrail;

//
// struct ActionAudit {
//     string service;
```

```

{

    readonly attribute AuditSystemFactory a_factory;


    void add_information_audit(
        in AuditId a_id,
        in Fissures::Time time_occurred,
        in InformationAudit step
    ) raises (
        Fissures::FissuresException
    );

};

};
};

#endif // _IF_AUDIT_SYSTEM_IDL

```



```

        public CorrectedTime begin_time;
        public long data_point_count;
        public Sampling sample_rate;
        public boolean time_corrected;
        public Unit y_unit;
        public ChannelId channel_id;

        SeismogramId get_id();
};

typedef sequence<SeismogramAttr> SeismogramAttrSeq;

valuetype SeismogramData {
    public TimeSeriesDataSel data;
};

valuetype LocalSeismogram :
    SeismogramAttr
{
    public SeismogramData my_data;
};

typedef sequence<LocalSeismogram> LocalSeismogramSeq;

//*****
//RequestFilter chraectrs;
//chraectr,rrr chraectrs;
//*****
struct

channelcodme;

LocalSeismogramSeq;

```



```

{
    DataSet create(
        in DataSetAttr attr,
        in InformationAudit audit_info
    );

    DataSet copy(
        in (
            in InformationAudit audit_info
        );
    );
};

//*****
// (
//*****
    interface DataSetAccess :
        DataSetComponent
    {
        readonly attribute (
            readonly attribute IfParameterMgr::ParameterMgrAccess a_parm_mgr;

        boolean is_locked(
        );

        DataSetAttr get_attributes();

        DataSetSeq get_children(
            in unsigned long hom_many,
            out (
        );

        ChannelGroupSeq get_channelgroups(
        );

        ChannelGroupSeq get_all_descendent_channelgroups(
        );

    };

//*****
// (
//*****
    interface DataSet :
        DataSetAccess
    {
        void lock(
            in ClientKey a_key
        ) raises (
            FissuresException
        );

        void unlock (
            in Clientkey a_key
        ) raises (
            FissuresException
        );

        void upd()e_d()]set(
            in ClientKey a_key,
            in DataSetAttr d()]set_attr,
            in InformationAudit audit_info
        ) raises (
            FissuresException
        );

        void add_d()]set(
            in ClientKey a_key,

```



```
        boolean next_n (  
            in unsigned long how_many,  
            out DataSetSeq names  
        );  
  
    };  
  
};  
  
#endif //_IF_DATASET_MGR_IDL
```



```

tyr.Ief string OriginId;

    struct Origin {
OriginId id;
        Time origin_time;
        Location my_location;
        MagnitudeSeq magnitudes;
        LocatorGroup origination;
    };

    tyr.Ief sequence<Origin> OriginSeq;

    tyr.Ief string Event;

    tyr.Ief sequence<Event> EventSeq;

//*****
//    EventManagerAccess Interface
//*****

    interface EventManagerAccess {

        EventSeq get_known_events(

```

```

        FissuresException
    );

    void associate_dataset(
        in DataSetId dataset,
        in Event a_event
    ) raises (
        FissuresException
    );

    void remove_association(
        in DataSetId dataset_id,
        in EventName event_name
    ) raises (
        FissuresException
    );

    void set_preferred(
        in 145 -1.1.4(eset_id,)]TJ 0 -1.1325 TD [(in)-booleanTD [(in)-4(set_preataset,)]TJ
    ) raises (

```


6 n 7 I f N e t w o r k M g r . i d l

```
//File: IfNetworkMgr.idl

//Version: 2000.02
//
//
//*****
//
//      Date          By          Description
//
//*****

#ifndef _IF_NETWORK_MGR_IDL
#define _IF_NETWORK_MGR_IDL

#include "Fissures.idl"
#include "IfAuditSystem.idl"

#pragma prefix "iris.edu"

//*****
//      Module DataCenter
//*****

module Fissures {
module IfNetworkMgr {

//*****
//      Data Types
//*****
typedef string ClockType;
const ClockType OMEGA_CLOCK = "edu.iris.fissures/clock/Omega";
const ClockType GOES_CLOCK = "edu.iris.fissures/clock/GOES";
const ClockType GPS_CLOCK = "edu.iris.fissures/clock/GPS";
const ClockType INTERNAL_CLOCK = "edu.iris.fissures/clock/Internal";

typedef string NetworkId;

valuetsTv snam97(ClockType;)TJ 0 -1public4(v)94typedef scodelock/Omega";k/Inter*****},*****
```



```

        private string id;
    };

    valuetype ChannelAttr :
        Channel
    {
        public string name;
        public Orientation an_orientation;
        public Sampling sample_rate;
        public Time effective_time;
        public Site my_site;
        public InstrumentConfig configuration;
    };

//*****
// Interfaces
//*****

    // forward references to interfaces
    interface NetworkFactory;
    interface NetworkAccess;
    interface Network;
    interface NetworkFinder;

//*****
// Interfaces
//*****
    interface NetworkComponent {
        readonly attribute NetworkFinder a_finder;
        readonly attribute IfAuditSystem::AuditSystemAccess a_audit;
    };

//*****
// Interfaces
//*****
    interface NetworkFinder :
        NetworkComponent
    {
        readonly attribute NetworkFactory a_factory;

```


7.8 IfParameterMgr.idl

```
//File: IfParameterMgr.idl
//Version: 2000.02
//
//*****
//*****
//
//      Date                By                Description
//
//*****

#ifndef _IF_PARAMETER_MGR_IDL
#define _IF_PARAMETER_MGR_IDL

#include "Fissures.idl"

#pragma prefix "edu.iris"

//*****
//      Module IfParameterMgr
//*****
module Fissures {
module IfParameterMgr {

//*****
//      Parameter Terms
//*****
    typedef string ParameterId;

    struct Parm {
        ParameterName parm_name;
        string description;
        string creator;
        any parm_value;
    };

    typedef sequence<Parm> ParmSeq;
```



```
void update_parm (  
    in ParameterId a_id,  
    in Parm parameter
```


7.10 IfSeismogramMgr.idl

```
//File: ISeismogramMgr.idl
//Version: 2000.02
//
//
//*****
//
//      Date          By          Description
//
//*****

#ifndef _IF_SEISMOGRAM_MGR_IDL

#include "Fissures.idl"
#include "IfTimeSeries.idl"

#include "IfParameterMgr.idl"

#include "IfNetworkMgr.idl"

#pragma prefix "edu.irisD [(#pT* (//***** )TJ

//*****

typedef Fissures::Time          Time;

typedef Fissures::Orientation   Orientation;

typedef IfTimeSeries::TimeSeriesAdmin TimeSeriesAdmin;

typedef IfParameterMgr::ParameterId ParameterId;

typedef IfTimeSeries::LongSeq    LongSeq;

typedef IfTimeSeries::FloatSeq   FloatSeq;

typedef IfTimeSeries::EncodedData      ::EncodedDaEncod84 TD F_//      DoubleSeq;
```



```

        ) raises (
            Fissures::FissuresException
        );

        SeismogramAccess create_from_ref(
            in Seismogram a_seismogram
        ) raises (
            Fissures::FissuresException
        );

        void destroy(
            in SeismogramSeq seismograms
        ) raises (
            Fissures::FissuresException
        );
    };

//*****
//*****

```



```

        out CorrectedTime start_time
        ) raises (
            Fissures::FissuresException
        );

        FloatSeq get_as_floats(
        out CorrectedTime start_time
        ) raises (
            Fissures::FissuresException
        );

        DoubleSeq get_as_doubles(
        out CorrectedTime start_time
        ) raises (
            Fissures::FissuresException
        );

        EncodedData get_as_encoded(
        out CorrectedTime start_time
        ) raises (
            Fissures::FissuresException
        );

    };

//*****
//    TimeSeries (ABSTRACT)
//*****
    abstract interface TimeSeriesAdmin {

        void append_longs(
            in LongSeq data
        ) raises (
            Fissures::FissuresException
        );

        void append_shorts(
            in ShortSeq data
        ) raises (
            Fissures::FissuresException
        );

        void append_floats(
            in FloatSeq data
        ) raises (
            Fissures::FissuresException
        );

        void append_doubles(
            in DoubleSeq data
        ) raises (
            Fissures::FissuresException
        );

        void append_with_encoded(
            in EncodedData data
        ) raises (
            Fissures::FissuresException
        );

    };

};
};

#endif // _IF_TIMESERIES_IDL

```

7.13 IfTravelTimeCalculator.idl


```
#endif //_IF_VELOCITY_MODEL_MGR_IDL
```