

30 September 2002

INTERFACE CONTROL DOCUMENT (ICD)
FOR
THE HAZARD PREDICTION AND ASSESSMENT
CAPABILITY (HPAC)
VERSION 4.0.1

Defense Threat Reduction Agency (DTRA)

RECORD OF CHANGE

| Change | Revision | Date | Description | Approved Change |
|--------|----------|-------------|--------------------|----------------------|
| xxx | 4.0.1 | 30 Sep 2002 | Inaugural HPAC ICD | <i>DTRA Approver</i> |

Contents

| | |
|---|-----------|
| I HPAC Operations | 1 |
| 1 General Information | 2 |
| 1.1 INTRODUCTION | 2 |
| 1.1.1 Scope | 2 |
| 1.1.2 Background | 2 |
| 1.2 OPERATIONAL ENVIRONMENT AND STANDARDS | 6 |
| 1.2.1 Operating System | 6 |
| 1.2.2 CORBA Environment | 6 |
| 1.2.3 Server Deployment Restrictions | 6 |
| 1.2.4 Standards | 6 |
| II Interfaces for Accessing HPAC | 8 |
| 2 HPACTool Library | 10 |
| 2.1 FORTRAN 90 MODULE DEFINITIONS | 10 |
| 2.1.1 contour_fd.f90 | 10 |
| 2.1.2 convert_fd.f90 | 10 |
| 2.1.3 domain_fd.f90 | 11 |
| 2.1.4 domstruct_fd.f90 | 12 |
| 2.1.5 effects_fd.f90 | 12 |
| 2.1.6 environment_fd.f90 | 13 |
| 2.1.7 field_fd.f90 | 14 |
| 2.1.8 filemgr_fd.f90 | 19 |
| 2.1.9 flags_fd.f90 | 20 |
| 2.1.10 hpactool_mod.f90 | 21 |
| 2.1.11 inpstruct_fd.f90 | 33 |
| 2.1.12 list_fd.f90 | 34 |
| 2.1.13 material_fd.f90 | 34 |
| 2.1.14 message_fd.f90 | 38 |
| 2.1.15 metstruct_fd.f90 | 39 |
| 2.1.16 msgstruct_fd.f90 | 39 |
| 2.1.17 mtlstruct_fd.f90 | 39 |
| 2.1.18 NWPNEffect_fd.f90 | 39 |
| 2.1.19 options_fd.f90 | 41 |
| 2.1.20 plotfx_fd.f90 | 41 |
| 2.1.21 plotlist_fd.f90 | 42 |

| | | |
|----------|--|------------|
| 2.1.22 | poparea_fd.f90 | 42 |
| 2.1.23 | prjstruct_fd.f90 | 43 |
| 2.1.24 | release_fd.f90 | 43 |
| 2.1.25 | relstruct_fd.f90 | 47 |
| 2.1.26 | sensor_fd.f90 | 47 |
| 2.1.27 | spcstruct_fd.f90 | 48 |
| 2.1.28 | structure_fd.f90 | 49 |
| 2.1.29 | terrain_fd.f90 | 52 |
| 2.1.30 | time_fd.f90 | 54 |
| 2.1.31 | timstruct_fd.f90 | 55 |
| 2.1.32 | tooluser_fd.f90 | 56 |
| 2.1.33 | type_fd.f90 | 61 |
| 2.1.34 | version_fd.f90 | 61 |
| 2.1.35 | weather_fd.f90 | 62 |
| 2.2 | C HEADER | 65 |
| 2.2.1 | hpactool.h | 65 |
| 3 | Detailed CORBA Services | 114 |
| 3.1 | CONFIGURING HPAC SERVICES | 114 |
| 3.1.1 | hpacserver.properties | 115 |
| 3.2 | LAUNCHING HPAC SERVICES | 117 |
| 3.2.1 | Windows Command Prompt launch batch file | 117 |
| 3.2.2 | Unix/Linux Bourne shell launch script | 120 |
| 3.2.3 | Unix/Linux Bourne shell terminate script | 122 |
| 3.3 | SERVICES OVERVIEW | 122 |
| 3.3.1 | Naming services | 124 |
| 3.3.2 | Incident Source Models | 124 |
| 3.3.3 | Interface Definition Language | 125 |
| 3.4 | DATA OBJECTS | 125 |
| 3.4.1 | files.idl | 126 |
| 3.4.2 | material.idl | 130 |
| 3.4.3 | project.idl | 136 |
| 3.4.4 | release.idl | 140 |
| 3.4.5 | server.idl | 159 |
| 3.4.6 | weatherT.idl | 170 |
| 3.5 | SCIPUFFFACTORY AND SCIPUFFSERVER | 176 |
| 3.5.1 | ScipuffServer Collaborations | 176 |
| 3.5.2 | effects.idl | 177 |
| 3.5.3 | plot.idl | 182 |
| 3.5.4 | scipuff.idl | 191 |
| 3.5.5 | ScipuffServer System Properties | 215 |
| 3.6 | MATERIAL SERVER | 215 |
| 3.7 | RADFILE MERGER | 217 |
| 3.7.1 | radfile.idl | 217 |
| 3.8 | STCALC SERVER | 220 |
| 3.8.1 | stcalc.idl | 220 |
| 3.9 | CHEMICAL-BIOLOGICAL FACILITY | 222 |

| | | |
|------------|--|------------|
| 3.9.1 | Services | 223 |
| 3.9.2 | AgentsList.idl | 223 |
| 3.9.3 | CBFactoHpac.idl | 226 |
| 3.9.4 | CloudTrans.idl | 227 |
| 3.9.5 | DamCat.idl | 227 |
| 3.9.6 | DWFI.idl | 228 |
| 3.9.7 | SWFI.idl | 230 |
| 3.9.8 | WeaponServer.idl | 233 |
| 3.10 | CHEMICAL-BIOLOGICAL WEAPON | 233 |
| 3.10.1 | ChemBioWeapon.idl | 233 |
| 3.11 | MISSILE INTERCEPT | 235 |
| 3.11.1 | MissileIntercept.idl | 235 |
| 3.12 | NUCLEAR WEAPON INCIDENT | 239 |
| 3.12.1 | nwi.idl | 239 |
| 3.13 | NUCLEAR FACILITY | 243 |
| 3.13.1 | Nfac.idl | 243 |
| 3.13.2 | analyst.idl | 248 |
| 3.14 | NUCLEAR WEAPON | 256 |
| 3.14.1 | Nwpn.idl | 256 |
| 3.15 | RADIOLOGICAL WEAPON | 257 |
| 3.15.1 | rwpn.idl | 257 |
| 3.16 | SMOKE MUNITION | 260 |
| 4 | IHPACServer | 261 |
| 4.1 | chembioweapon.idl | 261 |
| 4.2 | dispersion.idl | 263 |
| 4.3 | incident.idl | 264 |
| 4.4 | missileintercept.idl | 266 |
| 4.5 | nuclearweapon.idl | 267 |
| 4.6 | parameters.idl | 269 |
| 4.7 | plot.idl | 272 |
| 4.8 | project.idl | 276 |
| 4.9 | radweapon.idl | 278 |
| 4.10 | release.idl | 279 |
| 4.11 | server.idl | 284 |
| 5 | Java Beans and Components | 286 |
| 5.1 | SHARED JAR FILES | 286 |
| 5.2 | SERVER JAR FILES | 289 |
| 5.3 | CLIENT JAR FILES | 290 |
| III | Interfaces for Extending HPAC | 295 |
| 6 | Incident Source Models | 296 |
| 6.1 | MODEL SERVER FRAMEWORK | 296 |
| 6.1.1 | IncidentModelServerFactory | 296 |
| 6.1.2 | FactoryServer | 297 |

| | | |
|----------|---|-----|
| 6.1.3 | IncidentModelServer | 297 |
| 6.2 | MODEL BEAN FRAMEWORK | 299 |
| 6.2.1 | Model Bean | 299 |
| 6.2.2 | Model BeanInfo | 300 |
| 6.2.3 | Model Bean Customizer | 300 |
| 6.2.4 | Icon Definition | 301 |
| 6.2.5 | Configuring HPAC Models | 301 |
| 7 | Map Display | 303 |
| 7.1 | MAP DISPLAY FRAMEWORK | 303 |
| 7.1.1 | MapDisplay and MapProjection | 303 |
| 7.1.2 | ImageMapDisplay | 304 |
| 7.2 | OPENMAP MAP DISPLAY IMPLEMENTATION | 304 |
| 8 | Points of Interest | 306 |
| 8.1 | POI GROUP FRAMEWORK | 306 |
| 8.1.1 | POIGroup | 306 |
| 8.1.2 | SimplePOIGroup and SimplePOIOverlay | 306 |
| 8.1.3 | IncidentPOIOverlay and IncidentPOIIIcon | 307 |
| 8.2 | POI GROUP CONFIGURATION | 307 |
| 9 | Properties Serialization and Deserialization | 309 |
| 9.1 | PROPERTIES SERIALIZATION FRAMEWORK | 309 |
| 9.1.1 | String Property Representation | 309 |
| 9.1.2 | Hierarchical Object Property Representation | 311 |
| 9.1.3 | Implicit Serialization | 311 |
| 9.1.4 | Explicit Serialization | 311 |
| 9.1.5 | Arrays | 312 |
| 9.2 | PLUGGING INTO THE SERIALIZATION FRAMEWORK | 312 |
| A | Sample HPACtool Program | 314 |
| A.1 | HPACFunctions.h | 314 |
| A.2 | HPACFunctions.cpp | 323 |
| A.3 | HPACDefaults.h | 329 |
| A.4 | HPACDefaults.cpp | 338 |
| A.5 | example.cpp | 344 |
| A.6 | Makefile | 353 |
| A.7 | hpac.ini | 355 |
| A.8 | run.log | 356 |
| A.9 | run.out | 382 |
| B | Sample Detailed Services Client Programs | 403 |
| B.1 | DetailClient.java | 403 |
| B.2 | DetailClient.run.bat | 428 |
| B.3 | DetailClient.run.sh | 429 |
| B.4 | run.log | 430 |
| B.5 | run.out | 447 |

| | |
|--|------------|
| C Sample IHPACServer Client Program | 463 |
| C.1 IHPACClient.java | 463 |
| C.2 IHPACClient.run.bat | 467 |
| C.3 IHPACClient.run.sh | 468 |
| C.4 run.out | 468 |

List of Figures

| | | |
|-----|---|-----|
| 1.1 | CORBA communications | 3 |
| 1.2 | Factory pattern collaborations | 4 |
| 1.3 | Levels of HPAC access | 9 |
| 3.1 | Registered and per-client servers | 123 |
| 9.1 | Properties serialization framework. | 310 |

List of Tables

| | | |
|-----|---|-----|
| 3.1 | Server system properties | 118 |
| 3.2 | Incident source models | 125 |
| 3.3 | ScipuffServer system properties | 216 |
| 3.4 | ScipuffServerFactory system properties | 217 |
| 3.5 | Chemical-Biological Weapon model services | 224 |

List of Abbreviated Terms

| | |
|--------|---|
| API | application programming interface |
| COS | Common Object Services (CORBA) |
| COTS | commercial off-the-shelf |
| CORBA | Common Object Request Broker Architecture |
| DLL | dynamic link library |
| GUI | graphical user interface |
| HPAC | Hazard Prediction and Assessment Capability |
| IDL | [CORBA] Interface Definition Language |
| ITPTS | Integrated Target Planning Toolset |
| IOP | Internet Inter-Orb Protocol |
| JAR | Java archive |
| JFC | Java Foundation Classes |
| JNDI | Java Naming and Directory Interface |
| JVM | Java Virtual Machine |
| J2SE | Java 2 Standard Edition |
| OMG | Object Management Group |
| ORB | Object Request Broker |
| POI | Points of Interest |
| RMI | Remote Method Invocation |
| T&D | transport and dispersion |
| TCP/IP | Transmission Control Protocol/Internet Protocol |
| URL | Universal Resource Locator |

PART I

HPAC Operations

CHAPTER 1

General Information

1.1 INTRODUCTION

1.1.1 Scope

This document defines the interfaces for version 4.0.1 of the Hazard Prediction and Assessment Capability (HPAC). HPAC consists of components, libraries, and network accessible services providing many functions, including:

- calculation of atmospheric transport and dispersion of materials,
- assessing collateral effects,
- plotting dispersion and effects computations, and
- generating source terms from operational descriptions of incidents.

The scope of this document is limited to HPAC functionality and services and does not include auxiliary services often accessed during HPAC operation (e.g., Meteorological Data Servers) or environments in which HPAC is sometimes deployed.

1.1.2 Background

The HPAC application consists of individual components combined into a client application and a set of services. There are three classes of components: those shared between the client application and services, those used only on the client, and those dedicated to implementation of the services. Top level components include Project Editor client application and the server objects (service implementations) themselves.

HPAC may be deployed as a standalone application or in client-server mode, where the client application and service objects execute on a different hosts. Communication between client and server components uses standard Internet Transmission Control Protocol/Internet Protocol (TCP/IP) [1]. For a distributed object mechanism, HPAC uses the Common Object Request Broker Architecture (CORBA).

Applications and systems may access HPAC functionality by interacting with HPAC services, linking with HPAC libraries, or instantiating Java™ HPAC components. Further, an organization may extend HPAC by providing a custom incident source model implementation or supplying an implementation of one of the plugin frameworks defined by the HPAC client application.

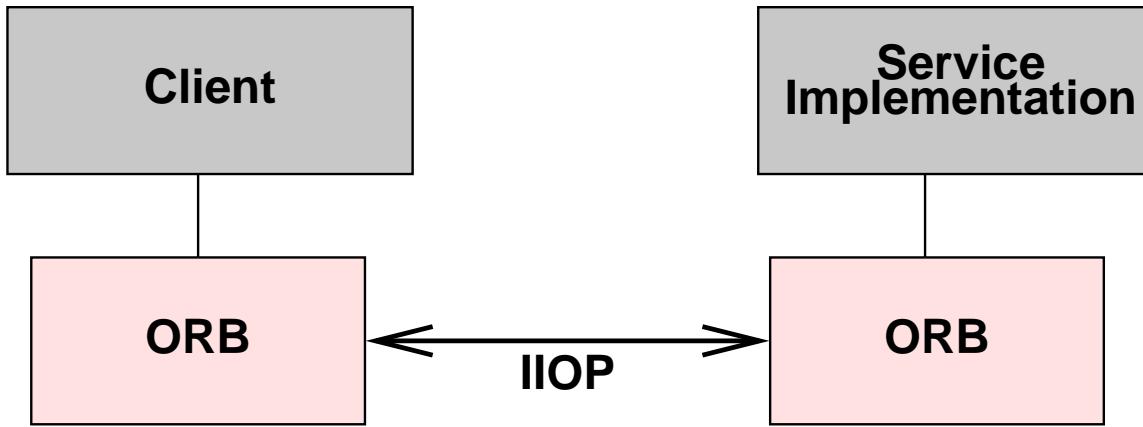


Figure 1.1: CORBA communications

HPAC Services

HPAC services are defined using the Object Management Group (OMG) Interface Definition Language (IDL), part of the CORBA specification. IDL definitions are compiled into a target language. For HPAC, the target language is always Java, but clients are limited only to languages supported by their Object Request Broker (ORB). Each client host must include a running ORB to communicate with the ORB running on the host with the service object. Communication between ORBs uses the Internet Inter-Orb Protocol (IIOP), as illustrated in Figure 1.1

For the most part, HPAC services follow the *factory pattern* [4] to support a connection-less, stateless processing model. The duration of a client-server session (i.e., the interactions between a client and a server object) is assumed to be relatively short. That is, clients do not obtain server references and hold them between groups of calls for related activity. Rather, a client obtains a per-client server, performs needed work, and then releases the server. When the same services are needed again later, another server instance is obtained. Thus, clients neither hold connections, nor do they depend upon any state which existed in the previous server instance. This eliminates the need for clients to check the state of connections and, if necessary, repair or re-establish them. Figure 1.2 illustrates the steps involved in client access to HPAC services.

Individual HPAC services are enumerated and described in Chapters 3 and 4.

Incident Source Models

The basis of HPAC is the transport and dispersion engine, the Second-order Closure Integrated Puff Model (SCIPUFF) developed by Titan Corporation [5, 6, 7, 8]. Input to SCIPUFF consists of detailed descriptions of releases and materials, meteorology or weather observations, and other auxilliary data such as land cover and terrain. HPAC includes several incident source models which take an operational description of an event or incident and produce one or more release descriptions for input to SCIPUFF.

HPAC is designed to be extensible with the addition of incident source models. Pluggable frameworks for source model services and client components are embedded in HPAC, and source model providers must supply implementations of both. The requirements and steps for implementing and plugging in a source model are described in Chapter 6.

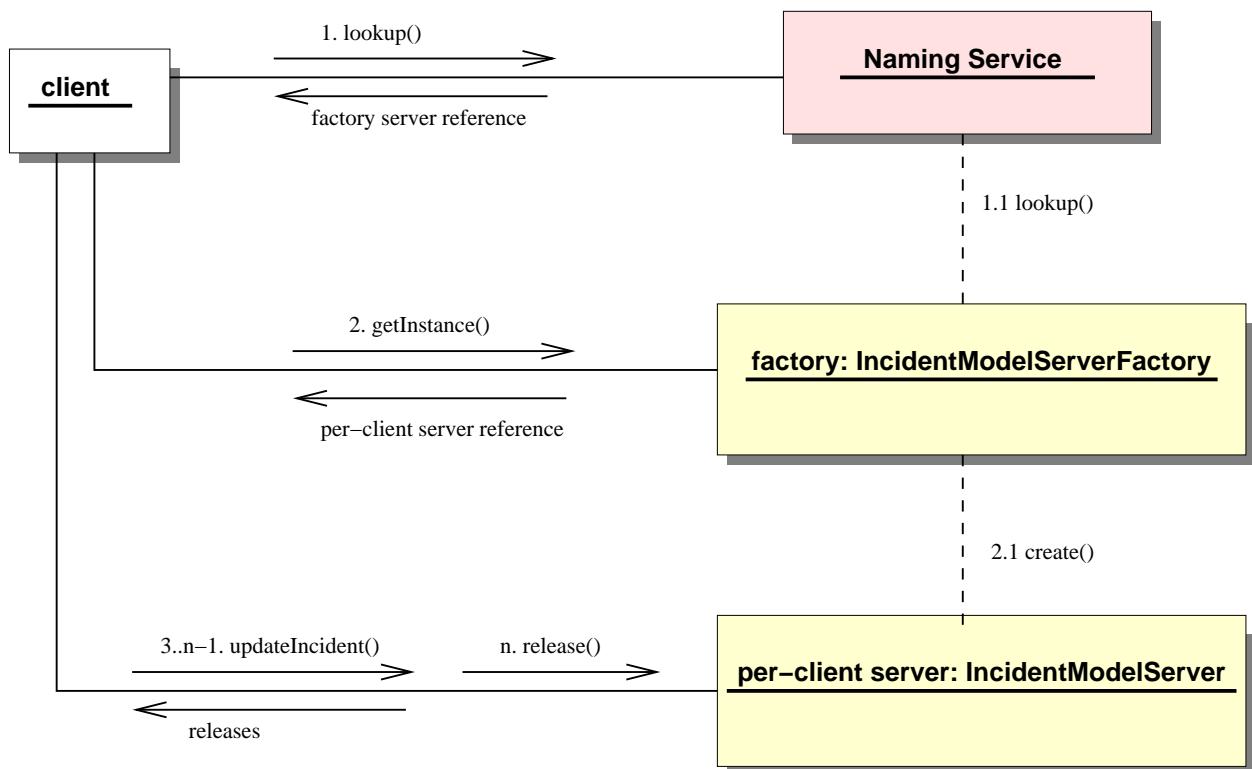


Figure 1.2: Factory pattern collaborations

Client Application Frameworks

In addition to source models, the HPAC client application provides two additional frameworks for customizing and/or extending HPAC: map displays and points of interest (POI).

The HPAC client application provides map displays based on OpenMap™ and simple image maps. The framework for implementing custom map displays and means of defining them for the application are described in Chapter 7.

The points of interest (POI) framework provides a means of defining data identifying facilities or locations that may be displayed on the map and optionally provide user interaction. HPAC provides two POI sets, nuclear reactors and weather stations. Chapter 8 describes implementation and definition of POI sets.

1.2 OPERATIONAL ENVIRONMENT AND STANDARDS

1.2.1 Operating System

HPAC Service Deployment

Many HPAC services are implemented in Java and are thus portable to any platform. Some services make use of native (e.g., C/C++, Fortran) code, but the native code has been ported to multiple platforms. Other services are implemented only for the Microsoft Windows server operating systems running on Intel processors, including NT 4.0 Workstation and Server, Windows 2000 Professional or Server, and Windows XP/Pro with no service pack requirements. In order to deploy all HPAC services, one of these operating systems is required.

Note HPAC services will execute under the Windows 9x series of operating systems (Windows 95, Windows 98, and Windows ME), but these operating systems are not supported.

Several HPAC services do not use native code and are thus platform portable. In addition to IHPACServer, specific portable services are identified in Chapter 3. These services may be hosted on any platform for which a Java 2 Standard Edition (J2SE) version 1.3 compliant environment is available [2].

Client Deployment

Clients accessing HPAC may be deployed on any platform for which a J2SE version 1.3 compliant environment is available.

1.2.2 CORBA Environment

HPAC uses the ORB supplied with the J2SE version 1.3. This ORB complies with version 2.0 of the CORBA specification [9] and the OMG IDL to Java Language Mapping Specification, formal, 99-07-03 [11]. The latter specification is aligned with revision 2.3 of the CORBA specification [10], but the J2SE ORB does not support "pass by value" semantics.

1.2.3 Server Deployment Restrictions

Deployment of HPAC services requires execution of the CORBA Common Object Services (COS) Naming Service provided with J2SE version 1.3.

Services accessed by HPAC client components running in a single application or Java Virtual Machine (JVM) instance must reside on the same server host.

1.2.4 Standards

CORBA

- The Common Object Request Broker: Architecture and Specification, Revision 2.0, Object Management Group, Inc., July, 1995.
- IDL to Java Language Mapping Specification, formal, 99-07-03, Object Management Group, Inc., June, 1999.

Java

- JavaTM 2 Platform, Standard Edition (J2SETM) Version 1.3.1.

PART II

Interfaces for Accessing HPAC

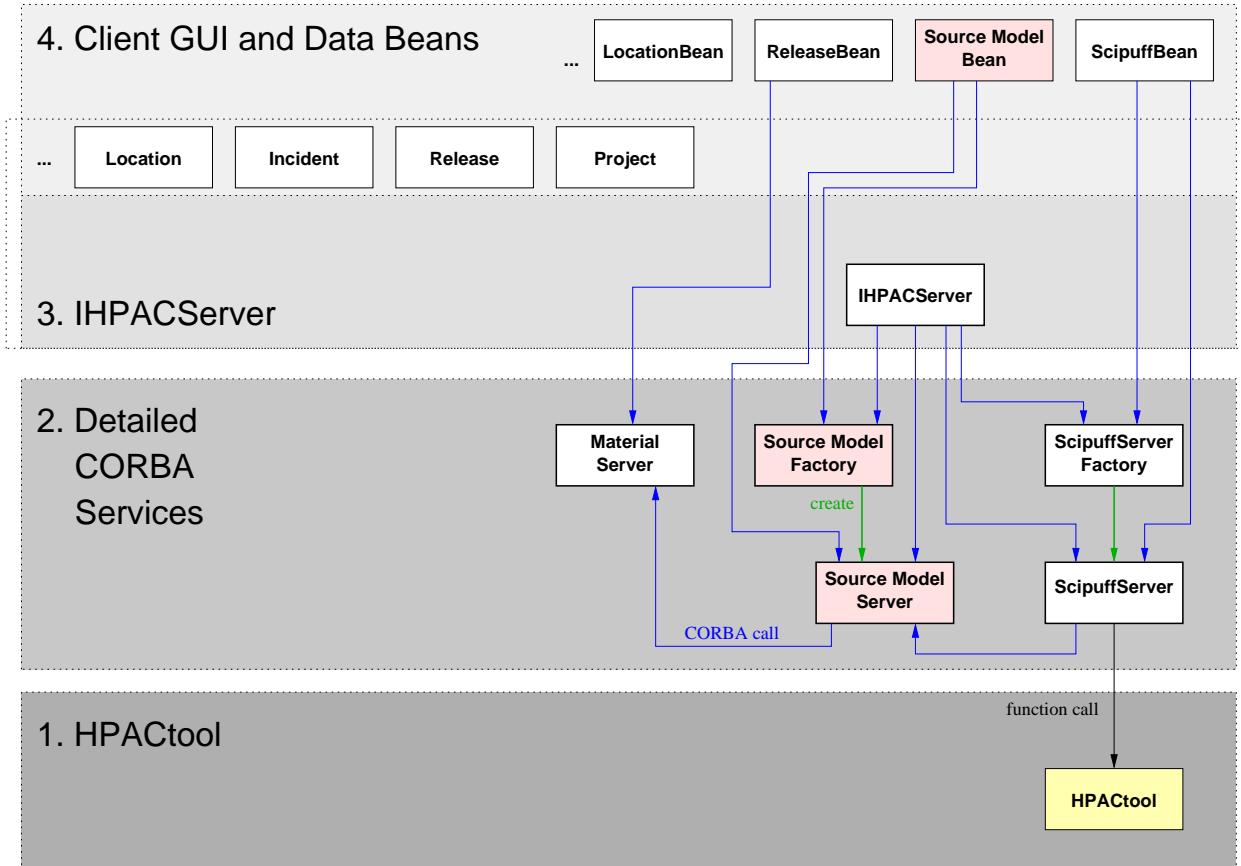


Figure 1.3: Levels of HPAC access

Four levels of HPAC components are exposed for access as illustrated in Figure 1.3:

- HPACtool library,
- detailed CORBA services,
- IHPACServer, and
- reusable Java beans and components.

The components identified in Figure 1.3 are representative and not an enumeration of all available components. Nonetheless, the figure depicts the relationship between the components. In particular, note that IHPACServer reuses many data components developed for the HPAC client application.

CHAPTER 2

HPACtool Library

HPACtool is a dynamic link library (DLL) available for programs written for 32-bit Microsoft Windows operating systems, including Windows 98, Windows ME, Windows NT 4.0 (workstation and server verions), Windows 2000 (professional and server verions), and Windows XP and XP/Pro. The application programming interface (API) for HPACtool is completely specified in a separate document [13]. HPACtool is written in Fortran 90 and provides module definitions for use by application programs. Further, C a language header file is provided for C programs.

HPACtool provides transport and dispersion (T&D) calculations and output generation. It includes no incident source model or effects computations. A sample C++ program accessing HPACtool is given in Appendix A.

Note HPACtool is non-reentrant. That is, each simultaneous HPACtool calculation must exist in a separate process.

2.1 FORTRAN 90 MODULE DEFINITIONS

Thirty-five module definition files listed below define data structures, functions, and subroutines for calls into the HPACtool DLL. For a detailed description of HPACtool functionality, refer to the HPACtool API document [13].

2.1.1 contour_fd.f90

```
MODULE contour_fd

  INTEGER, PARAMETER :: BLATLON_OUTPUT = 12
  INTEGER, PARAMETER :: LATLON_OUTPUT = 2**BLATLON_OUTPUT
  INTEGER, PARAMETER :: BCLOSE_CONTOUR = 11
  INTEGER, PARAMETER :: CLOSE_CONTOUR = 2**BCLOSE_CONTOUR

END MODULE contour_fd
```

2.1.2 convert_fd.f90

```
MODULE convert_fd
!==== HPAC conversion factors
```

```

REAL, PARAMETER :: HCF_HOUR2SEC = 3600.
REAL, PARAMETER :: HCF_SEC2HOUR = 1./HCF_HOUR2SEC

REAL, PARAMETER :: HCF_KM2M = 1000.
REAL, PARAMETER :: HCF_M2KM = 1./HCF_KM2M

REAL, PARAMETER :: HCF_M2MICRON = 1.e+6
REAL, PARAMETER :: HCF_MICRON2M = 1./HCF_M2MICRON

END MODULE convert_fd

```

2.1.3 domain_fd.f90

```

MODULE domain_fd

!===== Domain Type Parameters =====
INTEGER, PARAMETER :: HD_LATLON      = 1
INTEGER, PARAMETER :: HD_CARTESIAN   = 2
INTEGER, PARAMETER :: HD_UTM         = 3
INTEGER, PARAMETER :: HD_METERS      = 4

!===== referenceT =====
TYPE referenceT
  SEQUENCE
    REAL     x
    REAL     y
    REAL     lat
    REAL     lon
  END TYPE referenceT

!===== domainT =====
TYPE domainT
  SEQUENCE
    INTEGER map
    INTEGER zoneUTM
    REAL    xMax
    REAL    xMin
    REAL    yMax
    REAL    yMin
    REAL    zMax
    REAL    hRes
    REAL    vRes
  END TYPE domainT

!===== spatialT =====
TYPE spatialT
  SEQUENCE
    TYPE( domainT ) domain
    TYPE( referenceT ) reference
  END TYPE spatialT

```

```

END TYPE  spatialT

END MODULE domain_fd

```

2.1.4 domstruct_fd.f90

```

MODULE domstruct_fd
  USE domain_fd          !Basic domain structures
  USE prjstruct_fd       !Basic project structures

!===== pspatialT =====
  TYPE pspatialT
    SEQUENCE
      TYPE ( projectIDT ) project
      TYPE ( spatialT    ) spatial
    END TYPE pspatialT

END MODULE domstruct_fd

```

2.1.5 effects_fd.f90

```

MODULE effects_fd
!DEC$ IF DEFINED (NWPN)

!----- Release effects

  INTEGER,PARAMETER :: EI_MAXEFFECT      = 4

  INTEGER,PARAMETER :: EID_CIRCLE_BLAST   = 1
  INTEGER,PARAMETER :: EID_CIRCLE_THERM   = 2
  INTEGER,PARAMETER :: EID_CIRCLE_PROMPT  = 3
  INTEGER,PARAMETER :: EID_CASUALTY_PROMPT = 4

  INTEGER,PARAMETER :: EID_INIT           = 1
  INTEGER,PARAMETER :: EID_COMP            = 2
  INTEGER,PARAMETER :: EID_EXIT            = 3
  INTEGER,PARAMETER :: EID_COMP_NOCEP     = 4

  INTEGER,PARAMETER :: IEB_CIRCLE_BLAST    = EID_CIRCLE_BLAST - 1
  INTEGER,PARAMETER :: IEB_CIRCLE_THERM    = EID_CIRCLE_THERM - 1
  INTEGER,PARAMETER :: IEB_CIRCLE_PROMPT   = EID_CIRCLE_PROMPT - 1
  INTEGER,PARAMETER :: IEB_CASUALTY_PROMPT = EID_CASUALTY_PROMPT - 1

  INTEGER,PARAMETER :: IE_CIRCLE_BLAST     = 2**IEB_CIRCLE_BLAST
  INTEGER,PARAMETER :: IE_CIRCLE_THERM     = 2**IEB_CIRCLE_THERM
  INTEGER,PARAMETER :: IE_CIRCLE_PROMPT    = 2**IEB_CIRCLE_PROMPT
  INTEGER,PARAMETER :: IE_CASUALTY_PROMPT  = 2**IEB_CASUALTY_PROMPT

!DEC$ ENDIF
!DEC$ IF DEFINED (FXPLOT)
!----- Effect variables

```

```

INTEGER,PARAMETER :: EVB_GROUP      = 10 !Group bit
INTEGER,PARAMETER :: EVB_VAR        = 11 !Variance bit
INTEGER,PARAMETER :: EVB_HAZARD    = 15 !Hazard bit

INTEGER,PARAMETER :: EV_TOTAL       = 0   !Total
INTEGER,PARAMETER :: EV_VAPOR       = 1   !Liquid vapor phase
INTEGER,PARAMETER :: EV_LIQUID      = 2   !Liquid liquid phase (if deposition include secondary evaporation)
INTEGER,PARAMETER :: EV_LIQUID_DEP = 3   !Total Liquid liquid phase deposition
INTEGER,PARAMETER :: EV_GROUP       = (2**EVB_GROUP) !Group indicator
INTEGER,PARAMETER :: EV_VARIANCE    = (2**EVB_VAR)  !Variance indicator
INTEGER,PARAMETER :: EV_HAZARD     = (2**EVB_HAZARD) !Hazard indicator

!----- Plot effects structures

TYPE effectClassT
  SEQUENCE
    INTEGER      categoryID
    INTEGER      timeID
    INTEGER      linearInterp
    CHARACTER(64) name
  END TYPE effectClassT

TYPE effectPlotT
  SEQUENCE
    INTEGER      classIndx
    INTEGER      kindIndx
    INTEGER      category
    REAL         plotTime
    REAL         valueNotSet
  END TYPE effectPlotT

TYPE effectFieldT
  SEQUENCE
    INTEGER      typeID
    INTEGER      varID
    INTEGER      classDataIndex
    REAL         plotTime
  END TYPE effectFieldT

!DEC$ ENDIF
END MODULE effects_fd

```

2.1.6 environment_fd.f90

```

MODULE environment_fd

!==== parameters =====
INTEGER, PARAMETER :: HS_MAXENVIRO = 25
!==== enviroT = envsample_str =====

```

```

TYPE enviroT
SEQUENCE
REAL z !z
REAL pressure !mb
REAL potentialTemp !Potential Temp DegK
REAL humidity !gm/gm
REAL windUComp !m/sec E-W
REAL windVComp !m/sec N-S
REAL windWComp !m/sec
END TYPE enviroT

!===== environmentT = enviroment_str =====

TYPE environmentT
SEQUENCE
REAL sfcElevation !m
REAL sfcPressure !mb
REAL mixingLayerHeight !m
TYPE ( enviroT ) samples(HS_MAXENVIRO)
INTEGER nsamp
END TYPE environmentT

!===== enviroBLT =====

TYPE enviroBLT
SEQUENCE
REAL roughness !Roughness Height (m)
REAL canopy !Canopy Height (m) : <0 implies no canopy
REAL alpha !Canopy Flow index
REAL L !Monin-Obukov Length (m)
REAL surfaceLayer !Surface Layer height (m)
REAL mixingLayerHeight !Mixing Layer Height (m)
REAL wt !Surface heat flux (W/m2)
REAL uStar !Velocity Scale (Friction velocity) (m/s)
REAL wStar !Vertical velocity scale (m/s)
REAL dTdz !Surface temperature gradient (K/m)
END TYPE enviroBLT

END MODULE environment_fd

```

2.1.7 field_fd.f90

```

!=====
! Plot Field definition module
!=====
MODULE field_fd

USE domain_fd
USE AreaMode_fd

!----- Plot CLASS ID's

```

```

INTEGER, PARAMETER :: HP_EFFECT      = -1      !Effects
INTEGER, PARAMETER :: HP_CONC        =  0      !Concentration
INTEGER, PARAMETER :: HP_MET          =  1      !Meteorology/terrain
INTEGER, PARAMETER :: HP_DEP          =  2      !Surface deposition
INTEGER, PARAMETER :: HP_DOS          =  3      !Surface dosage
INTEGER, PARAMETER :: HP_SRFLOS       =  4      !Line-of-sight to surface
INTEGER, PARAMETER :: HP_TEXPOSE      =  5      !Toxic load exposure time

INTEGER, PARAMETER :: HP_PREVIOUS = -999 !Special for communicating with
                                         !effects module - use previous field
                                         !but add variance etc.

!----- Plot TIME LIST ID's

INTEGER, PARAMETER :: HP_NOTIME     =  0      !No times
INTEGER, PARAMETER :: HP_PUFFTIME    =  1      !Puff output times
INTEGER, PARAMETER :: HP_SRFTIME     =  2      !Surface integral output times
INTEGER, PARAMETER :: HP_METTIME     =  3      !Meteorology times
INTEGER, PARAMETER :: HP_RADTIME     =  4      !NFAC rad file times

!----- Plot CATEGORY indices

!DEC$ IF DEFINED (GULFWAR)
  INTEGER, PARAMETER :: HP_NUMCAT    =  7      !Number of types
!DEC$ ELSE
  INTEGER, PARAMETER :: HP_NUMCAT    =  6      !Number of types
!DEC$ ENDIF
  INTEGER, PARAMETER :: HP_SURF       =  1      !Surface integral field
  INTEGER, PARAMETER :: HP_SSLICE     =  2      !Surface slice
  INTEGER, PARAMETER :: HP_HSLICE     =  3      !Horizontal slice
  INTEGER, PARAMETER :: HP_VINT       =  4      !Vertically-integrated slice
  INTEGER, PARAMETER :: HP_VSLICE     =  5      !Vertical slice
  INTEGER, PARAMETER :: HP_TABLE      =  6      !Casualty table
!DEC$ IF DEFINED (GULFWAR)
  INTEGER, PARAMETER :: HP_SRFPRBMAX =  7      !Max probability from surface field
!DEC$ ENDIF

  INTEGER, PARAMETER :: HP_CATTYPE    =  0      !Special for communicating with
                                         !effects module - encode categories
                                         !as bits and this is a TYPE bit

!----- Plot CONTOUR output types

INTEGER, PARAMETER :: HP_RIGHTHOND  =  1      !Right-handed
INTEGER, PARAMETER :: HP_OPEN         =  0      !Open, i.e.,not closed contour
INTEGER, PARAMETER :: HP_LEFTOND    = -1      !Left-handed

REAL, PARAMETER      :: HP_SPV = -1.E+36

INTEGER, PARAMETER :: PLOT_ON       =  1
INTEGER, PARAMETER :: PLOT_OFF      = -1
INTEGER, PARAMETER :: PLOT_NULL     =  0

```

```

!=====
! PlotType Strings
!=====

!DEC$ IF DEFINED (GULFWAR)
CHARACTER(32), DIMENSION(HP_NUMCAT), PARAMETER :: CATEGORY_STRING = (/&
    'Surface           ', & !HP_SURF
    'Surface Slice     ', & !HP_SSLICE
    'Horizontal Slice  ', & !HP_HSLICE
    'Vertically Integrated Slice', & !HP_VINT
    'Vertical Slice    ', & !HP_VSLICE
    'Table             ', & !HP_TABLE
    'Max surface probability ', & !HP_SRFPRBMAX
/)
!DEC$ ELSE
CHARACTER(32), DIMENSION(HP_NUMCAT), PARAMETER :: CATEGORY_STRING = (/&
    'Surface           ', & !HP_SURF
    'Surface Slice     ', & !HP_SSLICE
    'Horizontal Slice  ', & !HP_HSLICE
    'Vertically Integrated Slice', & !HP_VINT
    'Vertical Slice    ', & !HP_VSLICE
    'Table             ', & !HP_TABLE
/)
!DEC$ ENDIF

!=====
! PlotField Point definition structure
!=====

TYPE HPACPointT
SEQUENCE
REAL           x          !X coordinate
REAL           y          !Y coordinate
END TYPE HPACPointT

!=====
! PlotField Contour Line definition structure
!=====

TYPE HPACLineT
SEQUENCE
INTEGER        index      !Contour index
INTEGER        start      !Start point
INTEGER        number     !Number of points
INTEGER        mode       !LeftHand/RightHand
END TYPE HPACLineT

!=====
! PlotField Vertical slice definition structure
!=====

TYPE HPACSliceT
SEQUENCE

```

```

      INTEGER             resolution      !Vertical slice resolution
      TYPE ( HPACPointT ) startPt       !Vertical slice Start Point
      TYPE ( HPACPointT ) endPt         !Vertical slice End Point
END TYPE HPACSliceT

!=====
! PlotField Coordinate definition structure
!=====

TYPE HPACFieldCoordinateT
  SEQUENCE
    INTEGER           mode          !HD_LATLON,HD_UTM,HD_CARTESIAN
    INTEGER           UTMZone       !UTM refernce zone if Mode=HD_UTM
    TYPE ( referenceT ) reference
      !Cartesian Reference point if Mode=HD_CARTESIAN
    TYPE ( HPACSliceT ) vertSlice   !Vertical Slice data if Mode < 0
END TYPE HPACFieldCoordinateT

!=====
! PlotField definition structure
!=====

TYPE HPACPlotFieldT
  SEQUENCE
    INTEGER           category     !Plot Category ID
    INTEGER           class        !Index into ClassStr array
    INTEGER           choice       !Index into ChoiceStr array
    INTEGER           kind         !Index into KindStr array
    INTEGER           timeID      !Time list ID (Puff,Srf,Met,Rad)
    REAL              userTime    !Run Time
    INTEGER           hazard      !.true. -> Use Hazard puffs
    INTEGER           maxCells
      !Maximum number of cells to allocate
    INTEGER           maxLev
      !Maximun number of levels of refinememnt
    INTEGER           fldLev
      !Actual number of levels of refinememnt - Filled in by HPACCCreateField
    REAL              resolution
      !Minimum grid cell size - Filled in by HPACCCreateField
    INTEGER           interpType
      !Interpolation type - Filled in by HPACCCreateField
    TYPE( HPACFieldCoordinateT ) coordinate
      !Field coordinate structure - Filled in by HPACCCreateField
    CHARACTER(16)      units
      !Field Units - Filled in by HPACCCreateField
    CHARACTER(128)     project
    CHARACTER(128)     path
END TYPE HPACPlotFieldT

!=====
! PlotField Category data structure
!=====


```

```

TYPE HPACPlotData
SEQUENCE
  REAL xmin      !Vert. Slice - X coordinate LH point
  REAL xmax      !Vert. Slice - X coordinate RH point
  REAL ymin      !Vert. Slice - Y coordinate LH point
  REAL ymax      !Vert. Slice - Y coordinate RH point
  REAL zmin      !Vert. Slice - Min Z value : Horz. Slice - Slice value
  REAL zmax      !Vert. Slice - Max Z value
  REAL zres      !Vert. Slice - Vertical resolution (Number of points)
END TYPE HPACPlotData

!=====
! Contour definiton structure
!=====

TYPE HPACContourElementT
SEQUENCE
  REAL          contour    !Contour value
  REAL          value      !Auxiliary input data (Hazard Area = exceed value)
  CHARACTER(16) label      !Display label
  REAL          area       !Contour Area
  REAL          population !Population associated with contour
END TYPE HPACContourElementT

!=====
! Contour list definition structure
!=====

TYPE HPACContourHeaderT
SEQUENCE
  INTEGER        number     !Number of contours in list
  REAL           scale      !Scale factor (Default=1.0)
  INTEGER        labelMode   !PLOT_ON/PLOT_OFF/PLOT_NULL
  INTEGER        drawMode    !PLOT_FILL/PLOT_DRAW/PLOT_BOTH/PLOT_OFF/PLOT_ON
  CHARACTER(16)  unit       !Units (Default='default')
END TYPE HPACContourHeaderT

!=====
! PlotType definition structure
!=====

TYPE HPACPlotTypeT
SEQUENCE
  INTEGER type      !PlotType Mean,Prob,Exceed,Hazard Area
  REAL   data       !Type data : Prob->exceed level :
  INTEGER areaMode
END TYPE HPACPlotTypeT

!=====
! DrawField drawing instruction structure
!=====

```

```

TYPE ARAPDrawT
  SEQUENCE
    INTEGER fillContour
    INTEGER drawContour
    INTEGER fill_Lo
    INTEGER fill_Hi
  END TYPE ARAPDrawT

!=====
!  WriteField drawing instruction structure
!=====

TYPE ARAPWriteT
  SEQUENCE
    INTEGER mode
    CHARACTER(128) filename
  END TYPE ARAPWriteT

!=====
!  PlotField Node definition structure
!=====

TYPE HPACPlotFieldNodeT
  SEQUENCE
    INTEGER      id          !Node ID value
    REAL         x           !X coordinate
    REAL         y           !Y coordinate
    REAL         z           !Z coordinate
    REAL         hx          !X grid resolution
    REAL         hy          !Y grid resolution
    REAL         v            !Field value
  END TYPE HPACPlotFieldNodeT

!=====
!  PlotField Triangle definition structure
!=====

TYPE HPACPlotFieldTriangleT
  SEQUENCE
    INTEGER      id          !Triangle ID value
    INTEGER      nidA        !Node ID value
    INTEGER      nidB        !Node ID value
    INTEGER      nidC        !Node ID value
  END TYPE HPACPlotFieldTriangleT

END MODULE field_fd

```

2.1.8 filemgr_fd.f90

```

MODULE filemgr_fd
!==== File Manager Results

  INTEGER, PARAMETER :: FileMgrUnknown = -2

```

```

INTEGER, PARAMETER :: FileMgrFailure = -1
INTEGER, PARAMETER :: FileMgrNull      = 0
INTEGER, PARAMETER :: FileMgrSuccess   = 1
INTEGER, PARAMETER :: FileMgrNegative  = FileMgrFailure
INTEGER, PARAMETER :: FileMgrAffirmative = FileMgrSuccess
INTEGER, PARAMETER :: FileMgrValid     = FileMgrSuccess
INTEGER, PARAMETER :: FileMgrInvalid   = FileMgrFailure
INTEGER, PARAMETER :: FileMgrError     = FileMgrFailure
INTEGER, PARAMETER :: FileMgrOn        = FileMgrSuccess
INTEGER, PARAMETER :: FileMgrOff       = FileMgrNull

!==== File Manager LastError codes

INTEGER, PARAMETER :: FME_EOF = -1
INTEGER, PARAMETER :: FME_LISTSIZE = 1
INTEGER, PARAMETER :: FME_NOTAVAIL = 2
INTEGER, PARAMETER :: FME_NOTFOUND = 3
INTEGER, PARAMETER :: FME_FAILURE = 4

END MODULE filemgr_fd

```

2.1.9 flags_fd.f90

```

MODULE flags_fd

!==== HPAC input flags types

INTEGER,PARAMETER :: HF_OFF          = 0
INTEGER,PARAMETER :: HFB_DYNAMIC     = 0
!DEC$ IF DEFINED (DENSE)
    INTEGER,PARAMETER :: HFB_DENSE     = 1
!DEC$ ENDIF
    INTEGER,PARAMETER :: HFB_STATIC    = 2
!DEC$ IF DEFINED (MULTCOMP)
    INTEGER,PARAMETER :: HFB_MULTICOMP = 3
!DEC$ ENDIF
    INTEGER,PARAMETER :: HF_DYNAMIC    = 2**HFB_DYNAMIC
!DEC$ IF DEFINED (DENSE)
    INTEGER,PARAMETER :: HF_DENSE      = 2**HFB_DENSE
!DEC$ ENDIF
    INTEGER,PARAMETER :: HF_STATIC     = 2**HFB_STATIC
!DEC$ IF DEFINED (MULTCOMP)
    INTEGER,PARAMETER :: HF_MULTICOMP   = 2**HFB_MULTICOMP
!DEC$ ENDIF

    INTEGER,PARAMETER :: HFB_FAST       = 0
    INTEGER,PARAMETER :: HF_FAST        = 2**HFB_FAST

!DEC$ IF DEFINED (HAZARD)
    INTEGER,PARAMETER :: HFB_HAZARD    = 1
    INTEGER,PARAMETER :: HFB_DUAL       = 2
    INTEGER,PARAMETER :: HF_HAZARD      = 2**HFB_HAZARD

```

```

INTEGER,PARAMETER :: HF_DUAL      = 2**HFB_DUAL
!DEC$ ENDIF

INTEGER,PARAMETER :: HFB_RESTART = 0
INTEGER,PARAMETER :: HF_RESTART  = 2**HFB_RESTART

!==== auditT =====
TYPE auditT
  SEQUENCE
    CHARACTER(80) title      !title
    CHARACTER(32) analyst    !audit_analyst
    CHARACTER(32) class      !audit_class
    CHARACTER(32) version    !audit_version
    CHARACTER(32) date       !audit_date
  END TYPE auditT

!==== flagT =====
TYPE flagsT
  SEQUENCE
    INTEGER        start      !(new/restart)
    INTEGER        method     !dynamic,dense_gas,static,multicomp
    INTEGER        mode       !run_mode,hazard
    INTEGER        prjEffects !HPAC 4.0 project effecs (incident based)
    TYPE ( auditT ) audit
  END TYPE flagsT

END MODULE flags_fd

```

2.1.10 hpactool_mod.f90

```

MODULE HPACtool

!Interface definitions for the HPACtool exported functions
! NB - can not write interfaces for three functions
!       HPACDefaultInput, HPACLoadInput, HPACWriteInput
! because the arguments are not of a fixed type

INTERFACE
  INTEGER FUNCTION HPACButton(CallerID,ButtonID)
    INTEGER, INTENT( IN ) :: CallerID      !USER ID Tag
    INTEGER, INTENT( IN ) :: ButtonID      !Progress Box Button ID
  END FUNCTION HPACButton
END INTERFACE

INTERFACE
  INTEGER FUNCTION HPACCContourCount( &
    CallerID, grdID, Field, PlotType, contourHead, contourList, &
    Mode, nLine, nPoint &
  )
  USE tooluser_fd
END INTERFACE

```

```

INTEGER,
INTEGER,
TYPE( HPACPlotFieldT ),
TYPE( HPACPlotTypeT ),
TYPE( HPACContourHeaderT ),
TYPE( HPACContourElementT ), DIMENSION(*),
INTEGER,
INTEGER,
INTEGER,
INTENT( IN ) :: CallerID
INTENT( IN ) :: grdID
INTENT( IN ) :: Field
INTENT( IN ) :: PlotType
INTENT( IN ) :: contourHead
INTENT( IN ) :: contourList
INTENT( IN ) :: Mode
INTENT( OUT ) :: nLine
INTENT( OUT ) :: nPoint
END FUNCTION HPACContourCount
END INTERFACE

INTERFACE
INTEGER FUNCTION
HPACContourField( &
    CallerID,grdID,Field,PlotType,contourHead,contourList,Mode,Line,Point &
)
USE tooluser_fd
INTEGER,
INTEGER,
TYPE( HPACPlotFieldT ),
TYPE( HPACPlotTypeT ),
TYPE( HPACContourHeaderT ),
TYPE( HPACContourElementT ), DIMENSION(*),
INTEGER,
TYPE( HPACLineT ), DIMENSION(*),
TYPE( HPACPointT ), DIMENSION(*),
INTENT( IN ) :: CallerID
INTENT( IN ) :: grdID
INTENT( IN ) :: Field
INTENT( IN ) :: PlotType
INTENT( IN ) :: contourHead
INTENT( INOUT ) :: contourList
INTENT( IN ) :: Mode
INTENT( OUT ) :: Line
INTENT( OUT ) :: Point
END FUNCTION HPACContourField
END INTERFACE

INTERFACE
INTEGER FUNCTION HPACCountMaterial(CallerID,file,nMtl)
USE tooluser_fd
INTEGER,           INTENT( IN ) :: CallerID !USER ID tag
TYPE( char128T ), INTENT( IN ) :: file      !filename
INTEGER,           INTENT( OUT ) :: nMtl      !number of materials in file
END FUNCTION HPACCountMaterial
END INTERFACE

INTERFACE
INTEGER FUNCTION HPACCountRelease(CallerID,file,nRel)
USE tooluser_fd
INTEGER,           INTENT( IN ) :: CallerID !USER ID tag
TYPE( char128T ), INTENT( IN ) :: file      !filename
INTEGER,           INTENT( OUT ) :: nRel      !number of releases in file
END FUNCTION HPACCountRelease
END INTERFACE

INTERFACE
INTEGER FUNCTION HPACCCreateField(CallerID,Field,ClassData)
USE tooluser_fd
INTEGER,           INTENT( IN      ) :: CallerID !USER ID Tag
TYPE( HPACPlotFieldT ), INTENT( INOUT ) :: Field     !Field descriptor

```

```

REAL, DIMENSION(*), INTENT( IN      ) :: ClassData
    !Additional Class data
END FUNCTION HPACCCreateField
END INTERFACE

INTERFACE
    INTEGER FUNCTION HPACCurrentTerrain(Location,Height)
        USE tooluser_fd
        TYPE( HPACPointT ), INTENT( IN      ) :: Location !User specified location
        REAL,           INTENT( INOUT   ) :: Height   !Terrain height
    END FUNCTION HPACCurrentTerrain
END INTERFACE

INTERFACE
    INTEGER FUNCTION HPACCurrentWeather(Location,Environment,BoundaryLayer)
        USE tooluser_fd
        TYPE( HPACPointT ), INTENT( IN      ) :: Location
            !User specified location
        TYPE( environmentT ), INTENT( INOUT   ) :: Environment
            !Local environment
        TYPE( enviroBLT ),   INTENT( OUT     ) :: BoundaryLayer
            !Local BL parameters
    END FUNCTION HPACCurrentWeather
END INTERFACE

INTEGER, EXTERNAL :: HPACDefaultInput
INTERFACE
    INTEGER FUNCTION HPACDefaultInput(CallerID,iInput,tInput0,tInput1)
        INTEGER, INTENT( IN      ) :: CallerID      !USER ID tag
        INTEGER, INTENT( IN      ) :: iInput        !Request type
        INTEGER, INTENT( OUT     ) :: tInput0       !Results data
        INTEGER, INTENT( OUT     ) :: tInput1       !Results data
    END FUNCTION HPACDefaultInput
END INTERFACE

INTERFACE
    INTEGER FUNCTION HPACDeleteField(CallerID,grdID)
        USE tooluser_fd
        INTEGER, INTENT( IN      ) :: CallerID      !USER ID tag
        INTEGER, INTENT( INOUT   ) :: grdID         !SAG grid ID
    END FUNCTION HPACDeleteField
END INTERFACE

INTERFACE
    INTEGER FUNCTION HPACDeleteProject(CallerID,project,request)
        USE tooluser_fd
        INTEGER,           INTENT( IN      ) :: CallerID !USER ID tag
        TYPE ( projectIDT ), INTENT( IN      ) :: project !Project ID
        INTEGER,           INTENT( IN      ) :: request   !Delete instructions
    END FUNCTION HPACDeleteProject
END INTERFACE

INTERFACE

```

```

INTEGER FUNCTION
HPACDrawField( &
    CallerID, grdID, Field, PlotType, contourHead, contourList, &
    GUIdraw, UserFill, UserDraw &
)
USE tooluser_fd
INTEGER,
    !USER ID tag
INTEGER,
    !SAG grid ID
TYPE( HPACPlotFieldT ),
    !Field descriptor
TYPE( HPACPlotTypeT ),
    !Plot definition
TYPE( HPACContourHeaderT ),
    !Contour array header
TYPE( HPACContourElementT ), DIMENSION(*), INTENT( INOUT ) :: contourList
    !Contour array
TYPE( ARAPDrawT ),
    !Draw instructions
INTEGER, EXTERNAL
    :: UserFill
    !Address of User supplied fill routine passed by value
INTEGER, EXTERNAL
    :: UserDraw
    !Address of User supplied draw routine passed by value
END FUNCTION HPACDrawField
END INTERFACE

INTERFACE
    INTEGER FUNCTION HPACDrawGrid(CallerID,grdID,UserDraw,mode)
        INTEGER,           INTENT( IN ) :: CallerID   !USER ID tag
        INTEGER,           INTENT( IN ) :: grdID      !SAG grid ID
        INTEGER, EXTERNAL          :: UserDraw     !Address of User supplied
                                         !draw routine passed by value
        INTEGER,           INTENT( IN ) :: mode       !Draw instructions
    END FUNCTION HPACDrawGrid
END INTERFACE

INTERFACE
    INTEGER FUNCTION HPACExitTool( )
    END FUNCTION HPACExitTool
END INTERFACE

INTERFACE
    INTEGER FUNCTION HPACGetAPIVersion( )
    END FUNCTION HPACGetAPIVersion
END INTERFACE

INTERFACE
    INTEGER FUNCTION HPACGetField(CallerID,grdID,Field,PlotType,nodes,triangles)
        USE tooluser_fd
        INTEGER,
            !USER ID tag
        INTEGER,
            INTENT( IN ) :: CallerID
            INTENT( IN ) :: grdID

```

```

        !SAG grid ID
TYPE( HPACPlotFieldT ),           INTENT( IN ) :: Field
        !Field descriptor
TYPE( HPACPlotTypeT ),           INTENT( IN ) :: PlotType
        !Plot definition
TYPE( HPACPlotFieldNodeT ),       DIMENSION(*),INTENT( OUT ) :: nodes
        !Plot definition
TYPE( HPACPlotFieldTriangleT ),  DIMENSION(*),INTENT( OUT ) :: triangles
        !Plot definition
END FUNCTION HPACGetField
END INTERFACE

INTERFACE
  INTEGER FUNCTION HPACGetFieldDomain(CallerID,grdID,nx0,ny0,x0,y0,dx0,dy0)
    INTEGER, INTENT( IN ) :: CallerID      !USER ID tag
    INTEGER, INTENT( IN ) :: grdID        !SAG grid ID
    INTEGER, INTENT( OUT ) :: nx0          !Base grid number X direction
    INTEGER, INTENT( OUT ) :: ny0          !Base grid number Y direction
    REAL,   INTENT( OUT ) :: x0            !Origin X direction
    REAL,   INTENT( OUT ) :: y0            !Origin Y direction
    REAL,   INTENT( OUT ) :: dx0           !Base grid size X direction
    REAL,   INTENT( OUT ) :: dy0           !Base grid size Y direction
  END FUNCTION HPACGetFieldDomain
END INTERFACE

INTERFACE
  INTEGER FUNCTION
    HPACGetFieldMinMax(CallerID,grdID,PlotType,mMin,mMax,vMin,vMax,fMin,fMax)
    USE tooluser_fd
    INTEGER,             INTENT( IN ) :: CallerID !USER ID tag
    INTEGER,             INTENT( IN ) :: grdID   !SAG grid ID
    TYPE( HPACPlotTypeT ), INTENT( IN ) :: PlotType !Plot definition
    REAL,               INTENT( OUT ) :: mMin   !Minimum value - mean field
    REAL,               INTENT( OUT ) :: mMax   !Maximum value - mean field
    REAL,               INTENT( OUT ) :: vMin   !Minimum value - variance field
    REAL,               INTENT( OUT ) :: vMax   !Maximum value - variance field
    REAL,               INTENT( OUT ) :: fMin   !Minimum value - plot field
    REAL,               INTENT( OUT ) :: fMax   !Maximum value - plot field
  END FUNCTION HPACGetFieldMinMax
END INTERFACE

INTERFACE
  INTEGER FUNCTION
    HPACGetFieldSize(CallerID,grdID,Field,PlotType,nNode,nTriangle)
    USE tooluser_fd
    INTEGER,             INTENT( IN ) :: CallerID !USER ID tag
    INTEGER,             INTENT( IN ) :: grdID   !SAG grid ID
    TYPE( HPACPlotFieldT ), INTENT( IN ) :: Field    !Field descriptor
    TYPE( HPACPlotTypeT ), INTENT( IN ) :: PlotType !Plot definition
    INTEGER,             INTENT( OUT ) :: nNode
        !Number of nodes in plot field
    INTEGER,             INTENT( OUT ) :: nTriangle
        !Number of triangels in plot field

```

```

    END FUNCTION HPACGetFieldSize
END INTERFACE

INTERFACE
    INTEGER FUNCTION
    HPACGetFieldTable( &
        CallerID, Field, ClassData, TableTitle, ColTitle, &
        RowTitle, Table &
    )
    USE tooluser_fd
    INTEGER,                                     INTENT( IN      ) :: CallerID
                                                !USER ID tag
    TYPE( HPACPlotFieldT ),           INTENT( INOUT ) :: Field
                                                !Field descriptor
    REAL,          DIMENSION(*), INTENT( IN      ) :: ClassData
                                                !ClassDataArray (Risk Level - Dual runs only
    TYPE( char32T ), DIMENSION(*), INTENT( OUT     ) :: TableTitle
                                                !Table titles
    TYPE( char32T ), DIMENSION(*), INTENT( OUT     ) :: ColTitle
                                                !Column headings
    TYPE( char32T ), DIMENSION(*), INTENT( OUT     ) :: RowTitle
                                                !Row headings
    INTEGER,          DIMENSION(*), INTENT( OUT     ) :: Table
                                                !Table data
    END FUNCTION HPACGetFieldTable
END INTERFACE

INTERFACE
    INTEGER FUNCTION
    HPACGetFieldTableSize(CallerID,Field,ClassData,nTable,nCol,nRow)
    USE tooluser_fd
    INTEGER,                                     INTENT( IN      ) :: CallerID  !USER ID tag
    TYPE( HPACPlotFieldT ), INTENT( IN      ) :: Field      !Field descriptor
    REAL,          DIMENSION(*), INTENT( IN      ) :: ClassData
                                                !ClassDataArray (Risk Level - Dual runs only
    INTEGER,          INTENT( OUT     ) :: nTable    !Number of tables
    INTEGER,          INTENT( OUT     ) :: nCol
                                                !Number of columns per table
    INTEGER,          INTENT( OUT     ) :: nRow
                                                !Number of rows per table
    END FUNCTION HPACGetFieldTableSize
END INTERFACE

INTERFACE
    INTEGER FUNCTION HPACGetFieldValue(CallerID,grdID,PlotType,xPnt,yPnt,fPnt)
    USE tooluser_fd
    INTEGER,          INTENT( IN      ) :: CallerID  !USER ID tag
    INTEGER,          INTENT( IN      ) :: grdID    !SAG grid ID
    TYPE( HPACPlotTypeT ), INTENT( IN      ) :: PlotType !Plot definition
    REAL,          INTENT( IN      ) :: xPnt      !X location
    REAL,          INTENT( IN      ) :: yPnt      !Y location
    REAL,          INTENT( OUT     ) :: fPnt      !Field value
    END FUNCTION HPACGetFieldValue

```

```

END INTERFACE

INTERFACE
  INTEGER FUNCTION HPACGetFieldValues(CallerID,grdID,PlotType,nPnt,xPnt,yPnt,fPnt)
    USE tooluser_fd
    INTEGER,           INTENT( IN ) :: CallerID      !USER ID tag
    INTEGER,           INTENT( IN ) :: grdID        !SAG grid ID
    TYPE( HPACPlotTypeT ), INTENT( IN ) :: PlotType   !Plot definition
    INTEGER,           INTENT( IN ) :: nPnt         !X location
    REAL, DIMENSION(nPnt), INTENT( IN ) :: xPnt       !X location
    REAL, DIMENSION(nPnt), INTENT( IN ) :: yPnt       !Y location
    REAL, DIMENSION(nPnt), INTENT( OUT ) :: fPnt      !Field value
  END FUNCTION HPACGetFieldValues
END INTERFACE

INTERFACE
  INTEGER FUNCTION HPACGetLastError(error)
    USE tooluser_fd
    TYPE( messageT ), INTENT( OUT ) :: error          !error message
  END FUNCTION HPACGetLastError
END INTERFACE

INTERFACE
  INTEGER FUNCTION HPACGetPlotClasses( &
    CallerID,Project,ClassStr,ChoiceStr,KindStr, &
    CatClassArray,ClassChoiceArray,ProjectCoordinate &
  )
    USE tooluser_fd
    INTEGER,           INTENT( IN ) :: CallerID
    !USER ID tag
    TYPE( projectIDT ),           INTENT( IN ) :: Project
    !Project ID
    TYPE( char64T ),             DIMENSION(*), INTENT( OUT ) :: ClassStr
    !Class strings
    TYPE( char64T ),             DIMENSION(*), INTENT( OUT ) :: ChoiceStr
    !Choice strings
    TYPE( char64T ),             DIMENSION(*), INTENT( OUT ) :: KindStr
    !Kind strings
    TYPE( HPACCCategoryClassT ), &
    DIMENSION(HP_NUMCAT,*),INTENT( OUT ) :: CatClassArray
    !Class/Category use array
    TYPE( HPACCClassChoiceT ),    DIMENSION(*), INTENT( OUT ) :: ClassChoiceArray
    !Class/Choice use array
    TYPE( HPACFieldCoordinateT ), INTENT( OUT ) :: ProjectCoordinate
    !Project coordinate descriptor
  END FUNCTION HPACGetPlotClasses
END INTERFACE

INTERFACE
  INTEGER FUNCTION HPACGetPlotTimes(CallerID,Project,TimePuff,TimeSrf,TimeMet)
    USE tooluser_fd

```

```

INTEGER, INTENT( IN ) :: CallerID !USER ID tag
TYPE( projectIDT ), INTENT( IN ) :: Project !Project ID
TYPE( HPACTimeT), DIMENSION(*), INTENT( OUT ) :: TimePuff !Puff times
TYPE( HPACTimeT), DIMENSION(*), INTENT( OUT ) :: TimeSrf
    !Surface grid times
TYPE( HPACTimeT), DIMENSION(*), INTENT( OUT ) :: TimeMet !Met times
END FUNCTION HPACGetPlotTimes
END INTERFACE

INTERFACE
    INTEGER FUNCTION HPACGetProjectAudit(CallerID,audit)
        USE tooluser_fd
        INTEGER, INTENT( IN ) :: CallerID !USER ID Tag
        TYPE( pauditT ), INTENT( INOUT ) :: audit !Project ID
    END FUNCTION HPACGetProjectAudit
END INTERFACE

INTERFACE
    INTEGER FUNCTION HPACGetProjectTerrain(CallerID,terrain,ths,tddx,tddy)
        USE tooluser_fd
        INTEGER, INTENT( IN ) :: CallerID !USER ID Tag
        TYPE( pterrainHeadT ), INTENT( INOUT ) :: terrain !Project ID
        REAL, DIMENSION(*), INTENT( OUT ) :: ths !Terrain array
        REAL, DIMENSION(*), INTENT( OUT ) :: tddx !Slope array
        REAL, DIMENSION(*), INTENT( OUT ) :: tddy !Slope array
    END FUNCTION HPACGetProjectTerrain
END INTERFACE

INTERFACE
    INTEGER FUNCTION HPACGetProjectTerrainHeader(CallerID,terrain,ths,tddx,tddy)
        USE tooluser_fd
        INTEGER, INTENT( IN ) :: CallerID !USER ID Tag
        TYPE( pterrainHeadT ), INTENT( INOUT ) :: terrain !Project ID
        REAL, DIMENSION(*), INTENT( OUT ) :: ths !Terrain array
        REAL, DIMENSION(*), INTENT( OUT ) :: tddx !Slope array
        REAL, DIMENSION(*), INTENT( OUT ) :: tddy !Slope array
    END FUNCTION HPACGetProjectTerrainHeader
END INTERFACE

INTERFACE
    INTEGER FUNCTION HPACGetProjectPuff(CallerID,puff,timeID, puffList, auxData)
        USE tooluser_fd
        INTEGER, INTENT( IN ) :: CallerID !USER ID Tag
        TYPE( ppuffHeadT ), INTENT( INOUT ) :: puff !Project ID
        INTEGER, INTENT( IN ) :: timeID !time ID
        TYPE( puffT ), DIMENSION(*), INTENT( OUT ) :: puffList !puff data
        REAL, DIMENSION(*), INTENT( OUT ) :: auxData !puff auxiliary data
    END FUNCTION HPACGetProjectPuff
END INTERFACE

INTERFACE
    INTEGER FUNCTION HPACGetProjectPuffHeader(CallerID,puff,timeID)
        USE tooluser_fd

```

```

    INTEGER,           INTENT( IN      ) :: CallerID      !USER ID Tag
    TYPE( ppuffHeadT ), INTENT( INOUT ) :: puff          !Project ID
    INTEGER,           INTENT( IN      ) :: timeID        !time ID
END FUNCTION HPACGetProjectPuffHeader
END INTERFACE

INTERFACE
    INTEGER FUNCTION HPACGetProjectVersion(CallerID,project)
        USE tooluser_fd
        INTEGER,           INTENT( IN      ) :: CallerID      !USER ID Tag
        TYPE( projectIDT ), INTENT( INOUT ) :: project       !Project ID
    END FUNCTION HPACGetProjectVersion
END INTERFACE

INTERFACE
    INTEGER FUNCTION HPACGetSCIPUFFVersion( )
    END FUNCTION HPACGetSCIPUFFVersion
END INTERFACE

INTERFACE
    INTEGER FUNCTION HPACGetSubstrates( substrate )
        USE tooluser_fd
        TYPE( char16T ), DIMENSION(*), INTENT( OUT ) :: substrate
    END FUNCTION
END INTERFACE

INTERFACE
    INTEGER FUNCTION HPACGetVersion( )
    END FUNCTION HPACGetVersion
END INTERFACE

INTERFACE
    INTEGER FUNCTION HPACGetString(iflag,string)
        USE tooluser_fd
        INTEGER,           INTENT( IN      ) :: iflag         !request flag
        TYPE( char128T ), INTENT( OUT ) :: string         !version string
    END FUNCTION HPACGetString
END INTERFACE

INTERFACE
    SUBROUTINE HPACInitError( )
    END SUBROUTINE HPACInitError
END INTERFACE

INTERFACE
    INTEGER FUNCTION HPACInitTool(CallerID,callback,request,limit,cINIfile)
        USE tooluser_fd
        INTEGER, INTENT( IN      ) :: CallerID !USER ID tag
        INTEGER, INTENT( IN      ) :: callback !Address of user callback function
        INTEGER, INTENT( INOUT ) :: request      !Initialization request
        TYPE( limitT ),   INTENT( IN      ) :: limit       !Array size limits
        TYPE( char128T ), INTENT( IN      ) :: cINIfile !INI file
    END FUNCTION HPACInitTool

```

```

END INTERFACE

INTERFACE
  INTEGER FUNCTION HPACLoadProject(CallerID,project,mtlList,relList)
    USE tooluser_fd
    INTEGER,                                     INTENT( IN     ) :: CallerID !USER ID Tag
    TYPE( projectT ),                           INTENT( INOUT ) :: project !Project ID
    TYPE( materialT ),DIMENSION(*), INTENT( OUT    ) :: mtlList !Material list
    TYPE( releaseT ), DIMENSION(*), INTENT( OUT    ) :: relList !Release list
  END FUNCTION HPACLoadProject
END INTERFACE

INTERFACE
  INTEGER FUNCTION HPACModifySensor(sensor)
    USE tooluser_fd
    TYPE( sensorT ), INTENT( IN     ) :: sensor      !New sensor position
  END FUNCTION HPACModifySensor
END INTERFACE

INTERFACE
  INTEGER FUNCTION HPACGetConcentration(conc)
    USE tooluser_fd
    TYPE( InteractiveSensor ), INTENT( INOUT ) :: conc !Interactive sensor
  END FUNCTION HPACGetConcentration
END INTERFACE

INTERFACE
  INTEGER FUNCTION HPACGetLOS(los)
    USE tooluser_fd
    TYPE( InteractiveSensor ), INTENT( INOUT ) :: los !Interactive sensor
  END FUNCTION HPACGetLOS
END INTERFACE

INTERFACE
  INTEGER FUNCTION HPACGetSrfPoint(srf)
    USE tooluser_fd
    TYPE( InteractiveSrf ), INTENT( INOUT ) :: srf !Interactive sensor
  END FUNCTION HPACGetSrfPoint
END INTERFACE

INTERFACE
  INTEGER FUNCTION HPACNewProject(CallerID,project,mtlList,relList)
    USE tooluser_fd
    INTEGER,                                     INTENT( IN     ) :: CallerID !USER ID Tag
    TYPE( createNewT ),                           INTENT( IN     ) :: project !Project ID
    TYPE( materialT ),DIMENSION(*), INTENT( OUT    ) :: mtlList !Material list
    TYPE( releaseT ), DIMENSION(*), INTENT( OUT    ) :: relList !Release list
  END FUNCTION HPACNewProject
END INTERFACE

INTERFACE
  INTEGER FUNCTION HPACNumPlotClasses(CallerID,Project,nClass,nChoice,nKind)

```

```

USE tooluser_fd
INTEGER,           INTENT( IN ) :: CallerID !USER ID tag
TYPE( projectIDT ), INTENT( IN ) :: Project !Project ID
INTEGER,           INTENT( OUT ) :: nClass   !Number of Class strings
INTEGER,           INTENT( OUT ) :: nChoice  !Number of Choice strings
INTEGER,           INTENT( OUT ) :: nKind    !Number of Kind strings
END FUNCTION HPACNumPlotClasses
END INTERFACE

INTERFACE
  INTEGER FUNCTION
  HPACNumPlotTimes(CallerID,Project,nTimePuff,nTimeSrf,nTimeMet,nTimeRad)
    USE tooluser_fd
    INTEGER, INTENT( IN ) :: CallerID !USER ID tag
    TYPE( projectIDT ), INTENT( IN ) :: Project !Project ID
    INTEGER, INTENT( OUT ) :: nTimePuff !Number of Puff times
    INTEGER, INTENT( OUT ) :: nTimeSrf  !Number of Surface grid times
    INTEGER, INTENT( OUT ) :: nTimeMet  !Number of Met times
    INTEGER, INTENT( OUT ) :: nTimeRad
      !Number of NFAC Radiation surface grid sub-times
  END FUNCTION
END INTERFACE

INTERFACE
  INTEGER FUNCTION HPACNumSubstrates( )
  END FUNCTION
END INTERFACE

INTERFACE
  INTEGER FUNCTION
  HPACPopAreaField( &
    CallerID,grdID,Field,PlotType,contourHead,contourList &
  )
    USE tooluser_fd
    INTEGER,           INTENT( IN ) :: CallerID      !USER ID tag
    INTEGER,           INTENT( IN ) :: grdID        !SAG grid ID
    TYPE( HPACPlotFieldT ), INTENT( IN ) :: Field       !Field descriptor
    TYPE( HPACPlotTypeT ), INTENT( IN ) :: PlotType     !Plot definition
    TYPE( HPACCContourHeaderT ), INTENT( IN ) :: contourHead
      !Contour array header
    TYPE( HPACCContourElementT ), DIMENSION(*), INTENT( INOUT ) :: contourList
      !Contour array
  END FUNCTION HPACPopAreaField
END INTERFACE

INTERFACE
  INTEGER FUNCTION HPACQuerySensor(sensor)
    USE tooluser_fd
    TYPE( sensorT ), INTENT( INOUT ) :: sensor          !Current sensor data
  END FUNCTION HPACQuerySensor
END INTERFACE

INTERFACE

```

```

INTEGER FUNCTION HPACRestartProject(CallerID,project)
  USE tooluser_fd
  INTEGER,           INTENT( IN ) :: CallerID      !USER ID Tag
  TYPE( createRstT ), INTENT( IN ) :: project      !Project input
END FUNCTION HPACRestartProject
END INTERFACE

INTERFACE
  INTEGER FUNCTION HPACRunProject(CallerID,run)
    USE tooluser_fd
    INTEGER,           INTENT( IN ) :: CallerID      !USER ID Tag
    TYPE( pendT ),   INTENT( IN ) :: run            !Run input
  END FUNCTION HPACRunProject
END INTERFACE

INTERFACE
  INTEGER FUNCTION HPACSizeProject(CallerID,project,nMtl,nRel)
    USE tooluser_fd
    INTEGER,           INTENT( IN ) :: CallerID      !USER ID tag
    TYPE( projectIdT ), INTENT( IN ) :: project      !Project ID
    INTEGER,           INTENT( OUT ) :: nMtl          !number of materials in file
    INTEGER,           INTENT( OUT ) :: nRel          !number of releases in file
  END FUNCTION HPACSizeProject
END INTERFACE

INTERFACE
  INTEGER FUNCTION HPACTransform(Cin,Cout,np,xp,yp)
    USE tooluser_fd
    TYPE( HPACFieldCoordinateT ), INTENT( IN ) :: Cin  !Input Coordinate data
    TYPE( HPACFieldCoordinateT ), INTENT( IN ) :: Cout !Output Coordinate data
    INTEGER,           INTENT( IN ) :: np             !number of points
    REAL,             DIMENSION(np),  INTENT( INOUT ) :: xp   !x coordinate arrray
    REAL,             DIMENSION(np),  INTENT( INOUT ) :: yp   !y coordinate arrray
  END FUNCTION HPACTransform
END INTERFACE

INTERFACE
  INTEGER FUNCTION HPACTransformPt(Cin,Cout,np,pt)
    USE tooluser_fd
    TYPE( HPACFieldCoordinateT ), INTENT( IN ) :: Cin  !Input Coordinate data
    TYPE( HPACFieldCoordinateT ), INTENT( IN ) :: Cout !Output Coordinate data
    INTEGER,           INTENT( IN ) :: np             !number of points
    TYPE( HPACPointT ),  DIMENSION(np), INTENT( INOUT ) :: pt   !pt arrray
  END FUNCTION HPACTransformPt
END INTERFACE

INTERFACE
  INTEGER FUNCTION HPACWriteField( &
    CallerID,grdID,Field,PlotType,contourHead,contourList, &
    GUIWrite,nComment,Comment &
  )
    USE tooluser_fd

```

```

INTEGER,                      INTENT( IN      ) :: CallerID      !USER ID tag
INTEGER,                      INTENT( IN      ) :: grdID        !SAG grid ID
TYPE( HPACPlotFieldT ),       INTENT( IN      ) :: Field
    !Field descriptor
TYPE( HPACPlotTypeT ),        INTENT( IN      ) :: PlotType
    !Plot definition
TYPE( HPACCContourHeaderT ),  INTENT( IN      ) :: contourHead
    !Contour array header
TYPE( HPACCContourElementT ), DIMENSION(*), INTENT( INOUT ) :: contourList
    !Contour array
TYPE( ARAPWriteT ),           INTENT( IN      ) :: GUIWrite
    !Draw instructions
INTEGER,                      INTENT( IN      ) :: nComment
    !Number of User supplied comments
TYPE( char128T ),            DIMENSION(nComment), INTENT( IN ) :: Comment
    !User supplied comments
END FUNCTION HPACWriteField
END INTERFACE

END MODULE HPACtool

```

2.1.11 inpstruct_fd.f90

```

MODULE inpstruct_fd
    USE flags_fd          !Basic flags structures
    USE options_fd         !Basic options structures
    USE list_fd            !Basic list structures
    USE prjstruct_fd       !Basic project structures
    USE time_fd            !Basic time structures

!===== pauditT =====
    TYPE pauditT
        SEQUENCE
        TYPE ( projectIDT ) project
        TYPE ( auditT      ) audit
    END TYPE pauditT

!===== pflagT =====
    TYPE pflagsT
        SEQUENCE
        TYPE ( projectIDT ) project
        TYPE ( flagST      ) flags
    END TYPE pflagsT

!===== poptionsT =====
    TYPE poptionsT
        SEQUENCE
        TYPE ( projectIDT ) project
        TYPE ( optionsT     ) options

```

```

END TYPE poptionsT

!===== statusT =====

TYPE statusT
SEQUENCE
INTEGER      status
INTEGER      nPuffType
TYPE ( timeT ) time
END TYPE statusT

END MODULE inpstruct_fd

```

2.1.12 list_fd.f90

```

MODULE list_fd

!===== listT =====

TYPE listHeadT
SEQUENCE
INTEGER  max
INTEGER  number
END TYPE listHeadT

END MODULE list_fd

```

2.1.13 material_fd.f90

```

MODULE material_fd

!===== HPAC Material types =====

INTEGER,PARAMETER :: HMB_GAS          = 0
INTEGER,PARAMETER :: HMB_PARTICLE     = 1
!DEC$ IF DEFINED (LIQUID)
    INTEGER,PARAMETER :: HMB_LIQUID     = 2
!DEC$ ENDIF
!DEC$ IF DEFINED (AEROSOL)
    INTEGER,PARAMETER :: HMB_AEROSOL   = 3
!DEC$ ENDIF
!DEC$ IF DEFINED (LIQUID)
    INTEGER,PARAMETER :: HMB_WETPARTICLE = 4
    INTEGER,PARAMETER :: HMB_2NDEVAP     = 20
!DEC$ ENDIF
!DEC$ IF DEFINED (NWPN)
    INTEGER,PARAMETER :: HMB_NWPN       = 21
!DEC$ ENDIF
!DEC$ IF DEFINED (NFAC)
    INTEGER,PARAMETER :: HMB_NFAC       = 22
!DEC$ ENDIF
!DEC$ IF DEFINED (HAZARD)
    INTEGER,PARAMETER :: HMB_HAZARD     = 23

```

```

!DEC$ ENDIF
!DEC$ IF DEFINED (MULTCOMP)
  INTEGER,PARAMETER :: HMB_MULTICOMP      = 24
!DEC$ ENDIF

  INTEGER,PARAMETER :: HMB_BASIC          = 10

  INTEGER,PARAMETER :: HM_GAS             = 2**HMB_GAS
  INTEGER,PARAMETER :: HM_PARTICLE        = 2**HMB_PARTICLE
!DEC$ IF DEFINED (LIQUID)
  INTEGER,PARAMETER :: HM_liquid          = 2**HMB_LIQUID
!DEC$ ENDIF
!DEC$ IF DEFINED (AEROSOL)
  INTEGER,PARAMETER :: HM_AEROSOL         = 2**HMB_AEROSOL
!DEC$ ENDIF
!DEC$ IF DEFINED (LIQUID)
  INTEGER,PARAMETER :: HM_WETPARTICLE    = 2**HMB_WETPARTICLE
  INTEGER,PARAMETER :: HM_2NDEVAP          = 2**HMB_2NDEVAP
!DEC$ ENDIF
!DEC$ IF DEFINED (NWPN)
  INTEGER,PARAMETER :: HM_NWPN            = 2**HMB_NWPN
!DEC$ ENDIF
!DEC$ IF DEFINED (NFAC)
  INTEGER,PARAMETER :: HM_NFAC            = 2**HMB_NFAC
!DEC$ ENDIF
!DEC$ IF DEFINED (HAZARD)
  INTEGER,PARAMETER :: HM_HAZARD          = 2**HMB_HAZARD
!DEC$ ENDIF
!DEC$ IF DEFINED (MULTCOMP)
  INTEGER,PARAMETER :: HM_MULTICOMP        = 2**HMB_MULTICOMP
!DEC$ ENDIF

!==== HPAC Surface file modes =====
  INTEGER,PARAMETER :: HSB_GROUPDEP      = 0
  INTEGER,PARAMETER :: HSB_GROUPDOS      = 1
  INTEGER,PARAMETER :: HSB_TOTALDEP      = 2
  INTEGER,PARAMETER :: HSB_TOTALDOS      = 3
  INTEGER,PARAMETER :: HS_NONE           = 0
  INTEGER,PARAMETER :: HS_GROUPDEP       = 2**HSB_GROUPDEP
  INTEGER,PARAMETER :: HS_GROUPDOS       = 2**HSB_GROUPDOS
  INTEGER,PARAMETER :: HS_TOTALDEP       = 2**HSB_TOTALDEP
  INTEGER,PARAMETER :: HS_TOTALDOS       = 2**HSB_TOTALDOS

!==== parameters =====
  INTEGER,PARAMETER :: HS_MAXMTLRESERVED = 2
  INTEGER,PARAMETER :: HS_MAXMTLBINSIZE   = 50
  INTEGER,PARAMETER :: HS_MAXMTLBASIC     = 17
  INTEGER,PARAMETER :: HS_MAXMTLGAS       = 7
!DEC$ IF DEFINED (AEROSOL)
  INTEGER,PARAMETER :: HS_MAXMTLAEROSOL   = 13
!DEC$ ENDIF

```

```

!DEC$ IF DEFINED (LIQUID)
  INTEGER,PARAMETER :: HS_MAXMTLLIQUID = 17
!DEC$ ENDIF
  INTEGER,PARAMETER :: HS_MAXMTLPARTICLE = 7
  INTEGER,PARAMETER :: HS_PADMTLGEN = HS_MAXMTLBASIC + HS_MAXMTLRESERVED &
+ 2*HS_MAXMTLBINSIZE + 1
  INTEGER,PARAMETER :: HS_PADMTLGAS = HS_MAXMTLBASIC - HS_MAXMTLGAS &
+ HS_MAXMTLRESERVED &
+ 2*HS_MAXMTLBINSIZE + 1

!DEC$ IF DEFINED (AEROSOL)
  INTEGER,PARAMETER :: HS_PADMTLAEROSOL = HS_MAXMTLBASIC - HS_MAXMTLAEROSOL &
+ HS_MAXMTLRESERVED &
+ 2*HS_MAXMTLBINSIZE + 1

!DEC$ ENDIF
!DEC$ IF DEFINED (LIQUID)
  INTEGER,PARAMETER :: HS_PADMTLLIQUID = HS_MAXMTLBASIC - HS_MAXMTLLIQUID &
+ HS_MAXMTLRESERVED
!DEC$ ENDIF
  INTEGER,PARAMETER :: HS_PADMTLPARTICLE = HS_MAXMTLBASIC - HS_MAXMTLPARTICLE &
+ HS_MAXMTLRESERVED

===== matGenT =====

TYPE matGenT
  SEQUENCE
    INTEGER padding(HS_PADMTLGEN)
END TYPE matGenT

===== matGasT =====

TYPE matGasT
  SEQUENCE
    REAL minConcentration
    REAL decayAmp
    REAL decayMin
!DEC$ IF DEFINED (NWPN)
    REAL NWPNdecay
!DEC$ ELSE
    REAL rNotUsed
!DEC$ ENDIF
    INTEGER save
    REAL gasDensity
    REAL gasDeposition
    INTEGER padding(HS_PADMTLGAS)
END TYPE matGasT

!DEC$ IF DEFINED (AEROSOL)
===== matAerosolt =====

TYPE matAerosolt
  SEQUENCE
    REAL minConcentration
    REAL decayAmp

```

```

      REAL      decayMin
!DEC$ IF DEFINED (NWPN)
      REAL      NWPNdecay
!DEC$ ELSE
      REAL      rNotUsed
!DEC$ ENDIF
      INTEGER save
      REAL      gasDeposition
      REAL      liquidDensity
      REAL      antoine(3)
      REAL      molWeight
      REAL      liqSpecificHeat
      REAL      gasSpecificHeat
      INTEGER padding(HS_PADMTLAEROSOL)
END TYPE matAerosolt

!DEC$ ENDIF
!DEC$ IF DEFINED (LIQUID)
===== matLiquidT =====

TYPE matLiquidT
SEQUENCE
      REAL      minConcentration
      REAL      decayAmp
      REAL      decayMin
!DEC$ IF DEFINED (NWPN)
      REAL      NWPNdecay
!DEC$ ELSE
      REAL      rNotUsed
!DEC$ ENDIF
      INTEGER save
      REAL      gasDensity
      REAL      gasDeposition
      REAL      liquidDensity(2)
      REAL      antoine(3)
      REAL      molWeight
      REAL      srfTension
      REAL      spreadFactor
      REAL      viscosity
      INTEGER nSizeBins
      REAL      binSize(HS_MAXMTLBINSIZE)
      REAL      binBounds(HS_MAXMTLBINSIZE+1)
      INTEGER padding(HS_PADMTLLIQUID)
END TYPE matLiquidT

!DEC$ ENDIF
===== matParticleT =====

TYPE matParticleT
SEQUENCE
      REAL      minConcentration
      REAL      decayAmp
      REAL      decayMin

```

```

!DEC$ IF DEFINED (NWPN)
    REAL      NWPNdecay
!DEC$ ELSE
    REAL      rNotUsed
!DEC$ ENDIF
    INTEGER  save
    REAL     density
    INTEGER nSizeBins
    REAL     binSize(HS_MAXMTLBINSIZE)
    REAL     binBounds(HS_MAXMTLBINSIZE+1)
    INTEGER padding(HS_PADMTLPARTICLE)
END TYPE matParticleT

!===== materialT =====
TYPE materialT
SEQUENCE
    INTEGER          type          !class->matl.ccls
    INTEGER          puffIndex    !iofp
!DEC$ IF DEFINED (FXPLOT)
    INTEGER          effectClass !HPAC 4.0
    INTEGER          effectAvail !HPAC 4.0
!DEC$ ELSE
    INTEGER, DIMENSION(2) :: iNotUsed
!DEC$ ENDIF
    TYPE ( matGenT )   matData    !Class specific parameters
    CHARACTER(16)     name        !mname->matl.cmat
    CHARACTER(16)     units       !units->matl.unit
    CHARACTER(64)     file        !file_name->matl.file
    CHARACTER(128)    path        !path_name->matl.path
END TYPE materialT

END MODULE material_fd

```

2.1.14 message_fd.f90

```

MODULE message_fd

!===== messageT =====
TYPE messageT
SEQUENCE
    INTEGER          iParm
    INTEGER          jParm
    CHARACTER(128)  aString
    CHARACTER(128)  bString
    CHARACTER(128)  cString
    CHARACTER(80)   routine
END TYPE messageT

END MODULE message_fd

```

2.1.15 metstruct_fd.f90

```
MODULE metstruct_fd
  USE weather_fd      !Basic weather structures
  USE prjstruct_fd    !Basic project structures

  !==== pweatherT

  TYPE pweatherT
    SEQUENCE
      TYPE( projectIDT ) project
      TYPE( weatherT ) weather
    END TYPE pweatherT

END MODULE metstruct_fd
```

2.1.16 msgstruct_fd.f90

```
MODULE msgstruct_fd
  USE message_fd      !Basic message structures
END MODULE msgstruct_fd
```

2.1.17 mtlstruct_fd.f90

```
MODULE mtlstruct_fd
  USE material_fd     !Basic material structures
  USE prjstruct_fd    !Basic project structures
  USE list_fd         !Basic list structures

  !==== mtlControlT =====

  TYPE mtlControlT
    SEQUENCE
      INTEGER          mode
      CHARACTER(8)     fileExtension
      CHARACTER(16)    searchName
    END TYPE mtlControlT

  !==== pmaterialT =====

  TYPE pmaterialT
    SEQUENCE
      TYPE( projectIDT ) project
      TYPE( listHeadT ) mtlHead
      TYPE( mtlControlT ) control
    END TYPE pmaterialT

END MODULE mtlstruct_fd
```

2.1.18 NWPNeffect_fd.f90

```
!ARAP$ IF DEFINED (NWPN)
MODULE NWPNeffect_fd
```

```

!DEC$ IF DEFINED (NWPN)

USE structure_fd

!---- effect ID

!  INTEGER, PARAMETER :: EID_CASUALTY_PROMPT = 4

!---- request input values

!  INTEGER, PARAMETER :: EID_INIT = 1
!  INTEGER, PARAMETER :: EID_COMP = 2
!  INTEGER, PARAMETER :: EID_EXIT = 3
!  INTEGER, PARAMETER :: EID_COMP_NOCEP = 4

!---- storage dimensions

INTEGER, PARAMETER :: CP_MAX_RADII      = 23
INTEGER, PARAMETER :: CP_MAX_PROTECT    = 10 !Arrays are 0:CP_MAX_PROTECT
INTEGER, PARAMETER :: CP_MAX_SETTINGS   = 2

!  INTEGER, PARAMETER :: URBAN = 1
!  INTEGER, PARAMETER :: RURAL = 2

!---- Prompt Casualty structures

TYPE casualtyPromptInitInT
  SEQUENCE
    INTEGER :: nProtect      !no. of protection types
    TYPE (char32T), DIMENSION(0:CP_MAX_PROTECT) :: ProtTypes
      !array of protection types
  END TYPE casualtyPromptInitInT

TYPE casualtyPromptInitOutT
  SEQUENCE
    INTEGER nWpn                !no. of weapons
    REAL, DIMENSION(CP_MAX_RADII) :: probFac !probability values
  END TYPE casualtyPromptInitOutT

TYPE casualtyPromptCompInT
  SEQUENCE
    INTEGER :: nWpn      !Size of casualtyPromptCompOutT list
  END TYPE casualtyPromptCompInT

TYPE casualtyPromptCompOutT
  SEQUENCE
    REAL, DIMENSION(3) :: loc      !(x,y,z) location
    REAL :: time        !release time
    REAL :: pa          !probability of arrival
    REAL :: cep          !weapon CEP
    REAL, DIMENSION(0:CP_MAX_PROTECT,CP_MAX_RADII) :: rCasualty !radial points
    REAL, DIMENSION(0:CP_MAX_PROTECT,CP_MAX_RADII) :: rFatality !radial points
    REAL, DIMENSION(0:CP_MAX_PROTECT) :: wrCasualty !range (km)

```

```

      REAL, DIMENSION(0:CP_MAX_PROTECT)          :: wrFatality !range (km)
END TYPE casualtyPromptCompOutT

!DEC$ ENDIF
END MODULE NWPNeffect_fd
!ARAP$ ENDIF

```

2.1.19 options_fd.f90

```

MODULE options_fd

!==== optionsT =====
TYPE optionsT
  SEQUENCE
    INTEGER      nzBL
    INTEGER      mGrd
    INTEGER      substrate
    REAL         timeAvg
    REAL         massMin
    REAL         delMin
    REAL         wwTrop
    REAL         epsTrop
    REAL         slTrop
    REAL         uuCalm
    REAL         slCalm
    REAL         zDosage
    REAL         dtSampler
    CHARACTER(64) samplerFile
    CHARACTER(128) samplerPath
  END TYPE optionsT

END MODULE options_fd

```

2.1.20 plotfx_fd.f90

```

MODULE plotfx_fd

INTEGER, PARAMETER :: FX_NFAC      = 1
INTEGER, PARAMETER :: FX_NWPN       = 2
INTEGER, PARAMETER :: FX_NWPNCAS   = 3
INTEGER, PARAMETER :: FX_NWPNFAT   = 4
INTEGER, PARAMETER :: FX_NWPNTAB   = 5
INTEGER, PARAMETER :: FX_TOXL      = 6
INTEGER, PARAMETER :: FX_FXTOOL   = 7
!DEC$ IF DEFINED (GULFWAR)
INTEGER, PARAMETER :: FX_SRFPRBMAX = 8      !Max prob from sfc dosage field
INTEGER, PARAMETER :: FX_SRFCPRBMX = 9      !Max prob from sfc conc field
!DEC$ ENDIF

INTEGER, PARAMETER :: MAX_PROTECTION_TYPES = 6
INTEGER, PARAMETER :: nORNL = MAX_PROTECTION_TYPES+1

```

```
END MODULE plotfx_fd
```

2.1.21 plotlist_fd.f90

```
MODULE plotlist_fd

  USE time_fd

!===== HPACClassChoiceT =====

  TYPE HPACClassChoiceT
    INTEGER available
    INTEGER kind          ! kind subplots available
    INTEGER ikind         ! points to kind string list
    INTEGER nkind         ! no. of kinds available
    INTEGER itime         ! time list id
    INTEGER usertime      ! user-specified time available
    CHARACTER(16) units   ! units string
  END TYPE HPACClassChoiceT

!===== HPACCATEGORYClassT =====

  TYPE HPACCATEGORYClassT
    INTEGER available
    INTEGER type          ! type subplot available
  END TYPE HPACCATEGORYClassT

!===== HPACTimeT =====

  TYPE HPACTimeT
    SEQUENCE
    TYPE( TimeT ) time
    INTEGER(4)  nItems
    CHARACTER(24) string
  END TYPE HPACTimeT

END MODULE plotlist_fd
```

2.1.22 poparea_fd.f90

```
MODULE poparea_fd

  INTEGER, PARAMETER :: POP_BON      = 0
  INTEGER, PARAMETER :: POP_BAREA     = 1
  INTEGER, PARAMETER :: POP_BEXPECT   = 2
  INTEGER, PARAMETER :: POP_OFF       = 0
  INTEGER, PARAMETER :: POP_ON        = 2**POP_BON      ! (1)
  INTEGER, PARAMETER :: POP_AREA      = 2**POP_BAREA     ! (2)
  INTEGER, PARAMETER :: POP_EXPECT    = 2**POP_BEXPECT   ! (4)

END MODULE poparea_fd
```

2.1.23 prjstruct_fd.f90

```
MODULE prjstruct_fd

!==== projectIDT =====

TYPE projectIDT
  SEQUENCE
    INTEGER          ID
    INTEGER          version
    CHARACTER(64)   name
    CHARACTER(128)  path
  END TYPE projectIDT

END MODULE prjstruct_fd
```

2.1.24 release_fd.f90

```
MODULE release_fd

!==== HPAC Release types =====

INTEGER, PARAMETER :: HRB_INST = 0
INTEGER, PARAMETER :: HRB_CONT = 1
!DEC$ IF DEFINED (INTERACTIVE)
  INTEGER, PARAMETER :: HRB_INTER = 2
!DEC$ ENDIF
  INTEGER, PARAMETER :: HRB_FILE = 20
  INTEGER, PARAMETER :: HRB_MOVE = 21
!DEC$ IF DEFINED (LIQUID)
  INTEGER, PARAMETER :: HRB_POOL = 22
!DEC$ ENDIF
  INTEGER, PARAMETER :: HRB_STACK = 23
  INTEGER, PARAMETER :: HRB_PUFF = 24
  INTEGER, PARAMETER :: HR_INST = 2**HRB_INST
  INTEGER, PARAMETER :: HR_CONT = 2**HRB_CONT
  INTEGER, PARAMETER :: HR_FILE = 2**HRB_FILE + HR_INST
  INTEGER, PARAMETER :: HR_MOVE = 2**HRB_MOVE + HR_CONT
!DEC$ IF DEFINED (LIQUID)
  INTEGER, PARAMETER :: HR_POOL = 2**HRB_POOL + HR_CONT
!DEC$ ENDIF
  INTEGER, PARAMETER :: HR_PUFF = 2**HRB_PUFF + HR_INST
  INTEGER, PARAMETER :: HR_STACK = 2**HRB_STACK + HR_CONT
!DEC$ IF DEFINED (INTERACTIVE)
  INTEGER, PARAMETER :: HR_INTER = 2**HRB_INTER
!DEC$ ENDIF

!==== HPAC Release distribution types =====

INTEGER, PARAMETER :: HDB_LOGNORM = 0
INTEGER, PARAMETER :: HD_LOGNORM = -2**HDB_LOGNORM

!==== parameters =====
```

```

INTEGER, PARAMETER :: HS_MAXRELRESERVED = 1
INTEGER, PARAMETER :: HS_MAXRELMAXSIZE = 51

INTEGER, PARAMETER :: HS_MAXRELCONTSIZE = 10
INTEGER, PARAMETER :: HS_MAXRELFILSIZE = 51
INTEGER, PARAMETER :: HS_MAXRELINSTSIZE = 13
INTEGER, PARAMETER :: HS_MAXRELCMOVSIZE = 13
!DEC$ IF DEFINED (LIQUID)
    INTEGER, PARAMETER :: HS_MAXRELPOOLSIZE = 3
!DEC$ ENDIF
    INTEGER, PARAMETER :: HS_MAXRELIPUFSIZE = 19
    INTEGER, PARAMETER :: HS_MAXRELCSTKSIZE = 8

    INTEGER, PARAMETER :: HS_PADRELGEN      = HS_MAXRELMAXSIZE + HS_MAXRELRESERVED
    INTEGER, PARAMETER :: HS_PADRELCONT     = HS_PADRELGEN - HS_MAXRELCONTSIZE
    INTEGER, PARAMETER :: HS_PADRELFIL      = HS_PADRELGEN - HS_MAXRELFILSIZE
    INTEGER, PARAMETER :: HS_PADRELINST     = HS_PADRELGEN - HS_MAXRELINSTSIZE
    INTEGER, PARAMETER :: HS_PADRELCMOV      = HS_PADRELGEN - HS_MAXRELCMOVSIZE
!DEC$ IF DEFINED (LIQUID)
    INTEGER, PARAMETER :: HS_PADRELPOOL     = HS_PADRELGEN - HS_MAXRELPOOLSIZE
!DEC$ ENDIF
    INTEGER, PARAMETER :: HS_PADRELIPUF     = HS_PADRELGEN - HS_MAXRELIPUFSIZE
    INTEGER, PARAMETER :: HS_PADRELCSTK     = HS_PADRELGEN - HS_MAXRELCSTKSIZE

==== relGenT =====

TYPE relGenT
    SEQUENCE
        INTEGER padding(HS_PADRELGEN)
    END TYPE relGenT

==== relContT =====

TYPE relContT
    SEQUENCE
        INTEGER distribution !subgroup
        REAL rate           !cmass
        REAL duration       !tdur/3600.
        REAL sigY           !sigy
        REAL sigZ           !sigz
        REAL MMD            !rel_param(REL_MMD_INDX)=lognorm_mmd
        REAL sigma          !rel_param(REL_SIGMA_INDX)=lognorm_sigma
        REAL momentum        !wmom
        REAL buoyancy        !buoy
        REAL dryFrac         !mass Fraction : Wet Particle release only
        INTEGER padding(HS_PADRELCONT)
    END TYPE relContT

==== relFileT =====

TYPE relFileT
    SEQUENCE

```

```

INTEGER      nRandom    !rel_param(REL RAND_INDX)=number_random
INTEGER      ranSeed     !rel_param(REL_SEED_INDX)=random_seed
REAL         ranSpread   !rel_param(REL_SPREAD_INDX)=random_spread
CHARACTER( 64) relFile   !name_rel
CHARACTER(128) relPath   !name_rel
INTEGER padding(HS_PADRELFILE)
END TYPE relFileT

!===== relInstT =====

TYPE relInstT
SEQUENCE
INTEGER distribution !subgroup
REAL mass          !cmass
REAL sigX          !sigx
REAL sigY          !sigy
REAL sigZ          !sigz
REAL MMD           !rel_param(REL_MMD_INDX)=lognorm_mmd
REAL sigma         !rel_param(REL_SIGMA_INDX)=lognorm_sigma
REAL momentum      !wmom
REAL buoyancy      !buoy
REAL dryFrac       !mass Fraction : Wet Particle release only
INTEGER nRandom    !rel_param(REL RAND_INDX)=number_random
INTEGER ranSeed     !rel_param(REL_SEED_INDX)=random_seed
REAL ranSpread     !rel_param(REL_SPREAD_INDX)=random_spread
INTEGER padding(HS_PADRELINST)
END TYPE relInstT

!===== relMoveT =====

TYPE relMoveT
SEQUENCE
INTEGER distribution !subgroup
REAL rate          !cmass
REAL duration      !tdur/3600.
REAL sigY          !sigy
REAL sigZ          !sigz
REAL MMD           !rel_param(REL_MMD_INDX)=lognorm_mmd
REAL sigma         !rel_param(REL_SIGMA_INDX)=lognorm_sigma
REAL momentum      !wmom
REAL buoyancy      !buoy
REAL dryFrac       !mass Fraction : Wet Particle release only
REAL velX          !urel
REAL velY          !vrel
REAL velZ          !wrel
INTEGER padding(HS_PADRELCMOV)
END TYPE relMoveT

!DEC$ IF DEFINED (LIQUID)
!===== relPoolT =====

TYPE relPoolT
SEQUENCE

```

```

REAL      mass    !cmass
REAL      sizeX  !sigx
REAL      sizeY  !sigy
INTEGER   padding(HS_PADRELPOOL)
END TYPE  relPoolT

!DEC$ ENDIF
!===== relPuffT =====

TYPE  relPuffT
SEQUENCE
INTEGER subgroup      !subgroup
REAL    mass          !cmass (c)
REAL    sxx           !Puff moments
REAL    sxy           !
REAL    sxz           !
REAL    syy           !
REAL    syz           !
REAL    szz           !
REAL    difhShear     !Small scale diffusivity (yvsc/c)
REAL    difhBuoy      !BL scale diffusivity (yvbc/c)
REAL    difhLSVxx     !Large scale diffusivities (xuc/c)
REAL    difhLSVxy     ! (xvc/c)
REAL    difhLSVyy     ! (yvc/c)
REAL    difVert        !Vertical diffusivity (zwc/c)
REAL    activeFrac    !Active fraction (cfo)
REAL    sigRatio       !SigmaC / Cbar
REAL    scaleLateral   !Lateral scale (si)
REAL    scaleStream    !Streamwise scale (si2)
REAL    scaleVert      !Vertical scale (sv)
INTEGER padding(HS_PADRELIPUF)
END TYPE  relPuffT

!===== relStackT =====

TYPE  relStackT
SEQUENCE
INTEGER distribution !subgroup
REAL    rate          !cmass
REAL    duration      !tdur/3600.
REAL    diameter      !size
REAL    MMD            !rel_param(REL_MMD_INDX)=lognorm_mmd
REAL    sigma          !rel_param(REL_SIGMA_INDX)=lognorm_sigma
REAL    exitVel        !wmom
REAL    exitTemp       !buoy
INTEGER padding(HS_PADRELCSTK)
END TYPE  relStackT

!===== releaseT =====

TYPE  releaseT
SEQUENCE
INTEGER      type      !reltyp

```

```

INTEGER          status      !HPAC 4.0 - Needs update flag
REAL             tRel        !trel/3600.
REAL             xRel        !xrel
REAL             yRel        !yrel
REAL             zRel        !zrel
REAL             horzUnc    !rel_param(REL_H_UNC_INDX)=horiz_uncertainty
REAL             vertUnc    !rel_param(REL_V_UNC_INDX)=vert_uncertainty
TYPE( relGent ) relData    !Type dependent data
CHARACTER(16)   material   !relmat
CHARACTER(32)   relName    !HPAC 4.0 Identifier
CHARACTER(48)   relDisplay !HPAC 4.0 display string
END TYPE releaseT

END MODULE release_fd

```

2.1.25 relstruct_fd.f90

```

MODULE relstruct_fd
  USE release_fd           !Basic release structures
  USE prjstruct_fd         !Basic project structures
  USE list_fd               !Basic list structures

!===== relControlT =====
  TYPE relControlT
    SEQUENCE
      INTEGER mode          !Indicates file type and search mode
      CHARACTER(32) searchID !Indicates release ID
      CHARACTER(8)  fileExtension !Indicates file extension if not SCN file
  END TYPE relControlT

!===== preleaseT =====
  TYPE preleaseT
    SEQUENCE
      TYPE( projectIDT ) project
      TYPE( listHeadT )   scnHead
      TYPE( relControlT ) control
  END TYPE preleaseT

END MODULE relstruct_fd

```

2.1.26 sensor_fd.f90

```

MODULE sensor_fd
  !----- Interactive sensor structure; used for queries or to
  !       modify sensor location or waypoints

  TYPE sensorT
    SEQUENCE
      INTEGER id            !Sensor no.
      INTEGER init          !HPACtrue -> clear waypoint list

```

```

REAL      time
REAL      x, y, z
REAL      az, el, dist
REAL      h
REAL      mean, var
END TYPE sensorT

!----- Interactive sensor structure; used to calculate concentration
!       or LOS at an arbitrary location

TYPE InteractiveSensor
SEQUENCE
CHARACTER(64) matname      !Material name
INTEGER isg                 !Subgroup no.
REAL      time                !current project time
REAL      x, y, z             !Location (project coord; z is MSL)
REAL      az, el, dist        !LOS azimuth, elevation angles & path length
REAL      mean, var
END TYPE InteractiveSensor

!----- Interactive surface field sensor structure; used to get
!       deposition or dosage at an arbitrary location

TYPE InteractiveSrf
SEQUENCE
CHARACTER(64) matname      !Material name
INTEGER isg                 !Subgroup no.
CHARACTER(64) stype         !'DEP' or 'DOS'
REAL      time                !current project time
REAL      x, y                !Location (project coord)
REAL      mean, var
END TYPE InteractiveSrf

END MODULE sensor_fd

```

2.1.27 spcstruct_fd.f90

```

MODULE spcstruct_fd
USE prjstruct_fd      !Basic project structures

!==== terrainHeadT =====
TYPE terrainHeadT
SEQUENCE
INTEGER status
INTEGER max
INTEGER nx
INTEGER ny
REAL    x0
REAL    y0
REAL    dx
REAL    dy

```

```

      REAL      hmin
END TYPE  terrainHeadT

!===== pterrainHeadT =====

TYPE  pterrainHeadT
SEQUENCE
  TYPE ( projectIDT ) project
  TYPE ( terrainHeadT ) terrain
END TYPE  pterrainHeadT

!===== puffHeadT =====

TYPE  puffHeadT
SEQUENCE
  INTEGER maxPuff
  INTEGER maxAux
  INTEGER nPuff
  INTEGER nAux
  REAL    time
END TYPE  puffHeadT

!===== ppuffHeadT =====

TYPE  ppuffHeadT
SEQUENCE
  TYPE ( projectIDT ) project
  TYPE ( puffHeadT ) puff
END TYPE  ppuffHeadT

!===== puffT =====

TYPE  puffT !Basic puff structure - currently identical to puff_str
SEQUENCE
  REAL    xbar,ybar,zbar
  REAL    sxx,sxy,sxz,syy,syz,szz
  REAL    axx,axy,axz,ayy,ayz,azz,det
  REAL    c,cc,xuc,xvc,yvc,yvsc,yvbc,zwc,wc,ccb
  REAL    si,si2,sv
  REAL    sr,cfo
  REAL    zi,zc
  REAL    uo,vo,wo
  INTEGER ityp,inxt,iprv,ipgd,idtl,idtn,iaux
END TYPE  puffT

END MODULE spcstruct_fd

```

2.1.28 structure_fd.f90

```

MODULE structure_fd

USE prjstruct_fd      !Basic project structures
USE msgstruct_fd      !Message/Error structures

```

```

USE timstruct_fd      !Time structures
USE domstruct_fd     !Domain structures
USE mtlstruct_fd     !Material structures
USE inpstruct_fd     !Other input structures
USE metstruct_fd     !Weather structures
USE relstruct_fd     !Release structures
USE environment_fd   !Enviroment structures
USE spcstruct_fd     !Special structures
                      ! - may be eliminated/expanded in the future

!==== limitT =====

TYPE limitT
  SEQUENCE
    INTEGER puffs
    INTEGER surfaceGrid
    INTEGER met1D
  END TYPE limitT

!==== charXXT =====

TYPE char16T
  SEQUENCE
    CHARACTER(16) string
  END TYPE char16T

TYPE char32T
  SEQUENCE
    CHARACTER(32) string
  END TYPE char32T

TYPE char64T
  SEQUENCE
    CHARACTER(64) string
  END TYPE char64T

TYPE char128T
  SEQUENCE
    CHARACTER(128) string
  END TYPE char128T

!==== inputT =====

TYPE inputT
  SEQUENCE
    TYPE ( ctrlT ) ctrl
    TYPE ( temporalT ) time
    TYPE ( flagsT ) flags
    TYPE ( spatialT ) domain
    TYPE ( optionsT ) options
    TYPE ( listHeadT ) mtlHead
  END TYPE inputT

```

```

!==== pinputT =====

TYPE pinputT
SEQUENCE
TYPE ( projectIDT ) project
TYPE ( inputT ) input
END TYPE pinputT

!==== standardInputT =====

TYPE standardInputT
SEQUENCE
TYPE ( temporalT ) time
TYPE ( flagsT ) flags
TYPE ( spatialT ) domain
TYPE ( optionsT ) options
TYPE ( listHeadT ) mtlHead
END TYPE standardInputT

!==== pstandardInputT =====

TYPE pstandardInputT
SEQUENCE
TYPE ( projectIDT ) project
TYPE ( standardInputT ) input
END TYPE pstandardInputT

!==== restartInputT =====

TYPE restartInputT
SEQUENCE
TYPE ( ctrlT ) ctrl
TYPE ( endT ) end
TYPE ( auditT ) audit
TYPE ( spatialT ) domain
TYPE ( optionsT ) options
END TYPE restartInputT

!==== prestartInputT =====

TYPE prestartInputT
SEQUENCE
TYPE ( projectIDT ) project
TYPE ( restartInputT ) input
END TYPE prestartInputT

!==== createNewT =====

TYPE createNewT
SEQUENCE
TYPE ( projectIDT ) project
TYPE ( standardInputT ) input
TYPE ( weatherT ) weather

```

```

      TYPE ( listHeadT      ) scnHead
END TYPE  createNewT

!===== createRstT =====
TYPE  createRstT
SEQUENCE
  TYPE ( projectIDT    ) project
  TYPE ( restartInputT ) input
  TYPE ( weatherT     ) weather
END TYPE  createRstT

!===== projectT =====
TYPE  projectT
SEQUENCE
  TYPE ( projectIDT ) project
  TYPE ( inputT      ) input
  TYPE ( weatherT   ) weather
  TYPE ( listHeadT   ) scnHead
  TYPE ( statusT     ) current
END TYPE  projectT

!===== updateRelT =====
TYPE  updateRelT
SEQUENCE
  INTEGER          mode
  REAL             currentTime
  REAL             nextUpdate
  TYPE ( releaseT  ) release
  TYPE ( environmentT ) environment
END TYPE  updateRelT

!===== computeEffT =====
TYPE  computeEffT
SEQUENCE
  INTEGER      incidentID
  INTEGER      effectID
  INTEGER      request
  INTEGER      addressIn
  INTEGER      addressOut
END TYPE  computeEffT

END MODULE structure_fd

```

2.1.29 terrain_fd.f90

```
MODULE terrain_fd
```

```
!===== HPAC mass-consistency types =====
```

```

INTEGER, PARAMETER :: HTB_SWIFT      = 0
INTEGER, PARAMETER :: HTB_SCIPUFF   = 1
INTEGER, PARAMETER :: HTB_AVAIL_T    = 9
INTEGER, PARAMETER :: HTB_USE_T     = 10
INTEGER, PARAMETER :: HTB_AVAIL_L    = 13
INTEGER, PARAMETER :: HTB_USE_L     = 14
INTEGER, PARAMETER :: HTB_CATEGORY  = 15
INTEGER, PARAMETER :: HT_NONE       = 0
INTEGER, PARAMETER :: HT_SWIFT      = 2**HTB_SWIFT
INTEGER, PARAMETER :: HT_SCIPUFF    = 2**HTB_SCIPUFF
INTEGER, PARAMETER :: HT_AVAIL_T    = 2**HTB_AVAIL_T
INTEGER, PARAMETER :: HT_USE_T     = 2**HTB_USE_T
INTEGER, PARAMETER :: HT_AVAIL_L    = 2**HTB_AVAIL_L
INTEGER, PARAMETER :: HT_USE_L     = 2**HTB_USE_L
INTEGER, PARAMETER :: HT_CATEGORY  = 2**HTB_CATEGORY

```

!==== HPAC met output types =====

```

INTEGER, PARAMETER :: HOB_OUTPUT = 0
INTEGER, PARAMETER :: HOB_OUTMET = 1
INTEGER, PARAMETER :: HOB_ASCII  = 2
INTEGER, PARAMETER :: HOB_2D    = 3
INTEGER, PARAMETER :: HOB_3D    = 4
INTEGER, PARAMETER :: HO_OFF    = 0
INTEGER, PARAMETER :: HO_OUTPUT = 2**HOB_OUTPUT
INTEGER, PARAMETER :: HO_OUTMET = 2**HOB_OUTMET
INTEGER, PARAMETER :: HO_ASCII  = 2**HOB_ASCII
INTEGER, PARAMETER :: HO_2D    = 2**HOB_2D
INTEGER, PARAMETER :: HO_3D    = 2**HOB_3D

```

!==== terrainMCT =====

```
INTEGER, PARAMETER :: HS_MAXZB = 50
```

```

TYPE terrainMCT
  SEQUENCE
    REAL      dtSWIFT
    INTEGER   maxIter(2)
    REAL      eps(2)
    REAL      alpha(2)
    INTEGER   nz
    REAL      z(HS_MAXZB)
END TYPE terrainMCT

```

!==== terrainT

```

TYPE terrainT
  SEQUENCE
    INTEGER      type
    TYPE( terrainMCT ) mc
    CHARACTER(64) name
    CHARACTER(128) path
END TYPE terrainT

```

```
END MODULE terrain_fd
```

2.1.30 time_fd.f90

```
MODULE time_fd
```

```
!===== Parameters ======
```

```
!===== HPAC Time reference types
```

```
INTEGER,PARAMETER :: HT_UTC      = 0
INTEGER,PARAMETER :: HT_LOCAL    = 1
```

```
!===== timeT ======
```

```
TYPE timeT
  SEQUENCE
    INTEGER reference !local
    INTEGER year      !year_start,year_end
    INTEGER month     !month_start,month_end
    INTEGER day       !day_start,day_end
    REAL    hour      !tstart,tend
    REAL    runTime   !tend_hr
  END TYPE timeT
```

```
!===== stepT ======
```

```
TYPE stepT
  SEQUENCE
    REAL    max        !delt
    REAL    output     !dt_save
  END TYPE stepT
```

```
!===== startT ======
```

```
TYPE startT
  SEQUENCE
    REAL          zone
    TYPE ( timeT ) time
  END TYPE startT
```

```
!===== endT ======
```

```
TYPE endT
  SEQUENCE
    TYPE ( timeT ) time
    TYPE ( stepT ) step
  END TYPE endT
```

```
!===== ctrlT ======
```

```
TYPE ctrlT
```

```

SEQUENCE
REAL           runTime
CHARACTER(64)   name
CHARACTER(128)  path
END TYPE ctrlT

END MODULE time_fd

```

2.1.31 timstruct_fd.f90

```

MODULE timstruct_fd
  USE time_fd          !Basic time structures
  USE prjstruct_fd     !Basic project structures

!==== pstartT =====
  TYPE pstartT
    SEQUENCE
      TYPE( projectIDT ) project
      TYPE( startT       ) start
    END TYPE pstartT

!==== pendT =====
  TYPE pendT
    SEQUENCE
      TYPE( projectIDT ) project
      TYPE( endT        ) end
    END TYPE pendT

!==== pctrlT =====
  TYPE pctrlT
    SEQUENCE
      TYPE( projectIDT ) project
      TYPE( ctrlT       ) ctrl
    END TYPE pctrlT

!==== temporalT =====
  TYPE temporalT
    SEQUENCE
      TYPE( startT ) start
      TYPE( endT   ) end
    END TYPE temporalT

!==== ptemporalT =====
  TYPE ptemporalT
    SEQUENCE
      TYPE( projectIDT ) project
      TYPE( temporalT  ) time
    END TYPE ptemporalT

```

```
END MODULE timstruct_fd
```

2.1.32 tooluser_fd.f90

```
MODULE tooluser_fd

USE convert_fd
USE field_fd
USE filemgr_fd
USE plotfx_fd
USE plotlist_fd
USE structure_fd
USE type_fd
USE effects_fd
USE sensor_fd

! ****
!          HPACTool User Definitions
! ****

!==== HPAC Results

INTEGER, PARAMETER :: HPACunknown      = FileMgrUnknown
INTEGER, PARAMETER :: HPACfailure       = FileMgrFailure
INTEGER, PARAMETER :: HPACnull          = FileMgrNull
INTEGER, PARAMETER :: HPACsuccess        = FileMgrSuccess
INTEGER, PARAMETER :: HPACnegative       = FileMgrNegative
INTEGER, PARAMETER :: HPACaffirmative    = FileMgrAffirmative
INTEGER, PARAMETER :: HPACvalid          = FileMgrValid
INTEGER, PARAMETER :: HPACinvalid         = FileMgrInvalid
INTEGER, PARAMETER :: HPACerror          = FileMgrError
INTEGER, PARAMETER :: HPACon             = FileMgrOn
INTEGER, PARAMETER :: HPACoff            = FileMgrOff
INTEGER, PARAMETER :: HPACtrue           = HPACon
INTEGER, PARAMETER :: HPACfalse          = HPACoff

!==== HPAC Errors

INTEGER, PARAMETER :: FORTRAN_IOS_ERROR = 3*(16**6)          !(Z'30000000')
INTEGER, PARAMETER :: CALLBACK_ERROR     = 2*(16**6)+8*(16**5) !(Z'28000000')
INTEGER, PARAMETER :: FILEMGR_ERROR     = 2*(16**6)+4*(16**5) !(Z'24000000')

!==== HPAC Buttons

INTEGER, PARAMETER :: SHOWLEFTBUTTON     = 1
INTEGER, PARAMETER :: SHOWMIDDLEBUTTON   = 2
INTEGER, PARAMETER :: SHOWRIGHTBUTTON    = 4
INTEGER, PARAMETER :: ENABLELEFTBUTTON   = 8
INTEGER, PARAMETER :: ENABLERIGHTBUTTON  = 32
INTEGER, PARAMETER :: ENABLEMIDDLEBUTTON = 16

!==== HPAC CallBack Messages
```

```

INTEGER, PARAMETER :: HM_CHECK      = 0
INTEGER, PARAMETER :: HM_SETWAIT    = 1
INTEGER, PARAMETER :: HM_RELEASEWAIT = 2
INTEGER, PARAMETER :: HM_SETCLOCK    = 3
INTEGER, PARAMETER :: HM_STEPCLOCK   = 4
INTEGER, PARAMETER :: HM_STOPCLOCK   = 5

INTEGER, PARAMETER :: HM_EXITEFF     = 32

!      Parameter = Pointer to MessageT Structure

INTEGER, PARAMETER :: HM_MESSAGE     = 1024
INTEGER, PARAMETER :: HM_PROGRESSMSG = HM_MESSAGE
INTEGER, PARAMETER :: HM_INFO        = HM_MESSAGE + 1
INTEGER, PARAMETER :: HM_ERROR       = HM_MESSAGE + 2
INTEGER, PARAMETER :: HM_REPLY        = HM_MESSAGE + 3
INTEGER, PARAMETER :: HM_STOP         = HM_MESSAGE + 4
INTEGER, PARAMETER :: HM_BTUTONTAG   = HM_MESSAGE + 5

!      Parameter = Integer value

INTEGER, PARAMETER :: HM_INTEGER     = 512
INTEGER, PARAMETER :: HM_START        = HM_INTEGER + 1
INTEGER, PARAMETER :: HM_PROGRESSBAR = HM_INTEGER + 3
INTEGER, PARAMETER :: HM_BUTTONSTATE = HM_INTEGER + 5

!      Parameter = Pointer to Structure

INTEGER, PARAMETER :: HM_STRUCTURE   = 256

!      Parameter = Pointer to ReleaseT Structure

INTEGER, PARAMETER :: HM_RELEASE      = HM_STRUCTURE + 1

!      Parameter = Pointer to UpdateRelT Structure

INTEGER, PARAMETER :: HM_UPDATEREL   = HM_STRUCTURE + 2

!DEC$ IF DEFINED (NWPN)
!      Parameter = Pointer to ComputeEffT Structure

INTEGER, PARAMETER :: HM_COMPUTEEFF  = HM_STRUCTURE + 32

!DEC$ ENDIF
!      Parameter = Pointer to an array

INTEGER, PARAMETER :: HM_ARRAY       = 128

!      Parameter = Pointer to Real Array

INTEGER, PARAMETER :: HM_SYNC        = HM_ARRAY + 1      !t,t+delt

```

```

!      Parameter = Pointer to Integer Array

INTEGER, PARAMETER :: HM_INITEFF = HM_ARRAY + 32      !nIncident
INTEGER, PARAMETER :: HM_HASEFF  = HM_ARRAY + 33      !IncidentID, effectID

!==== HPAC Initialization return values
!      Bits

INTEGER, PARAMETER :: HIB_UTM      = 0
INTEGER, PARAMETER :: HIB_SWIFT    = 1
!DEC$ IF DEFINED (POPLIB)
INTEGER, PARAMETER :: HIB_POP     = 3
!DEC$ ENDIF
!DEC$ IF DEFINED (NWPN)
INTEGER, PARAMETER :: HIB_RIPD    = 4
!DEC$ ENDIF
!DEC$ IF DEFINED (FXPLOT)
INTEGER, PARAMETER :: HIB_FX      = 5
!DEC$ ENDIF

!      Values

INTEGER, PARAMETER :: HI_UTM      = 2**HIB_UTM
INTEGER, PARAMETER :: HI_SWIFT    = 2**HIB_SWIFT
!DEC$ IF DEFINED (POPLIB)
INTEGER, PARAMETER :: HI_POP      = 2**HIB_POP
!DEC$ ENDIF
!DEC$ IF DEFINED (NWPN)
INTEGER, PARAMETER :: HI_RIPD    = 2**HIB_RIPD
!DEC$ ENDIF
!DEC$ IF DEFINED (FXPLOT)
INTEGER, PARAMETER :: HI_FX       = 2**HIB_FX
!DEC$ ENDIF

!==== HPAC Control Identifiers

INTEGER, PARAMETER :: HC_LEFTBUTTON = 201
INTEGER, PARAMETER :: HC_MIDDLEBUTTON = 202
INTEGER, PARAMETER :: HC_RIGHTBUTTON = 203

!==== HPAC input types

INTEGER, PARAMETER :: HPACB_STATUS   = 10
INTEGER, PARAMETER :: HPACB INCIDENT = 9
INTEGER, PARAMETER :: HPACB WEATHER   = 8
INTEGER, PARAMETER :: HPACB RELEASE   = 7
INTEGER, PARAMETER :: HPACB MATERIAL  = 6
INTEGER, PARAMETER :: HPACB OPTIONS   = 5
INTEGER, PARAMETER :: HPACB DOMAIN    = 4
INTEGER, PARAMETER :: HPACB FLAGS     = 3
INTEGER, PARAMETER :: HPACB END       = 2
INTEGER, PARAMETER :: HPACB START     = 1

```

```

INTEGER, PARAMETER :: HPACB_CTRL      = 0
INTEGER, PARAMETER :: HPAC_STATUS     = 2**HPACB_STATUS
INTEGER, PARAMETER :: HPAC INCIDENT   = 2**HPACB INCIDENT
INTEGER, PARAMETER :: HPAC WEATHER    = 2**HPACB WEATHER
INTEGER, PARAMETER :: HPAC RELEASE    = 2**HPACB RELEASE
INTEGER, PARAMETER :: HPAC MATERIAL   = 2**HPACB MATERIAL
INTEGER, PARAMETER :: HPAC OPTIONS    = 2**HPACB OPTIONS
INTEGER, PARAMETER :: HPAC DOMAIN    = 2**HPACB DOMAIN
INTEGER, PARAMETER :: HPAC FLAGS     = 2**HPACB FLAGS
INTEGER, PARAMETER :: HPAC END       = 2**HPACB END
INTEGER, PARAMETER :: HPAC START     = 2**HPACB START
INTEGER, PARAMETER :: HPAC CTRL      = 2**HPACB CTRL
INTEGER, PARAMETER :: HPAC TIME      = HPAC_START + HPAC_END
INTEGER, PARAMETER :: HPAC RUN       = HPAC_CTRL + HPAC_END
INTEGER, PARAMETER :: HPAC INPUT     = HPAC_CTRL + HPAC_TIME + HPAC_FLAGS + &
                                         HPAC_DOMAIN + HPAC_OPTIONS + HPAC_MATERIAL
INTEGER, PARAMETER :: HPAC RESTART   = HPAC_CTRL + HPAC_TIME + HPAC_FLAGS + &
                                         HPAC_DOMAIN + HPAC_OPTIONS + HPAC_MATERIAL
INTEGER, PARAMETER :: HPAC COMPLETE  = HPAC_INPUT + HPAC_RELEASE + HPAC_WEATHER

```

!==== HPAC Input control modes

```

INTEGER, PARAMETER :: HCB_FILE      = 0
INTEGER, PARAMETER :: HCB_SEARCH    = 1
INTEGER, PARAMETER :: HCB_APPEND    = 2
INTEGER, PARAMETER :: HCB_REPLACE   = 3
INTEGER, PARAMETER :: HC_FILE      = 2**HCB_FILE
INTEGER, PARAMETER :: HC_SEARCH    = 2**HCB_SEARCH
INTEGER, PARAMETER :: HC_APPEND    = 2**HCB_APPEND
INTEGER, PARAMETER :: HC_REPLACE   = 2**HCB_REPLACE

```

!==== HPAC conversion modes

```

INTEGER, PARAMETER :: HCB_UTM      = 0
INTEGER, PARAMETER :: HCB_LLA      = 1
INTEGER, PARAMETER :: HC_UTM      = 2**HCB_UTM
INTEGER, PARAMETER :: HC_LLA      = 2**HCB_LLA

```

!==== HPAC update modes

```

INTEGER, PARAMETER :: HUB_TIME     = 10
INTEGER, PARAMETER :: HUB_SPACE    = 11
INTEGER, PARAMETER :: HUB INCIDENT = 12
INTEGER, PARAMETER :: HUB LASTCHANCE = 20
INTEGER, PARAMETER :: HUB ENVIRONMENT = 21
INTEGER, PARAMETER :: HUB MATERIAL = 22
INTEGER, PARAMETER :: HU_TIME     = 2**HUB_TIME
INTEGER, PARAMETER :: HU_SPACE    = 2**HUB_SPACE
INTEGER, PARAMETER :: HU INCIDENT = 2**HUB INCIDENT + HU SPACE + HU TIME
INTEGER, PARAMETER :: HU LASTCHANCE = 2**HUB LASTCHANCE
INTEGER, PARAMETER :: HU ENVIRONMENT = 2**HUB ENVIRONMENT
INTEGER, PARAMETER :: HU MATERIAL = 2**HUB MATERIAL

```

```
!==== HPAC Release status modes
```

```
INTEGER, PARAMETER :: HS_INVALID      = 0
INTEGER, PARAMETER :: HS_VALID       = 1
```

```
!==== HPAC Project status modes
```

```
INTEGER, PARAMETER :: HSB_COPYSCN    = 0
INTEGER, PARAMETER :: HSB_HASTERRAIN = 1
INTEGER, PARAMETER :: HSB_HASDOS     = 2
INTEGER, PARAMETER :: HSB_HASDEP     = 3
INTEGER, PARAMETER :: HSB_HASPUFFS   = 10
INTEGER, PARAMETER :: HS_COPYSCN     = 2**HSB_COPYSCN
INTEGER, PARAMETER :: HS_HASTERRAIN   = 2**HSB_HASTERRAIN
INTEGER, PARAMETER :: HS_HASDOS      = 2**HSB_HASDOS
INTEGER, PARAMETER :: HS_HASDEP      = 2**HSB_HASDEP
INTEGER, PARAMETER :: HS_HASPUFFS    = 2**HSB_HASPUFFS
```

```
!==== HPAC Project file codes
```

```
INTEGER, PARAMETER :: HDB_METFILE    = 0
INTEGER, PARAMETER :: HDB_TERFILE    = 1
INTEGER, PARAMETER :: HDB_RELFILE    = 2
INTEGER, PARAMETER :: HDB_SAMFILE    = 3
INTEGER, PARAMETER :: HDB_INPFILE    = 4
INTEGER, PARAMETER :: HDB_MSCFILE    = 5
INTEGER, PARAMETER :: HDB_SCNFILE    = 6
INTEGER, PARAMETER :: HDB_RADFILE    = 7
INTEGER, PARAMETER :: HDB_PRJFILE    = 8
INTEGER, PARAMETER :: HDB_PUFFILE    = 9
INTEGER, PARAMETER :: HDB_DOSFILE    = 10
INTEGER, PARAMETER :: HDB_DEPFILE    = 11
INTEGER, PARAMETER :: HDB_LOGFILE    = 12
! INTEGER, PARAMETER :: HDB_ATPFILE    = 13
INTEGER, PARAMETER :: HDB_MCWFILE    = 14
INTEGER, PARAMETER :: HDB_PCEFILE    = 15
INTEGER, PARAMETER :: HDB_SMPFILE    = 16
INTEGER, PARAMETER :: HDB_ICDFILE    = 17
INTEGER, PARAMETER :: HD_MAXBITS    = 17
INTEGER, PARAMETER :: HD_METFILE    = 2**HDB_METFILE
INTEGER, PARAMETER :: HD_TERFILE    = 2**HDB_TERFILE
INTEGER, PARAMETER :: HD_RELFILE    = 2**HDB_RELFILE
INTEGER, PARAMETER :: HD_SAMFILE    = 2**HDB_SAMFILE
INTEGER, PARAMETER :: HD_INPFILE    = 2**HDB_INPFILE
INTEGER, PARAMETER :: HD_MSCFILE    = 2**HDB_MSCFILE
INTEGER, PARAMETER :: HD_SCNFILE    = 2**HDB_SCNFILE
INTEGER, PARAMETER :: HD_RADFILE    = 2**HDB_RADFILE
INTEGER, PARAMETER :: HD_PRJFILE    = 2**HDB_PRJFILE
INTEGER, PARAMETER :: HD_PUFFILE    = 2**HDB_PUFFILE
INTEGER, PARAMETER :: HD_DOSFILE    = 2**HDB_DOSFILE
INTEGER, PARAMETER :: HD_DEPFILE    = 2**HDB_DEPFILE
INTEGER, PARAMETER :: HD_LOGFILE    = 2**HDB_LOGFILE
! INTEGER, PARAMETER :: HD_ATPFILE    = 2**HDB_ATPFILE
```

```

INTEGER, PARAMETER :: HD_MCWFILE      = 2**HDB_MCWFILE
INTEGER, PARAMETER :: HD_PCEFILE      = 2**HDB_PCEFILE
INTEGER, PARAMETER :: HD_SMPFILE      = 2**HDB_SMPFILE
INTEGER, PARAMETER :: HD_ICDFILE      = 2**HDB_ICDFILE
INTEGER, PARAMETER :: HD_INPUT        = HD_INPFILE + HD_MSCFILE + HD_SCNFILE
INTEGER, PARAMETER :: HD_OUTPUT        = HD_PRJFILE + HD_PUFFILE + HD_DOSFILE + &
                                         HD_DEPFILE + HD_LOGFILE + &
                                         HD_PCEFILE + HD_SMPFILE ! + HD_ATPPFILE

```

END MODULE tooluser_fd

2.1.33 type_fd.f90

```

MODULE AreaMode_fd

USE poparea_fd

INTEGER, PARAMETER :: HP_BON          = POP_BON
INTEGER, PARAMETER :: HP_BAREA         = POP_BAREA
INTEGER, PARAMETER :: HP_BEXPECT       = POP_BEXPECT
INTEGER, PARAMETER :: HP_OFF           = POP_OFF
INTEGER, PARAMETER :: HP_ON            = 2**HP_BON      ! (1)
INTEGER, PARAMETER :: HP_AREA          = 2**HP_BAREA    ! (2)
INTEGER, PARAMETER :: HP_EXPECT         = 2**HP_BEXPECT ! (4)

END MODULE AreaMode_fd

MODULE type_fd

INTEGER, PARAMETER :: HP_NUMTYP      = 4
INTEGER, PARAMETER :: HP_MEAN         = 1
INTEGER, PARAMETER :: HP_PROB         = 2
INTEGER, PARAMETER :: HP_EXCEED       = 3
INTEGER, PARAMETER :: HP_VARIANCE     = 4

CHARACTER(32), DIMENSION(HP_NUMTYP), PARAMETER :: TYPE_STRING = (/&
    'Mean Value ( M ) ', & !HP_MEAN
    'Probability ( P[v>E] ) ', & !HP_PROB
    'Exceedance ( v[Pc>P] ) ', & !HP_EXCEED
    'Variance ( V ) ', & !HP_VARIANCE
/)

END MODULE type_fd

```

2.1.34 version_fd.f90

```

MODULE version_fd

!==== HPAC Versions

INTEGER, PARAMETER :: HVB_VERSION20 = 18
INTEGER, PARAMETER :: HVB_VERSION30 = 20
INTEGER, PARAMETER :: HV_VERSION20 = 2**HVB_VERSION20

```

```

INTEGER, PARAMETER :: HV_VERSION30 = 2**HVB_VERSION30

END MODULE version_fd

```

2.1.35 weather_fd.f90

```

MODULE weather_fd
  USE version_fd      !Basic version parameters
  USE terrain_fd      !Basic terrain structures

!===== HPAC Weather types =====

  INTEGER, PARAMETER :: HWB_METFIX      = 0
  INTEGER, PARAMETER :: HWB_METSRF      = 1
  INTEGER, PARAMETER :: HWB_METPRF      = 2
  INTEGER, PARAMETER :: HWB_METMRF      = 3
  INTEGER, PARAMETER :: HWB_METMED      = 4
!DEC$ IF DEFINED (CLIMO)
  INTEGER, PARAMETER :: HWB_METSCLI     = 5
  INTEGER, PARAMETER :: HWB_METPCLI     = 6
!DEC$ ENDIF
  INTEGER, PARAMETER :: HWB_METOPER     = 7
!DEC$ IF DEFINED (CLIMO)
  INTEGER, PARAMETER :: HWB METHIST     = 8
!DEC$ ENDIF
  INTEGER, PARAMETER :: HWB_METFCST     = 9
  INTEGER, PARAMETER :: HWB_METCURRE    = 10
  INTEGER, PARAMETER :: HW_METNONE      = 0
  INTEGER, PARAMETER :: HW_METFIX       = 2**HWB_METFIX
  INTEGER, PARAMETER :: HW_METSRF       = 2**HWB_METSRF
  INTEGER, PARAMETER :: HW_METPRF       = 2**HWB_METPRF
  INTEGER, PARAMETER :: HW_METMRF       = 2**HWB_METMRF
  INTEGER, PARAMETER :: HW_METMED       = 2**HWB_METMED
!DEC$ IF DEFINED (CLIMO)
  INTEGER, PARAMETER :: HW_METSCLI      = 2**HWB_METSCLI
  INTEGER, PARAMETER :: HW_METPCLI      = 2**HWB_METPCLI
!DEC$ ENDIF
  INTEGER, PARAMETER :: HW_METOPER      = 2**HWB_METOPER
!DEC$ IF DEFINED (CLIMO)
  INTEGER, PARAMETER :: HW METHIST      = 2**HWB METHIST
!DEC$ ENDIF
  INTEGER, PARAMETER :: HW_METFCST      = 2**HWB_METFCST
  INTEGER, PARAMETER :: HW_METCURRE     = 2**HWB_METCURRE

!===== HPAC Precipitation types =====

  INTEGER, PARAMETER :: HPB_PRCPMET    = 0
  INTEGER, PARAMETER :: HPB_RAIN        = 1
  INTEGER, PARAMETER :: HPB_SNOW        = 2
  INTEGER, PARAMETER :: HPB_LIGHT       = 3
  INTEGER, PARAMETER :: HPB_MODERATE   = 4
  INTEGER, PARAMETER :: HP_NONE        = 0
  INTEGER, PARAMETER :: HP_PRCPINP     = HP_NONE

```

```

INTEGER, PARAMETER :: HP_PRCPMET = 2**HPB_PRCPMET
INTEGER, PARAMETER :: HP_RAIN      = 2**HPB_RAIN
INTEGER, PARAMETER :: HP_SNOW      = 2**HPB_SNOW
INTEGER, PARAMETER :: HP_LIGHT     = 2**HPB_LIGHT
INTEGER, PARAMETER :: HP_MODERATE = 2**HPB_MODERATE
INTEGER, PARAMETER :: HP_HEAVY     = HP_MODERATE + HP_LIGHT

```

!==== HPAC Boundary Layer types =====

```

INTEGER, PARAMETER :: HBB_BLSIMPLE = 0
INTEGER, PARAMETER :: HBB_BL CALC = 1
INTEGER, PARAMETER :: HBB_BLPRF   = 2
INTEGER, PARAMETER :: HBB_BLOBS   = 3
INTEGER, PARAMETER :: HBB_BLMED   = 4
INTEGER, PARAMETER :: HBB_BLOPER  = 7
INTEGER, PARAMETER :: HB_NONE    = 0
INTEGER, PARAMETER :: HB_BLSIMPLE = 2**HBB_BLSIMPLE
INTEGER, PARAMETER :: HB_BL CALC = 2**HBB_BL CALC
INTEGER, PARAMETER :: HB_BLPRF   = 2**HBB_BLPRF
INTEGER, PARAMETER :: HB_BLOBS   = 2**HBB_BLOBS
INTEGER, PARAMETER :: HB_BLMED   = 2**HBB_BLMED
INTEGER, PARAMETER :: HB_BLOPER  = 2**HBB_BLOPER

```

!==== HPAC Surface Moisture types =====

```

INTEGER, PARAMETER :: HM_MSTDRY  = 1
INTEGER, PARAMETER :: HM_MSTNORM = 2
INTEGER, PARAMETER :: HM_MSTWET  = 3

```

!==== HPAC LSV types =====

```

INTEGER, PARAMETER :: HLB_LSVINP = 0
INTEGER, PARAMETER :: HLB_LSVMOD = 1
INTEGER, PARAMETER :: HLB_LSVOBS = 3
INTEGER, PARAMETER :: HLB_LSVOPER = 7
INTEGER, PARAMETER :: HLB_LSVOPER_20 = HLB_LSVOPER + HV B VERSION20
INTEGER, PARAMETER :: HL_NONE    = 0
INTEGER, PARAMETER :: HL_LSVINP = 2**HLB_LSVINP
INTEGER, PARAMETER :: HL_LSVMOD = 2**HLB_LSVMOD
INTEGER, PARAMETER :: HL_LSVOBS = 2**HLB_LSVOBS
INTEGER, PARAMETER :: HL_LSVOPER = 2**HLB_LSVOPER
INTEGER, PARAMETER :: HL_LSVOPER_20 = 2**HLB_LSVOPER_20

```

!==== HPAC Units =====

```

INTEGER, PARAMETER :: HU_DEG    = 1
INTEGER, PARAMETER :: HU_DMS    = 2
INTEGER, PARAMETER :: HU MPS   = 1
INTEGER, PARAMETER :: HU_KTS    = 2
INTEGER, PARAMETER :: HU MPH   = 3
INTEGER, PARAMETER :: HU_KPH    = 4
INTEGER, PARAMETER :: HU FPS   = 5

```

```

!==== metFlagsT =====

TYPE metFlagsT
  SEQUENCE
    INTEGER reference
    INTEGER doMC
    INTEGER doOutput
    INTEGER notUsed
    REAL tOutput
    REAL slHazard
  END TYPE metFlagsT

!==== metMetT =====

INTEGER, PARAMETER :: HS_MAXMETINP = 20

TYPE metMetT
  SEQUENCE
    INTEGER type
    INTEGER nnPrf
    INTEGER nnSfc
    REAL timeBin
    INTEGER nStations
    INTEGER monthDay
    INTEGER unitSpd
    INTEGER unitDir
    REAL speed
    REAL direction
    CHARACTER(128) input(HS_MAXMETINP)
  END TYPE metMetT

!==== metBLT =====

TYPE metBLT
  SEQUENCE
    INTEGER type
    INTEGER wetness
    REAL ziDay
    REAL ziNight
    REAL hfluxDay
    REAL hfluxNight
    REAL canopy
    REAL roughness
    REAL albedo
    REAL bowen
    REAL cloud
    REAL canopyParam
  END TYPE metBLT

!==== metLSVT =====

TYPE metLSVT
  SEQUENCE

```

```

INTEGER type
REAL    uu
REAL    sl
END TYPE metLSVT

!===== metPrecipT =====

TYPE metPrecipT
  SEQUENCE
    INTEGER type
    INTEGER class
  END TYPE metPrecipT

!===== weatherT =====

TYPE weatherT
  SEQUENCE
    TYPE( metFlagsT ) flags
    TYPE( metMetT ) met
    TYPE( metBLT ) bl
    TYPE( metLSVT ) lsv
    TYPE( metPrecipT ) precip
    TYPE( terrainT ) terrain
  END TYPE weatherT

END MODULE weather_fd

```

2.2 C HEADER

A single C language header file containing C structure definitions and function prototypes corresponding to all the Fortran 90 module definition files is listed below. For each Fortran 90 type, a C structure is defined and `typedef'd` as the same name with the first letter capitalized to adhere to typical C language naming conventions. Generally, but not universally, type names end with a capital T.

2.2.1 hpactool.h

```

#ifndef _hpactool_h_
#define _hpactool_h_
*****
*      NAME:          hpactool.h
*      HISTORY:
*          12 Aug 2002      Version 4.0.1, Win32 only
*****
#if defined(__cplusplus) || defined(c_plusplus )
extern "C"
{
#endif

/*****

```

```

*           contour_fd.f90                                     *
***** **** **** **** **** **** **** **** **** **** **** **** **** **** /
#define BLATLON_OUTPUT   12
#define LATLON_OUTPUT    (1 << BLATLON_OUTPUT)
#define BCLOSE_CONTOUR   11
#define CLOSE_CONTOUR    (1 << BCLOSE_CONTOUR)

***** **** **** **** **** **** **** **** **** **** **** **** **** **** **** /
*           convert_fd.f90                                     *
***** **** **** **** **** **** **** **** **** **** **** **** **** **** /
/*
 *  Conversion Factors
 */
#define HCF_HOUR2SEC     3600.0
#define HCF_SEC2HOUR     (1.0 / HCF_HOUR2SEC)

#define HCF_KM2M          1000.0
#define HCF_2KM          (1.0 / HCF_KM2M)

#define HCF_M2MICRON      1.e6
#define HCF_MICRON2M      (1.0 / HCF_M2MICRON)

***** **** **** **** **** **** **** **** **** **** **** **** **** **** **** /
*           domain_fd.f90                                     *
***** **** **** **** **** **** **** **** **** **** **** **** **** **** /
/*
 *  Domain Type Parameters
 */
#define HD_OVCARTESIAN    0
#define HD_LATLON         1
#define HD_OVLATLON       1
#define HD_CARTESIAN      2
#define HD_OVUTM          2
#define HD_UTM            3
#define HD_MAXBITS        17
#define HD_METERS         4

typedef struct referenceT
{
  float x;
  float y;
  float lat;
  float lon;
} ReferenceT;

typedef struct domainT
{
  int map;
  int zoneUtm;
}

```

```

float xMax;
float xMin;
float yMax;
float yMin;
float zMax;
float hRes;
float vRes;
} DomainT;

typedef struct spatialT
{
    DomainT domain;
    ReferenceT reference;
} SpatialT;

/*
typedef struct spatialT
{
    DomainT domain;
    ReferenceT reference;
    int status[HS_MAXDOMAINSTATUS];
} SpatialT;
*/

/********************* field_fd.f90 ****************************/
#define HP_EFFECT      -1
#define HP_EFFECT      -1
#define HP_MET         1
#define HP_DEP          2
#define HP_DOS          3
#define HP_SRFLOS       4
#define HP_TEXPOSE      5

#define HP_PREVIOUS     -999

#define HP_NOTIME        0
#define HP_PUFFTIME      1
#define HP_SRFTIME       2
#define HP_METIME        3
#define HP_RADIME        4

#define HP_NUMCAT        6
#define HP_SURF          1
#define HP_SSLICE        2
#define HP_HSLICE        3
#define HP_VINT          4
#define HP_VSLICE        5
#define HP_TABLE          6
#define HP_CATTYPEn      0

```

```

#define HP_RIGHTHAND      1
#define HP_OPEN            0
#define HP_LEFTHAND        -1

#define HP_SPV             -1.e+36

#define PLOT_ON            1
#define PLOT_OFF           -1
#define PLOT_NULL          0

#define PLOT_LOG           1
#define PLOT_LIN            2
#define PLOT_USER          0

typedef struct HPACPointT
{
    float      x;
    float      y;
} HPACPointT;

typedef struct HPACLineT
{
    int       index;
    int       start;
    int       number;
    int       mode;
} HPACLineT;

typedef struct HPACSliceT
{
    int       resolution;
    HPACPointT   startPt;
    HPACPointT   endPt;
} HPACSliceT;

typedef struct HPACFieldCoordinateT
{
    int       mode;
    int       UTMzone;
    ReferenceT   reference;
    HPACSliceT   vertSlice;
} HPACFieldCoordinateT;

typedef struct HPACPlotFieldT
{
    int       category;
    int       theclass;
}

```

```

int          choice;
int          kind;
int          timeID;
float        userTime;
int          hazard;
int          maxCells;
int          maxLevel;
int          fldLevel;
float        resolution;
int          interpType;
HPACFieldCoordinateT coordinate;
char         units[ 16 ];
char         project[ 128 ];
char         path[ 128 ];
} HPACplotFieldT;

typedef struct HPACplotData
{
    float      xmin;
    float      xmax;
    float      ymin;
    float      ymax;
    float      zmin;
    float      zmax;
    float      zres;
} HPACplotData;

typedef struct HPACContourElementT
{
    float      contour;
    float      value;
    char       label[ 16 ];
    float      area;
    float      population;
} HPACContourElementT;

typedef struct HPACContourHeaderT
{
    int        number;
    float     scale;
    int        labelMode;
    int        drawMode;
    char       unit[ 16 ];
} HPACContourHeaderT;

typedef struct HPACplotTypeT
{
    int        type;
    float     data;
}

```

```

int           areaMode;
} HPACPlotTypeT;

typedef struct HPACPlotFieldNodeT
{
    int           id;
    float         x, y, z;
    float         hx, hy, v;
} HPACPlotFieldNodeT;

typedef struct HPACPlotFieldTriangleT
{
    int           id;
    int           nidA;
    int           nidB;
    int           nidC;
} HPACPlotFieldTriangleT;

/**************************************************************************
*          effects_fd.f90
*****
*/
/*
 * Release Effect (NWPN) Constants
 */
#define EI_MAXEFFECT      4

#define EID_CIRCLE_BLAST      1
#define EID_CIRCLE_THERM      2
#define EID_CIRCLE_PROMPT     3
#define EID_CASUALTY_PROMPT   4

#define EID_INIT              1
#define EID_COMP              2
#define EID_EXIT              3
#define EID_COMP_NOCEP        4

#define IEB_CIRCLE_BLAST      (EID_CIRCLE_BLAST - 1)
#define IEB_CIRCLE_THERM      (EID_CIRCLE_THERM - 1)
#define IEB_CIRCLE_PROMPT     (EID_CIRCLE_PROMPT - 1)
#define IEB_CASUALTY_PROMPT   (EID_CASUALTY_PROMPT - 1)

#define IE_CIRCLE_BLAST       (1 << IEB_CIRCLE_BLAST)
#define IE_CIRCLE_THERM       (1 << IEB_CIRCLE_THERM)
#define IE_CIRCLE_PROMPT      (1 << IEB_CIRCLE_PROMPT)
#define IE_CASUALTY_PROMPT    (1 << IEB_CASUALTY_PROMPT)

/*
 * Effect Variables
 */
#define EVB_GROUP             10

```

```

#define EVB_VAR           11
#define EVB_HAZARD        15

#define EV_TOTAL          0
#define EV_VAPOR          1
#define EV_LIQUID          2
#define EV_LIQUID_DEP      3
#define EV_GROUP          (1 << EVB_GROUP)
#define EV_VARIANCE        (1 << EVB_VAR)
#define EV_HAZARD          (1 << EVB_HAZARD)

typedef struct effectClassT
{
    int         categoryID;
    int         timeID;
    int         linearInterp;
    char        name[ 64 ];
} EffectClassT;

typedef struct effectPlotT
{
    int         classIndx;
    int         kindIndx;
    int         category;
    float       plotTime;
    float       valueNotSet;
} EffectPlotT;

typedef struct effectFieldT
{
    int         typeID;
    int         varID;
    int         classDataIndex;
    float       plotTime;
} EffectFieldT;

/*************************************************************************
*               environment_fd.f90
*************************************************************************/
#define HS_MAXENVIRO      25

typedef struct enviroT
{
    float       z;
    float       pressure;
    float       potentialTemp;
    float       humidity;

```

```

float          windUComp;
float          windVComp;
float          windWComp;
} EnviroT;

typedef struct environmentT
{
float          sfcElevation;
float          sfcPressure;
float          mixingLayerHeight;
EnviroT        samples[ HS_MAXENVIRO ];
int           nSamp;
} EnvironmentT;

typedef struct enviroBLT
{
float          roughness;
float          canopy;
float          alpha;
float          L;
float          surfaceLayer;
float          mixingLayerHeight;
float          wt;
float          uStar;
float          wStar;
float          dTdZ;
} EnviroBLT;

/*************************************************************************
*               filemgr_fd.f90
*   File Manager Results
*/
#define FileMgrUnknown      -2
#define FileMgrFailure       -1
#define FileMgrNull          0
#define FileMgrSuccess        1
#define FileMgrNegative      FileMgrFailure
#define FileMgrAffirmative   FileMgrSuccess
#define FileMgrValid          FileMgrSuccess
#define FileMgrInvalid        FileMgrFailure
#define FileMgrError          FileMgrFailure
#define FileMgrOn             FileMgrSuccess
#define FileMgrOff            FileMgrNull

/* LastError Results */
#define FME_EOF              -1

```

```

#define FME_LISTSIZE      1
#define FME_NOTAVAIL     2
#define FME_NOTFOUND      3
#define FME_FAILURE       4

/**************************************************************************
*           flags_fd.f90
*************************************************************************/
/*
 * Input Flags Types
 */
#define HF_OFF            0

#define HFB_DYNAMIC        0
#define HFB_DENSE          1
#define HFB_STATIC          2
#define HFB_MULTICOMP       3

#define HF_DYNAMIC         (1 << HFB_DYNAMIC)
#define HF_DENSE            (1 << HFB_DENSE)
#define HF_STATIC           (1 << HFB_STATIC)
#define HF_MULTICOMP        (1 << HFB_MULTICOMP)

#define HFB_FAST            0
#define HF_FAST             (1 << HFB_FAST)

#define HFB_HAZARD          1
#define HFB_DUAL            2
#define HF_HAZARD           (1 << HFB_HAZARD)
#define HF_DUAL              (1 << HFB_DUAL)

#define HFB_RESTART          0
#define HF_RESTART           (1 << HFB_RESTART)

typedef struct auditT
{
    char          title[80];
    char          analyst[32];
    char          className[32];
    char          version[32];
    char          date[32];
} AuditT;

typedef struct flagsT
{
    int           start;
    int           method;
    int           mode;
    int           prjEffects;
    AuditT        audit;
}

```

```

} Flagst;

/*****************
*      hpactool_mod.f90
*****************/
/* ***** */
*      options_fd.f90
***** */

typedef struct OptionsT
{
    int          nzBL;
    int          mGrd;
    int          substrate;
    float        timeAvg;
    float        massMin;
    float        delMin;
    float        wwTrop;
    float        epsTrop;
    float        slTrop;
    float        uuCalm;
    float        slCalm;
    float        zDosage;
    float        dtSampler;
    char         samplerFile[ 64 ];
    char         samplerPath[ 128 ];
} OptionsT;

/*****************
*      list_fd.f90
*****************/
/* ***** */
*      prjstruct_fd.f90
***** */

typedef struct listHeadT
{
    int          max;
    int          number;
} ListHeadT;

/*****************
*      prjstruct_fd.f90
***** */

typedef struct projectIDT
{
    int          id;
    int          version;
    char         name[ 64 ];
    char         path[ 128 ];
}

```

```

} ProjectIDT;

/********************* domstruct_fd.f90 ********************/
typedef struct pspatialT
{
    ProjectIDT    project;
    SpatialT      spatial;
} PSpatialT;

/********************* time_fd.f90 ********************/
#define HT_LOCAL          1
#define HT_UTC            0

typedef struct timeT
{
    int           reference;
    int           year;
    int           month;
    int           day;
    float         hour;
    float         runTime;
} TimeT;

typedef struct stepT
{
    float         max;
    float         output;
} StepT;

typedef struct startT
{
    float         zone;
    TimeT         time;
} StartT;

typedef struct endT
{
    TimeT         time;
    StepT         step;
} EndT;

typedef struct ctrlT

```

```

{
float          runTime;
char           name[ 64 ];
char           path[ 128 ];
} CtrlT;

*****
*      inpstruct_fd.f90
*****
typedef struct pauditT
{
ProjectIDT    project;
AuditT        audit;
} PAuditT;

typedef struct pflagsT
{
ProjectIDT    project;
FlagsT         flags;
} PFlagsT;

typedef struct poptionsT
{
ProjectIDT    project;
OptionsT       options;
} POptionsT;

typedef struct statusT
{
int status;
int nPuffType;
TimeT time;
} StatusT;

*****
*      material_fd.f90
*****
/*
*  HPAC Material Types
*/
#define HMB_GAS          0
#define HMB_PARTICLE      1
#define HMB_LIQUID        2
#define HMB_AEROSOL       3
#define HMB_WETPARTICLE   4
#define HMB_2NDEVAP       20
#define HMB_NWPN          21
#define HMB_NFAC          22

```

```

#define HMB_HAZARD      23
#define HMB_MULTICOMP   24
#define HMB_BASIC       10

#define HM_GAS          (1 << HMB_GAS)
#define HM_PARTICLE     (1 << HMB_PARTICLE)
#define HM_LIQUID        (1 << HMB_LIQUID)
#define HM_AEROSOL      (1 << HMB_AEROSOL)
#define HM_WETPARTICLE   (1 << HMB_WETPARTICLE)
#define HM_2NDEVAP       (1 << HMB_2NDEVAP)
#define HM_NWPN          (1 << HMB_NWPN)
#define HM_NFAC          (1 << HMB_NFAC)
#define HM_HAZARD        (1 << HMB_HAZARD)
#define HM_MULTICOMP     (1 << HMB_MULTICOMP)

/*
 * Surface File Modes
 */
#define HSB_GROUPDEP    0
#define HSB_GROUPDOS    1
#define HSB_TOTALDEP    2
#define HSB_TOTALDOS    3

#define HS_NONE          0
#define HS_GROUPDEP     (1 << HSB_GROUPDEP)
#define HS_GROUPDOS     (1 << HSB_GROUPDOS)
#define HS_TOTALDEP     (1 << HSB_TOTALDEP)
#define HS_TOTALDOS     (1 << HSB_TOTALDOS)

/*
 * Parameters
 */
#define HS_MAXMTLRESERVED 2
#define HS_MAXMTLBINSIZE 50
#define HS_MAXMTLBASIC 17
#define HS_MAXMTLGAS 7
#define HS_MAXMTLAEROSOL 13
#define HS_MAXMTLLIQUID 17
#define HS_MAXMTLPARTICLE 7

#define HS_PADMTLGEN    (HS_MAXMTLBASIC + \
                      HS_MAXMTLRESERVED + (2*HS_MAXMTLBINSIZE) + 1)
#define HS_PADMTLGAS    (HS_MAXMTLBASIC - HS_MAXMTLGAS + \
                      HS_MAXMTLRESERVED + (2*HS_MAXMTLBINSIZE) + 1)

#define HS_PADMTLAEROSOL (HS_MAXMTLBASIC - HS_MAXMTLAEROSOL + \
                        HS_MAXMTLRESERVED + (2*HS_MAXMTLBINSIZE) + 1)
#define HS_PADMTLLIQUID (HS_MAXMTLBASIC - HS_MAXMTLLIQUID + HS_MAXMTLRESERVED)
#define HS_PADMTLPARTICLE (HS_MAXMTLBASIC - HS_MAXMTLPARTICLE + \
                        HS_MAXMTLRESERVED)

```

typedef struct matGenT

```

{
int           padding[ HS_PADMTLGEN ];
} MatGenT;

typedef struct matAerosolT
{
float         minConcentration;
float         decayAmp;
float         decayMin;
float         NWPNDelay;
int          rNotUsed;
float         gasDeposition;
float         liquidDensity;
float         antoine[ 3 ];
float         molWeight;
float         liqSpecificHeat;
float         gasSpecificHeat;
int          padding[ HS_PADMTLAEROSOL ];
} MatAerosolT;

typedef struct matGasT
{
float         minConcentration;
float         decayAmp;
float         decayMin;
float         NWPNDelay;
int          rNotUsed;
float         gasDensity;
float         gasDeposition;
int          padding[ HS_PADMTLGAS ];
} MatGasT;

typedef struct matLiquidT
{
float         minConcentration;
float         decayAmp;
float         decayMin;
float         NWPNDelay;
int          rNotUsed;
float         gasDensity;
float         gasDeposition;
float         liquidDensity[ 2 ];
float         antoine[ 3 ];
float         molWeight;
float         srfTension;
float         spreadFactor;
float         viscosity;
int          nSizeBins;
float         binSize[ HS_MAXMTLBINSIZE ];
float         binBounds[ HS_MAXMTLBINSIZE+1 ];
}

```

```

int           padding[ HS_PADMTLLIQUID ];
} MatLiquidT;

typedef struct materialT
{
    int          type;
    int          puffIndex;
    int          effectClass;
    int          effectAvail;
    MatGenT     matData;
    char         name[ 16 ];
    char         units[ 16 ];
    char         file[ 64 ];
    char         path[ 128 ];
} MaterialT;

typedef struct matParticleT
{
    float        minConcentration;
    float        decayAmp;
    float        decayMin;
    float        NWPNDelay;
    int          save;
    float        density;
    int          nSizeBins;
    float        binSize[ HS_MAXMTLBINSIZE ];
    float        binBounds[ HS_MAXMTLBINSIZE+1 ];
    int          padding[ HS_PADMTLPARTICLE ];
} MatParticleT;

/*************************************************************************
*               message_fd.f90                                         *
*************************************************************************/
typedef struct messageT
{
    int          iParm;
    int          jParm;
    char         aString[ 128 ];
    char         bString[ 128 ];
    char         cString[ 128 ];
    char         routine[ 80 ];
} MessageT;

/*************************************************************************
*               version_fd.f90                                         *
*************************************************************************/
#define HVB_VERSION20      18
#define HVB_VERSION30      20

```

```

#define HV_VERSION20      (1 << HVB_VERSION20)
#define HV_VERSION30      (1 << HVB_VERSION30)

/*
 *      terrain_fd.f90
 */
/*
 * Mass Consistency Types
 */
#define HTB_SWIFT          0
#define HTB_SCIPUFF         1
#define HTB_AVAIL_T         9
#define HTB_USE_T           10
#define HTB_AVAIL_L         13
#define HTB_USE_L           14
#define HTB_CATEGORY        15

#define HT_NONE             0
#define HT_SWIFT            (1 << HTB_SWIFT)
#define HT_SCIPUFF          (1 << HTB_SCIPUFF)
#define HT_AVAIL_T          (1 << HTB_AVAIL_T)
#define HT_USE_T             (1 << HTB_USE_T)
#define HT_AVAIL_L          (1 << HTB_AVAIL_L)
#define HT_USE_L             (1 << HTB_USE_L)
#define HT_CATEGORY          (1 << HTB_CATEGORY)

/*
 * Met Output Types
 */
#define HOB_OUTPUT          0
#define HOB_OUTMET          1
#define HOB_ASCII            2
#define HOB_2D               3
#define HOB_3D               4

#define HO_OUTPUT            (1 << HOB_OUTPUT)
#define HO_OUTMET            (1 << HOB_OUTMET)
#define HO_ASCII              (1 << HOB_ASCII)
#define HO_2D                 (1 << HOB_2D)
#define HO_3D                 (1 << HOB_3D)

#define HS_MAXZB            50

typedef struct terrainMCT
{
    float      dtSWIFT;
    int       maxIter[ 2 ];
    float      eps[ 2 ];
    float      alpha[ 2 ];
    int       nz;
}

```

```

float          z[ HS_MAXZB ];
} TerrainMCT;

typedef struct terrainT
{
    int           type;
    TerrainMCT   mc;
    char          name[ 64 ];
    char          path[ 128 ];
} TerrainT;

/**************************************************************************
*      weather_fd.f90
*****
*/
/*
* Weather Types
*/
#define HWB_METFIX      0
#define HWB_METSRF      1
#define HWB_METPRF      2
#define HWB_METMRF      3
#define HWB_METMED      4
#define HWB_METSCLI     5
#define HWB_METPCLI     6
#define HWB_METOPER      7
#define HWB METHIST      8
#define HWB METFCST      9
#define HWB METCURR     10

#define HW_METNONE      0
#define HW_METFIX       (1 << HWB_METFIX)
#define HW_METSRF       (1 << HWB_METSRF)
#define HW_METPRF       (1 << HWB_METPRF)
#define HW_METMRF       (1 << HWB_METMRF)
#define HW_METMED       (1 << HWB_METMED)
#define HW_METSCLI      (1 << HWB_METSCLI)
#define HW_METPCLI      (1 << HWB_METPCLI)
#define HW_METOPER       (1 << HWB_METOPER)
#define HW METHIST       (1 << HWB METHIST)
#define HW METFCST       (1 << HWB METFCST)
#define HW METCURR       (1 << HWB METCURR)

/*
* Precipitation Types
*/
#define HPB_PRCPMET    0
#define HPB_RAIN        1
#define HPB_SNOW        2
#define HPB_LIGHT       3
#define HPB_MODERATE    4

```

```

#define HP_NONE          0
#define HP_PRCPINP      HP_NONE
#define HP_PRCPMET      (1 << HPB_PRCPMET)
#define HP_RAIN          (1 << HPB_RAIN)
#define HP_SNOW          (1 << HPB_SNOW)
#define HP_LIGHT         (1 << HPB_LIGHT)
#define HP_MODERATE     (1 << HPB_MODERATE)
#define HP_HEAVY         (HP_MODERATE + HP_LIGHT)

/*
 * Boundary Layer Types
 */
#define HBB_BLSIMPLE    0
#define HBB_BLCALC       1
#define HBB_BLPRF        2
#define HBB_BLOBS        3
#define HBB_BLMED        4
#define HBB_BLOPER       7

#define HB_NONE          0
#define HB_BLSIMPLE     (1 << HBB_BLSIMPLE)
#define HB_BLCALC        (1 << HBB_BLCALC)
#define HB_BLPRF         (1 << HBB_BLPRF)
#define HB_BLOBS         (1 << HBB_BLOBS)
#define HB_BLMED         (1 << HBB_BLMED)
#define HB_BLOPER        (1 << HBB_BLOPER)

/*
 * Surface Moisture Types
 */
#define HM_MSTDRY        1
#define HM_MSTNORM       2
#define HM_MSTWET        3

/*
 * LSV Types
 */
#define HLB_LSVINP       0
#define HLB_LSVMOD       1
#define HLB_LSVOBS       3
#define HLB_LSVOPER      7
#define HLB_LSVOPER_20   (HLB_LSVOPER + HVB_VERSION20)

#define HL_NONE          0
#define HL_LSVINP        (1 << HLB_LSVINP)
#define HL_LSVMOD        (1 << HLB_LSVMOD)
#define HL_LSVOBS        (1 << HLB_LSVOBS)
#define HL_LSVOPER       (1 << HLB_LSVOPER)
#define HL_LSVOPER_20   (1 << HLB_LSVOPER_20)

/*
 * HPAC Units
*/

```

```

#define HU_DEG          1
#define HU_DMS          2
#define HU_MPS          1
#define HU_KTS          2
#define HU MPH          3
#define HU_KPH          4
#define HU FPS          5

typedef struct MetFlagsT
{
    int      reference;
    int      doMC;
    int      doOutput;
    int      notUsed;
    float   tOutput;
    float   slHazard;
} MetFlagsT;

#define HS_MAXMETINP    20

typedef struct metMetT
{
    int      type;
    int      nnPrf;
    int      nnSfc;
    float   timeBin;
    int      nStations;
    int      monthDay;
    int      unitSpd;
    int      unitDir;
    float   speed;
    float   direction;
    char    input[ HS_MAXMETINP][ 128 ];
} MetMetT;

typedef struct metBLT
{
    int      type;
    int      wetness;
    float   ziDay;
    float   ziNight;
    float   hfluxDay;
    float   hfluxNight;
    float   canopy;
    float   roughness;
    float   albedo;
    float   bowen;
    float   cloud;
    float   canopyParam;
} MetBLT;

```

```

typedef struct metLSVT
{
    int          type;
    float        uu;
    float        sl;
} MetLSVT;

typedef struct metPrecipT
{
    int          type;
    int          classType;
} MetPrecipT;

typedef struct weatherT
{
    MetFlagsT    flags;
    MetMetT     met;
    MetBLT      bl;
    MetLSVT     lsv;
    MetPrecipT  precip;
    TerrainT   terrain;
} WeatherT;

/*************************************************************************
*           metstruct_fd.f90
*****
*/

typedef struct pweatherT
{
    ProjectIDT  project;
    WeatherT    weather;
} PWeatherT;

/*************************************************************************
*           mtlstruct_fd.f90
*****
*/

typedef struct mtlControlT
{
    int          mode;
    char         fileExtension[ 8 ];
    char         searchName[ 16 ];
} MtlControlT;

typedef struct pmaterialT
{

```

```

ProjectIDT    project;
ListHeadT     mtlHead;
MtlControlT   control;
} PMaterialT;

/*************************************************************************
*          plotfx_fd.f90
*************************************************************************/
#define FX_NFAC           1
#define FX_NWPN            2
#define FX_NWPNCAS         3
#define FX_NWPNFAT         4
#define FX_NWPNTAB         5
#define FX_TOXL            6
#define FX_FXTOL            7
#define FX_SRFPRBMAX       8
#define FX_SRFCPRBMX       9

#define MAX_PROTECTION_TYPES 6
#define NORNL      (MAX_PROTECTION_TYPES + 1)

/*************************************************************************
*          plotlist_fd.f90
*************************************************************************/

typedef struct HPACClassChoiceT
{
    int           available;
    int           kind;
    int           ikind;
    int           nkind;
    int           itime;
    int           usertime;
    char          units[ 16 ];
} HPACClassChoiceT;

typedef struct HPACCATEGORYCLASST
{
    int           available;
    int           type;
} HPACCATEGORYCLASST;

typedef struct HPACTIMEt
{
    TimeT         time;
    int           nItems;
    char          string[ 24 ];
} HPACTIMEt;

```

```

/*
 *          poparea_fd.f90
 */
#define POP_BON      0
#define POP_BAREA    1
#define POP_BEXPECT  2

#define POP_OFF      0
#define POP_ON       (1 << POP_BON)
#define POP_AREA     (1 << POP_BAREA)
#define POP_EXPECT   (1 << POP_BEXPECT)

/*
 *          release_fd.f90
 */
/*
 * Release Types
 */
#define HRB_INST      0
#define HRB_CONT      1
#define HRB_INTER     2
#define HRB_FILE      20
#define HRB_MOVE      21
#define HRB_POOL      22
#define HRB_STACK     23
#define HRB_PUFF      24

#define HR_INST      (1 << HRB_INST)
#define HR_CONT      (1 << HRB_CONT)
#define HR_FILE      ((1 << HRB_FILE) + HR_INST)
#define HR_MOVE      ((1 << HRB_MOVE) + HR_CONT)
#define HR_POOL      ((1 << HRB_POOL) + HR_CONT)
#define HR_PUFF      ((1 << HRB_PUFF) + HR_INST)
#define HR_STACK     ((1 << HRB_STACK) + HR_CONT)
#define HR_INTER     (1 << HRB_INTER)

/*
 * Release Distribution Types
 */
#define HDB_LOGNORM   0
#define HD_LOGNORM    ((-1) * (1 << HDB_LOGNORM))

/*
 * Parameters
 */
#define HS_MAXRELRESERVED 1
#define HS_MAXRELMAXSIZE  51

#define HS_MAXRELCONTSIZE 10
#define HS_MAXRELFILSIZE  51
#define HS_MAXRELINSTSIZE 13

```

```

#define HS_MAXRELCMOVSIZE      13
#define HS_MAXRELPOOLSIZE      3
#define HS_MAXRELIPUFSIZE      19
#define HS_MAXRELCSTKSIZE       8

#define HS_PADRELGEN      (HS_MAXRELMAXSIZE + HS_MAXRELRESERVED)
#define HS_PADRELCONT     (HS_PADRELGEN - HS_MAXRELCONTSIZE)
#define HS_PADRELFILE      (HS_PADRELGEN - HS_MAXRELFILESIZE)
#define HS_PADRELINST      (HS_PADRELGEN - HS_MAXRELINSTSIZ)
#define HS_PADRELCMOV      (HS_PADRELGEN - HS_MAXRELCMOVSIZE)
#define HS_PADRELPOOL      (HS_PADRELGEN - HS_MAXRELPOOLSIZ)
#define HS_PADRELIPUF      (HS_PADRELGEN - HS_MAXRELIPUFSIZE)
#define HS_PADRELCSTK      (HS_PADRELGEN - HS_MAXRELCSTKSIZE)

typedef struct relGenT
{
    int           padding[ HS_PADRELGEN ];
} RelGenT;

typedef struct relContT
{
    int           distribution;
    float         rate;
    float         duration;
    float         sigY;
    float         sigZ;
    float         MMD;
    float         sigma;
    float         momentum;
    float         buoyancy;
    float         dryFrac;
    int           padding[ HS_PADRELCONT ];
} RelContT;

typedef struct relFileT
{
    int           nRandom;
    int           ranSeed;
    float         ranSpread;
    char          relFile[ 64 ];
    char          relPath[ 128 ];
    int           padding[ HS_PADRELFILE ];
} RelFileT;

typedef struct relInstT
{
    int           distribution;
    float         mass;
    float         sigX;

```

```

float      sigY;
float      sigZ;
float      MMD;
float      sigma;
float      momentum;
float      buoyancy;
float      dryFrac;
int       nRandom;
int       ranSeed;
float      ranSpread;
int       padding[ HS_PADRELINST ];
} RelInstT;

typedef struct relMoveT
{
    int       distribution;
    float     rate;
    float     duration;
    float     sigY;
    float     sigZ;
    float     MMD;
    float     sigma;
    float     momentum;
    float     buoyancy;
    float     dryFrac;
    float     velX;
    float     velY;
    float     velZ;
    int       padding[ HS_PADRELCMOV ];
} RelMoveT;

typedef struct relPoolT
{
    float     mass;
    float     sizeX;
    float     sizeY;
    int       padding[ HS_PADRELPOOL ];
} RelPoolT;

typedef struct relPuffT
{
    int       subgroup;
    float     mass;
    float     sxx;
    float     sxy;
    float     sxz;
    float     syy;
    float     syz;
    float     szz;
    float     difhShear;
}

```

```

float      difhBuoy;
float      difhLSVxx;
float      difhLSVxy;
float      difhLSVyy;
float      difVert;
float      activeFrac;
float      sigRatio;
float      scaleLateral;
float      scaleStream;
float      scaleVert;
int       padding[ HS_PADRELIPUF ];
} RelPuffT;

```

```

typedef struct relStackT
{
    int      distribution;
    float    rate;
    float    duration;
    float    diameter;
    float    MMD;
    float    sigma;
    float    exitVel;
    float    exitTemp;
    int       padding[ HS_PADRELCSTK ];
} RelStackT;

```

```

typedef struct releaseT
{
    int      type;
    int      status;
    float    tRel;
    float    xRel;
    float    yRel;
    float    zRel;
    float    horzUnc;
    float    vertUnc;
#ifndef HPAC_4_0_2
    float    pa;
#endif
    RelGenT  relData;
    char     material[ 16 ];
    char     relName[ 32 ];
    char     relDisplay[ 48 ];
} ReleaseT;

```

```

*****
*          relstruct_fd.f90
*****
****
```

```
typedef struct relControlT
```

```

{
int mode;
char searchID[ 32 ];
char fileExtension[ 8 ];
} RelControlT;

typedef struct preleaseT
{
ProjectIDT project;
ListHeadT scnHead;
RelControlT control;
} PReleaseT;

/**************************************************************************
*          spcstruct_fd.f90
*****
****/



typedef struct terrainHeadT
{
int status;
int max;
int nx;
int ny;
float x0;
float y0;
float dx;
float dy;
float hmin;
} TerrainHeadT;

typedef struct pterrainHeadT
{
ProjectIDT project;
TerrainHeadT terrain;
} PTerrainHeadT;

typedef struct puffHeadT
{
int maxPuff;
int maxAux;
int nPuff;
int nAux;
float time;
} PuffHeadT;

typedef struct ppuffHeadT
{

```

```

ProjectIDT    project;
PuffHeadT     puff;
} PPuffHeadT;

typedef struct puffT
{
    float      xbar, ybar, zbar;
    float      sxx, sxy, sxz, syy, syz, szz;
    float      axx, axy, axz, ayy, ayz, azz, det;
    float      c, cc, xuc, xvc, yvc, yvsc, yvbc, zwc, wc, ccb;
    float      si, si2, sv;
    float      sr, cfo;
    float      zi, zc;
    float      uo, vo, wo;
    int       ityp, inxt, iprv, ipgd, idtl, idtn, iaux;
} PuffT;

/********************* timstruct_fd.f90 ********************/
*****
```

```

typedef struct pstartT
{
    ProjectIDT    project;
    StartT        start;
} PStartT;

typedef struct pendT
{
    ProjectIDT    project;
    EndT          end;
} PEndT;

typedef struct pctrlT
{
    ProjectIDT    project;
    CtrlT         ctrl;
} PCtrlT;

typedef struct temporalT
{
    StartT        start;
    EndT          end;
} TemporalT;

typedef struct ptemporalT
{
```

```

ProjectIDT    project;
TemporalT     time;
} PTemporalT;

/********************* structure_fd.f90 ********************/
*  

typedef struct limitT
{
  int          puffs;
  int          surfaceGrid;
  int          met1D;
} LimitT;

typedef struct char16T
{
  char         string[ 16 ];
} Char16T;

typedef struct char32T
{
  char         string[ 32 ];
} Char32T;

typedef struct char64T
{
  char         string[ 64 ];
} Char64T;

typedef struct char128T
{
  char         string[ 128 ];
} Char128T;

typedef struct inputT
{
  CtrlT        ctrl;
  TemporalT   time;
  FlagsT       flags;
  SpatialT    domain;
  OptionsT    options;
  ListHeadT   mtlHead;
} InputT;

typedef struct pinputT

```

```

{
ProjectIDT    project;
InputT        input;
} PInputT;

typedef struct standardInputT
{
TemporalT    time;
FlagsT        flags;
SpatialT      domain;
OptionsT      options;
ListHeadT     mtlHead;
} StandardInputT;

typedef struct pstandardInputT
{
ProjectIDT    project;
StandardInputT input;
} PStandardInputT;

typedef struct restartInputT
{
CtrlT         ctrl;
EndT          end;
AuditT        audit;
SpatialT      domain;
OptionsT      options;
} RestartInputT;

typedef struct prestartInputT
{
ProjectIDT    project;
RestartInputT input;
} PRestartInputT;

typedef struct createNewT
{
ProjectIDT      project;
StandardInputT  input;
WeatherT        weather;
ListHeadT       scnHead;
} CreateNewT;

typedef struct createRstT
{
ProjectIDT    project;
RestartInputT input;
}

```

```

WeatherT      weather;
} CreateRstT;

typedef struct projectT
{
ProjectIDT    project;
InputT        input;
WeatherT      weather;
ListHeadT     scnHead;
StatusT       current;
} ProjectT;

typedef struct updateRelT
{
int           mode;
float         currentTime;
float         nextUpdate;
ReleaseT      release;
EnvironmentT  environment;
} UpdateRelT;

typedef struct computeEffT
{
int           incidentID;
int           effectID;
int           request;
int           addressIn;
int           addressOut;
} ComputeEffT;

/*************************************************************************
*          nwpneffect_fd.f90
*************************************************************************/
#define CP_MAX_RADII          23
#define CP_MAX_PROTECT         10
#define CP_MAX_SETTINGS         2

typedef struct casualtyPromptInitInT
{
int           nProtect;
Char32T       ProtTypes[ CP_MAX_PROTECT + 1 ];
} CasualtyPromptInitInT;

typedef struct casualtyPromptInitOutT
{
int           nWpn;
float         probFac[ CP_MAX_RADII ];
}

```

```

} CasualtyPromptInitOutT;

typedef struct casualtyPromptCompInT
{
    int          nWpn;
} CasualtyPromptCompInT;

typedef struct casualtyPromptCompOutT
{
    float        loc[ 3 ];
    float        time;
    float        pa;
    float        cep;
    float        rCasualty[ CP_MAX_RADII ][ CP_MAX_PROTECT + 1 ];
    float        rFatality[ CP_MAX_RADII ][ CP_MAX_PROTECT + 1 ];
    float        wrCasualty[ CP_MAX_PROTECT ];
    float        wrFatality[ CP_MAX_PROTECT ];
} CasualtyPromptCompOutT;

/**************************************************************************
*           sensor_fd.f90
*****
/
```

```

typedef struct sensorT
{
    int          id;
    int          init;
    float        time;
    float        x, y, z;
    float        az, el, dist;
    float        h;
    float        mean, var;
} SensorT;

typedef struct InteractiveSensor
{
    char         matname[ 64 ];
    int          isg;
    float        time;
    float        x, y, z;
    float        az, el, dist;
    float        mean, var;
} InteractiveSensor;

typedef struct InteractiveSrf
{
    char         matname[ 64 ];
    int          isg;

```

```

char          stype[ 64 ];
float         time;
float         x, y;
float         mean, var;
} InteractiveSrf;

/*************************************************************************
*          tooluser_fd.f90
*************************************************************************/
#define HPACtrue      1
#define HPACfalse     0
#define HPACCon       1
#define HPACOff       0

#define HPACunknown   -2
#define HPACfailure   -1
#define HPACnull      0
#define HPACsuccess   1
#define HPACnegative  -1
#define HPACaffirmative 1
#define HPACvalid     1
#define HPACinvalid   -1
#define HPACerror     -1

#define FORTRAN_IOS_ERROR      0x30000000
#define CALLBACK_ERROR          0x28000000
#define FILEMGR_ERROR           0x24000000

#define SHOWLEFTBUTTON          1
#define SHOWMIDDLEBUTTON        2
#define SHOWRIGHTBUTTON         4
#define ENABLELEFTBUTTON        8
#define ENABLERIDDLEBUTTON     16
#define ENABLERIGHTBUTTON      32

#define HM_CHECK               0
#define HM_SETWAIT              1
#define HM_RELEASEWAIT          2
#define HM_SETCLOCK              3
#define HM_STEPCLOCK             4
#define HM_STOPCLOCK              5

#define HM_EXITEFF              32

#define HM_MESSAGE              1024
#define HM_PROGRESSMSG          (HM_MESSAGE)
#define HM_INFO                  (HM_MESSAGE+1)
#define HM_ERROR                 (HM_MESSAGE+2)
#define HM_REPLY                 (HM_MESSAGE+3)
#define HM_STOP                  (HM_MESSAGE+4)
#define HM_BUTTONONTAG           (HM_MESSAGE+5)

```

```

#define HM_INTEGER      512
#define HM_START        (HM_INTEGER+1)
#define HM_PROGRESSBAR  (HM_INTEGER+3)
#define HM_BUTTONSTATE  (HM_INTEGER+5)

#define HM_STRUCTURE     256
#define HM_RELEASE       (HM_STRUCTURE + 1)
#define HM_UPDATEREL    (HM_STRUCTURE + 2)
#define HM_COMPUTEEFF   (HM_STRUCTURE + 32)

#define HM_ARRAY         128
#define HM_SYNC          (HM_ARRAY + 1)
#define HM_INITEFF       (HM_ARRAY + 32)
#define HM_HASEFF        (HM_ARRAY + 33)

#define HIB_UTM          0
#define HIB_SWIFT         1
#define HIB_POP           3
#define HIB_RIPD          4
#define HIB_FX            5

#define HI_UTM           (1 << HIB_UTM)
#define HI_SWIFT          (1 << HIB_SWIFT)
#define HI_POP            (1 << HIB_POP)
#define HI_RIPD           (1 << HIB_RIPD)
#define HI_FX             (1 << HIB_FX)

#define HC_LEFTBUTTON    201
#define HC_MIDDLEBUTTON  202
#define HC_RIGHTBUTTON   203

#define HPACB_STATUS     10
#define HPACB INCIDENT   9
#define HPACB WEATHER    8
#define HPACB RELEASE    7
#define HPACB MATERIAL   6
#define HPACB OPTIONS    5
#define HPACB DOMAIN     4
#define HPACB FLAGS      3
#define HPACB END         2
#define HPACB START       1
#define HPACB CTRL        0

#define HPAC STATUS       (1 << HPACB_STATUS)
#define HPAC INCIDENT     (1 << HPACB INCIDENT)
#define HPAC WEATHER      (1 << HPACB WEATHER)
#define HPAC RELEASE      (1 << HPACB RELEASE)
#define HPAC MATERIAL     (1 << HPACB_MATERIAL)
#define HPAC OPTIONS      (1 << HPACB_OPTIONS)
#define HPAC DOMAIN       (1 << HPACB DOMAIN)
#define HPAC FLAGS        (1 << HPACB FLAGS)
#define HPAC END          (1 << HPACB_END)
#define HPAC START        (1 << HPACB_START)

```

```

#define HPAC_CTRL          (1 << HPACB_CTRL)
#define HPAC_TIME           (HPAC_START + HPAC_END)
#define HPAC_RUN             (HPAC_CTRL + HPAC_END)
#define HPAC_INPUT            (HPAC_CTRL + HPAC_TIME + HPAC_FLAGS + \
                           HPAC_DOMAIN + HPAC_OPTIONS + HPAC_MATERIAL)
#define HPAC_RESTART          (HPAC_CTRL + HPAC_TIME + HPAC_FLAGS + \
                           HPAC_DOMAIN + HPAC_OPTIONS + HPAC_MATERIAL)
#define HPAC_COMPLETE          (HPAC_INPUT + HPAC_RELEASE + HPAC_WEATHER)

#define HCB_FILE              0
#define HCB_SEARCH             1
#define HCB_APPEND             2
#define HCB_REPLACE             3

#define HC_FILE          (1 << HCB_FILE)
#define HC_SEARCH          (1 << HCB_SEARCH)
#define HC_APPEND          (1 << HCB_APPEND)
#define HC_REPLACE          (1 << HCB_REPLACE)

#define HCB_UTM               0
#define HCB_LLA                1

#define HC_UTM          (1 << HCB_UTM)
#define HC_LLA            (1 << HCB_LLA)

#define HUB_TIME              10
#define HUB_SPACE             11
#define HUB INCIDENT           12
#define HUB_LASTCHANCE         20
#define HUB_ENVIRONMENT        21
#define HUB_MATERIAL            22

#define HU_TIME          (1 << HUB_TIME)
#define HU_SPACE          (1 << HUB_SPACE)
#define HU INCIDENT           ((1 << HUB INCIDENT) + HU_SPACE + HU_TIME)
#define HU_LASTCHANCE         (1 << HUB_LASTCHANCE)
#define HU_ENVIRONMENT        (1 << HUB_ENVIRONMENT)
#define HU_MATERIAL            (1 << HUB_MATERIAL)

#define HS_INVALID             0
#define HS_VALID               1

#define HSB_COPYSCN            0
#define HSB_HASTERRAIN          1
#define HSB_HASDOS              2
#define HSB_HASDEP              3
#define HSB_HASPUFFS            10

#define HS_COPYSCN          (1 << HSB_COPYSCN)
#define HS_HASTERRAIN          (1 << HSB_HASTERRAIN)
#define HS_HASDEP              (1 << HSB_HASDEP)
#define HS_HASDOS              (1 << HSB_HASDOS)
#define HS_HASPUFFS            (1 << HSB_HASPUFFS)

```

```

#define HDB_METFILE      0
#define HDB_TERFILE      1
#define HDB_RELFILE      2
#define HDB_SAMFILE      3
#define HDB_INPFILE      4
#define HDB_MSCFILE      5
#define HDB_SCNFILE      6
#define HDB_RADFILE      7
#define HDB_PRJFILE      8
#define HDB_PUFFILE      9
#define HDB_DOSFILE      10
#define HDB_DEPFILE      11
#define HDB_LOGFILE      12
#define HDB_ATPFILE      13
#define HDB_MCWFILE      14
#define HDB_PCEFILE      15
#define HDB_SMPFILE      16
#define HDB_ICDFILE      17

#define HD_MAXBITS        17
#define HD_METFILE        (1 << HDB_METFILE)
#define HD_TERFILE        (1 << HDB_TERFILE)
#define HD_RELFILE        (1 << HDB_RELFILE)
#define HD_SAMFILE        (1 << HDB_SAMFILE)
#define HD_INPFILE        (1 << HDB_INPFILE)
#define HD_MSCFILE        (1 << HDB_MSCFILE)
#define HD_SCNFILE        (1 << HDB_SCNFILE)
#define HD_RADFILE        (1 << HDB_RADFILE)
#define HD_PRJFILE        (1 << HDB_PRJFILE)
#define HD_PUFFILE        (1 << HDB_PUFFILE)
#define HD_DOSFILE        (1 << HDB_DOSFILE)
#define HD_DEPFILE        (1 << HDB_DEPFILE)
#define HD_LOGFILE        (1 << HDB_LOGFILE)
#define HD_ATPFILE        (1 << HDB_ATPFILE)
#define HD_MCWFILE        (1 << HDB_MCWFILE)
#define HD_PCEFILE        (1 << HDB_PCEFILE)
#define HD_SMPFILE        (1 << HDB_SMPFILE)
#define HD_ICDFILE        (1 << HDB_ICDFILE)
#define HD_INPUT           (HD_INPFILE + HD_MSCFILE + HD_SCNFILE)
#define HD_OUTPUT          (HD_PRJFILE + HD_PUFFILE + HD_DOSFILE + \
                           HD_DEPFILE + HD_LOGFILE + \
                           HD_PCEFILE + HD_SMPFILE)

#define CALLBACK_ERROR     0x28000000
#define FILEMGR_ERROR      0x24000000
#define FORTRAN_IOS_ERROR   0x30000000

/*****************
* type_fd.f90
*****************/
#define HP_BON            0

```

```

#define HP_BAREA          1
#define HP_BEXPECT         2

#define HP_OFF            0
#define HP_ON             (1 << HP_BON)
#define HP_AREA            (1 << HP_BAREA)
#define HP_EXPECT           (1 << HP_BEXPECT)

#define HP_NUMTYP          4
#define HP_MEAN            1
#define HP_PROB             2
#define HP_EXCEED           3
#define HP_VARIANCE         4

#define TYPE_STRING_MEAN      "Mean Value ( M )"
#define TYPE_STRING_PROB      "Probability ( P[v>E] )"
#define TYPE_STRING_EXCEED    "Exceedance ( v[Pc>P] )"
#define TYPE_STRING_VARIANCE "Variance ( V )"

/********************* version_fd.f90 ********************/
#define HVB_VERSION20        18
#define HVB_VERSION30        20

#define HV_VERSION20          (1 << HVB_VERSION20)
#define HV_VERSION30          (1 << HVB_VERSION30)

/********************* Value Constants ********************/
#define INTEGER_DEFAULT     ((1 << 16) - 1)
#define INTEGER_DEFERRED    (2 * INTEGER_DEFAULT)
#define INTEGER_NOTSET      ((-1) * INTEGER_DEFAULT)

#define REAL_DEFAULT        1.0E+36F
#define REAL_DEFERRED       (2 * REAL_DEFAULT)
#define REAL_NOTSET         ((-1.0F) * REAL_DEFAULT)

/********************* TYPE: HPACCallbackProc ********************/
/* NOTES:
 *   The second parameter can be a MessageT *, and int
 *   value, or not used.
 */
typedef int              (__stdcall *HPACCallbackProc)(
                           int caller_id,
                           int message_id,

```

```

        int param
    );

/*
*      Functions
*/
*****



/*
*      FUNCTION:          HPACButton()                      *
*      (hpactool_mod.f90)                                *
*/
int
HPACButton( int caller_id, int button_id ) ;




/*
*      FUNCTION:          HPACCheckInput()                  *
*      (hpactool_mod.f90)                                *
*/
int
HPACCheckInput(
    int *           caller_id,
    int *           request_type,
    int *           input0,
    int *           input1
);





/*
*      FUNCTION:          HPACContourCount()                *
*      (hpactool_mod.f90)                                *
*/
int
HPACContourCount(
    int *           caller_id,
    int *           grid_id,
    HPACPlotFieldT *   field,
    HPACPlotTypeT *   plot_type,
    HPACContourHeaderT * contour_head,
    HPACContourElementT * contour_list,
    int *           mode,
    int *           rline,
    int *           point_count
);





/*
*      FUNCTION:          HPACContourField()               *
*      (hpactool_mod.f90)                                *
*/
int

```

```

HPACContourField(
    int *           caller_id,
    int *           grid_id,
    HPACPlotFieldT * field,
    HPACPlotTypeT * plot_type,
    HPACContourHeaderT * contour_head,
    HPACContourElementT * contour_list,
    int *           mode,
    HPACLineT *     line,
    HPACPointT *    points
);

/*****************
*      FUNCTION:      HPACCountMaterial()          *
*      (hpactool_mod.f90)                         *
*****************/
int
HPACCountMaterial(
    int *           caller_id,
    Char128T *     file,
    int *           material_count
);

/*****************
*      FUNCTION:      HPACCountRelease()          *
*      (hpactool_mod.f90)                         *
*****************/
int
HPACCountRelease(
    int *           caller_id,
    Char128T *     file,
    int *           release_count
);

/*****************
*      FUNCTION:      HPACCCreateField()          *
*      (hpactool_mod.f90)                         *
*****************/
int
HPACCCreateField(
    int *           caller_id,
    HPACPlotFieldT * field,
    float *         class_data
);

/*****************
*      FUNCTION:      HPACCurrentTerrain()        *
*      (hpactool_mod.f90)                         *
*****************/

```

```

int
HPACCurrentTerrain( HPACPointT * location, float * height );



/* *****
*      FUNCTION:          HPACCurrentWeather()           *
*      (hpactool_mod.f90)                                *
***** */
int
HPACCurrentWeather(
    HPACPPlotFieldT *      location,
    EnvironmentT *         environment,
    EnviroBLT *            boundary_layer
);





/* *****
*      FUNCTION:          HPACDefaultInput()           *
*      (hpactool_mod.f90)                                *
***** */
int
HPACDefaultInput(
    int *                  caller_id,
    int *                  request_type,
    int *                  out0,
    int *                  out1
);





/* *****
*      FUNCTION:          HPACDeleteField()           *
*      (hpactool_mod.f90)                                *
***** */
int
HPACDeleteField( int * caller_id, int * grid_id );






/* *****
*      FUNCTION:          HPACDeleteProject()          *
*      (hpactool_mod.f90)                                *
***** */
int
HPACDeleteProject( int * caller_id, ProjectIDT *, int request );






/* *****
*      FUNCTION:          HPACEExitTool()             *
*      (hpactool_mod.f90)                                *
***** */
int
HPACEExitTool();

```

```

*****  

*      FUNCTION:          HPACGetAPIVersion()          *  

*      (hpactool_mod.f90)          *  

*****  

int  

HPACGetAPIVersion( );  

*****  

*      FUNCTION:          HPACGetConcentration()        *  

*      (hpactool_mod.f90)          *  

*****  

int  

HPACGetConcentration( InteractiveSensor * conc );  

*****  

*      FUNCTION:          HPACGetField()           *  

*      (hpactool_mod.f90)          *  

*****  

int  

HPACGetField(  

    int *             caller_id,  

    int *             grid_id,  

    HPACPlotFieldT *   field,  

    HPACPlotTypeT *   plot_type,  

    HPACPlotFieldNodeT * nodes,  

    HPACPlotFieldTriangleT * triangles  

);  

*****  

*      FUNCTION:          HPACGetFieldDomain()         *  

*      (hpactool_mod.f90)          *  

*****  

int  

HPACGDomainetField(  

    int *             caller_id,  

    int *             grid_id,  

    int *             nx0,  

    int *             ny0,  

    float *            x0,  

    float *            y0,  

    float *            dx0,  

    float *            dy0  

);  

*****  

*      FUNCTION:          HPACGetFieldMinMax()        *  

*      (hpactool_mod.f90)          *  

*****  

int

```

```

HPACGetFieldMinMax(
    int *           caller_id,
    int *           grid_id,
    HPACPlotTypeT * plot_type,
    float *         mmin,
    float *         mmax,
    float *         vmin,
    float *         vmax,
    float *         fmin,
    float *         fmax
);

/*********************************************
*      FUNCTION:      HPACGetFieldSize()      *
*      (hpactool_mod.f90)                    *
*****************************************/
int
HPACGetFieldSize(
    int *           caller_id,
    int *           grid_id,
    HPACPlotFieldT * field,
    int *           nnodes,
    int *           ntriangles
);

/*********************************************
*      FUNCTION:      HPACGetFieldTable()     *
*      (hpactool_mod.f90)                    *
*****************************************/
int
HPACGetFieldTable(
    int *           caller_id,
    HPACPlotFieldT * field,
    float *         class_data,
    Char32T *       table_title,
    Char32T *       col_title,
    Char32T *       row_title,
    int *           table_data
);

/*********************************************
*      FUNCTION:      HPACGetFieldTableSize()  *
*      (hpactool_mod.f90)                    *
*****************************************/
int
HPACGetFieldTableSize(
    int *           caller_id,
    HPACPlotFieldT * field,
    float *         class_data,
    int *           ntables,

```

```

int *          ncols,
int *          nrows
);

/*****************
*      FUNCTION:      HPACGetFieldValue()
*      (hpactool_mod.f90)
*****************/
int
HPACGetFieldValue(
    int *          caller_id,
    int *          grid_id,
    HPACPlotTypeT * plot_type,
    float *        xloc,
    float *        yloc,
    float *        field_value
);

/*****************
*      FUNCTION:      HPACGetFieldValues()
*      (hpactool_mod.f90)
*****************/
int
HPACGetFieldValues(
    int *          caller_id,
    int *          grid_id,
    HPACPlotTypeT * plot_type,
    int *          point_count,
    float *        xlocs,
    float *        ylocs,
    float *        field_values
);

/*****************
*      FUNCTION:      HPACGetLastError()
*      (hpactool_mod.f90)
*****************/
int
HPACGetLastError( MessageT * error );

/*****************
*      FUNCTION:      HPACGetLOS()
*      (hpactool_mod.f90)
*****************/
int
HPACGetLOS( InteractiveSensor * los );

```

```

*      FUNCTION:          HPACGetPlotClasses()                      *
*      (hpactool_mod.f90)                                         *
******/                                                      
int
HPACGetPlotClasses(
    int *           caller_id,
    ProjectIDT *   project,
    Char64T *       class_strings,
    Char64T *       choice_strings,
    Char64T *       kind_strings,
    HPACCATEGORYCLASST * cat_class_array, /* HP_NUMCAT cols */
    HPACCLASSCHOICET * class_choice_array,
    HPACFIELDCOORDINATET * project_coord
);

******/
*      FUNCTION:          HPACGetPlotTimes()                     *
*      (hpactool_mod.f90)                                         *
******/
int
HPACGetPlotTimes(
    int *           caller_id,
    ProjectIDT *   project,
    HPACTIMET *    puff_times,
    HPACTIMET *    surface_times,
    HPACTIMET *    met_times
);

******/
*      FUNCTION:          HPACGetProjectAudit()                  *
*      (hpactool_mod.f90)                                         *
******/
int
HPACGetProjectAudit(
    int *           caller_id,
    PAUDITT *       audit
);

******/
*      FUNCTION:          HPACGetProjectTerrain()                *
*      (hpactool_mod.f90)                                         *
******/
int
HPACGetProjectTerrain(
    int *           caller_id,
    PTERRAINHEADT * terrain,
    float *         ths,
    float *         tddx,
    float *         tddy
);

```

```

*****
*      FUNCTION:      HPACGetProjectTerrainHeader()          *
*      (hpactool_mod.f90)                                     *
*****
int
HPACGetProjectTerrainHeader(
    int *           caller_id,
    PTerrainHeadT * terrain,
    float *         ths,
    float *         tddx,
    float *         tddy
);

*****
*      FUNCTION:      HPACGetProjectPuff()                  *
*      (hpactool_mod.f90)                                     *
*****
int
HPACGetProjectPuff(
    int *           caller_id,
    PPuffHeadT *   puff,
    int *           time_id,
    PuffT *         puff_list,
    float *         aux_data
);

*****
*      FUNCTION:      HPACGetProjectPuffHeader()           *
*      (hpactool_mod.f90)                                     *
*****
int
HPACGetProjectPuffHeader(
    int *           caller_id,
    PPuffHeadT *   puff,
    int *           time_id
);

*****
*      FUNCTION:      HPACGetProjectVersion()              *
*      (hpactool_mod.f90)                                     *
*****
int
HPACGetProjectVersion(
    int *           caller_id,
    ProjectIDT *   project
);

```

```

*****
*      FUNCTION:      HPACGetSCIPUFFVersion()          *
*      (hpactool_mod.f90)                            *
*****
int
HPACGetSCIPUFFVersion();

*****
*      FUNCTION:      HPACGetSrfPoint()           *
*      (hpactool_mod.f90)                            *
*****
int
HPACGetSrfPoint( InteractiveSrf * srf );

*****
*      FUNCTION:      HPACGetSubstrates()        *
*      (hpactool_mod.f90)                            *
*****
int
HPACGetSubstrates( Char16T * substrates );

*****
*      FUNCTION:      HPACGetVersion()          *
*      (hpactool_mod.f90)                            *
*****
int
HPACGetVersion();

*****
*      FUNCTION:      HPACGetString()           *
*      (hpactool_mod.f90)                            *
*****
int
HPACGetString( int * flag, Char128T * string );

*****
*      FUNCTION:      HPACInitError()           *
*      (hpactool_mod.f90)                            *
*****
int
HPACInitError();

*****
*      FUNCTION:      HPACInitTool()            *
*      (hpactool_mod.f90)                            *
*****
int

```

```

HPACInitTool(
    int *           caller_id,
    HPACC callbackProc   callback,
    int *           request,
    LimitT *        limits,
    Char128T *      ini_file
);

/*****************
*     FUNCTION:      HPACLoadProject()          *
*     (hpactool_mod.f90)                      *
*****************/
int
HPACLoadProject(
    int *           caller_id,
    ProjectT *      project,
    MaterialT *     material_list,
    ReleaseT *      release_list
);

/*****************
*     FUNCTION:      HPACModifySensor()         *
*     (hpactool_mod.f90)                      *
*****************/
int
HPACModifySensor( SensorT * sensor );

/*****************
*     FUNCTION:      HPACNewProject()          *
*     (hpactool_mod.f90)                      *
*****************/
int
HPACNewProject(
    int *           caller_id,
    CreateNewT *    project,
    MaterialT *     material_list,
    ReleaseT *      release_list
);

/*****************
*     FUNCTION:      HPACNumPlotClasses()       *
*     (hpactool_mod.f90)                      *
*****************/
int
HPACNumPlotClasses(
    int *           caller_id,
    ProjectIDT *   project,
    int *           class_count,
    int *           choice_count,

```

```

int * kind_count
);

/********************* FUNCTION: HPACNumPlotTimes() ********************
*   (hpactool_mod.f90)
****************************/
int
HPACNumPlotTimes(
    int * caller_id,
    ProjectIDT * project,
    int * puff_time_count,
    int * surface_time_count,
    int * met_time_count,
    int * rad_surf_grid_sub_times
);

/********************* FUNCTION: HPACNumSubstrates() ********************
*   (hpactool_mod.f90)
****************************/
int
HPACNumSubstrates();

/********************* FUNCTION: HPACPopAreaField() ********************
*   (hpactool_mod.f90)
****************************/
int
HPACPopAreaField(
    int * caller_id,
    int * grid_id,
    HPACPlotFieldT * field,
    HPACPlotTypeT * plot_type,
    HPACContourHeaderT * contour_head,
    HPACContourElementT * contour_list
);

/********************* FUNCTION: HPACQuerySensor() ********************
*   (hpactool_mod.f90)
****************************/
int
HPACQuerySensor( SensorT * sensor );

/********************* FUNCTION: HPACRestartProject() ********************
*   (hpactool_mod.f90)
****************************/

```

```
*****
int
HPACRestartProject(
    int *                      caller_id,
    CreateRstT *               project
);

*****  

*      FUNCTION:          HPACRunProject() *
*      (hpactool_mod.f90) *
*****  

int
HPACRunProject(
    int *                      caller_id,
    PEndT *                    run
);

*****  

*      FUNCTION:          HPACSizeProject() *
*      (hpactool_mod.f90) *
*****  

int
HPACSizeProject(
    int *                      caller_id,
    ProjectIDT *              project,
    int *                      material_count,
    int *                      release_count
);

*****  

*      FUNCTION:          HPACTransform() *
*      (hpactool_mod.f90) *
*****  

int
HPACTransform(
    HPACFieldCoordinateT *    coord_in,
    HPACFieldCoordinateT *    coord_out,
    int *                     point_count,
    float *                   xpts,
    float *                   ypts
);

*****  

*      FUNCTION:          HPACTransformPt() *
*      (hpactool_mod.f90) *
*****  

int
HPACTransformPt(
    HPACFieldCoordinateT *    coord_in,
```

```

HPACFieldCoordinateT * coord_out,
int * point_count,
HPACPointT * points
);

/********************* FUNCTION: HPACWriteInput() ********************/
* (hpactool_mod.f90)
**************************** */

int
HPACWriteInput(
    int * caller_id,
    int * request,
    int * in0,
    int * in1
);

#endif defined(__cplusplus) || defined(c_plusplus )
};

#endif
#endif

```

CHAPTER 3

Detailed CORBA Services

There are several objects comprising the detailed CORBA services available from HPAC. Further there is a specific mechanism for specifying what services are available when the HPAC server process is launched.

Detailed services provide all the mechanisms necessary to exercise HPAC calculations and results to their fullest extent. This includes incident source model calculations to produce detailed releases from descriptions of operational incidents.

3.1 CONFIGURING HPAC SERVICES

The `mil.dtra.server.impl.ServerLauncher` class implements the HPAC service launcher. It reads a server configuration file in Java properties format. A Universal Resource Locator (URL) for this properties file must be specified as the `hpacserver.propsURL` system property, as shown in the launch scripts in Section 3.2. The `hpacserver.properties` file in the `server/` subdirectory of the standalone HPAC distribution either should be used as is or should be the basis for any customizations. A version resulting from an install under Windows into the `c:\hpac4` directory is listed below. Customizations include adding incident source model services as described in Chapter 6.

Property files are formatted as *key=value* pairs with spaces significant. A single property key and value must appear on a single line, or as shown above, a line may be escaped with a backslash character ('\').

There are three property key stems: `hpacserver.servers`, `hpacserver.rmiservers`, and `scipuffserver`. Keys beginning with `scipuffserver` specify various properties of the T&D engine. The other key stems are for enumerated lists of services to launch. Numbers at the end of the property keys are significant and must occur in sequence. That is, if there is no `hpacserver.servers.2` property, only two services will be launched, those specified by the `.0` and `.1` key extensions.

CORBA and Remote Method Invocation (RMI) services are specified by `hpacserver.servers` and `hpacserver.rmiservers`, respectively. Property values are specified in two parts with a delimiting comma. The first part of the value is a service name. Note the name is specified in this file for documentation purposes only. The name under which the service is bound in the naming service is specified by the `getDefau ltName()` method of the service implementations, as described in Section 6.1.2. The second part of the value is the full path to the factory service implementation.

There are additional system properties which must be specified when executing mil. dtra. hpac. server. impl. ServerLauncher. These are specified in Table 3.1. *Note all property key names are case sensitive.*

3.1.1 hpacserver.properties

```
#-----
#-      hpacserver.servers (CORBA servers)
#-----
hpacserver.servers.0=\
ScipuffFactory,\n
mil.dtra.hpac.server.scipuff.impl.ScipuffServerFactoryImpl

hpacserver.servers.1=\
MaterialServer,\n
mil.dtra.hpac.material.server.impl.MaterialServerImpl

hpacserver.servers.2=\
RadFileMerger,\n
mil.dtra.hpac.server.radfile.impl.RadFileMergerImpl

hpacserver.servers.3=\
CBFacWeaponFactory,\n
mil.dtra.hpac.models.CBFac.server.impl.CBFacServerFactoryImpl

hpacserver.servers.4=\
ChemBioWeaponFactory,\n
mil.dtra.hpac.models.cbwpn.server.impl.ChemBioWeaponServerFactoryImpl

hpacserver.servers.5=\
SmokeWeaponFactory,\n
mil.dtra.hpac.models.swpn.server.impl.SmokeWeaponServerFactoryImpl

hpacserver.servers.6=\
MissileInterceptFactory,\n
mil.dtra.hpac.models.mint.server.impl.MissileInterceptServerFactoryImpl

hpacserver.servers.7=\
NWIFactory,\n
mil.dtra.hpac.models.nwi.server.impl.NWIServerFactoryImpl

hpacserver.servers.8=\
NfacServerFactory,\n
mil.dtra.hpac.models.nfac.server.impl.NfacServerFactoryImpl

hpacserver.servers.9=\
NwpnServerFactory,\n
mil.dtra.hpac.models.nwpn.server.impl.NwpnServerFactoryImpl

hpacserver.servers.10=\
StcalcServer,\n
mil.dtra.hpac.server.stcalc.impl.StcalcServerImpl
```

```

hpacserver.servers.11=\
RpnpServerFactory,\

mil.dtra.hpac.models.rwpn.server.impl.RWPNServerFactoryImpl

hpacserver.servers.12=\
IHPACServerFactory,\

mil.dtra.hpac.ihpacserver.server.impl.HPACServerFactoryImpl

#-----
#-      hpacserver.rmiservers (RMI servers) -
#-----
hpacserver.rmiservers.0=\
FileServer,\

mil.dtra.hpac.server.impl.FileServerImpl

#-----
#-      This is a list of effects dlls to be loaded by HPAC tool at -
#-      plot time. -
#-----
scipuffserver.effectsdlls=FxCoda.dll,Obscure.dll

#-----
#-      The number of seconds to wait until killing the server-side -
#-      per-client JVM when the client is closed in standalone mode. -
#-      Should be at least 1 second. -
#-----
scipuffserver.JVMkilldelay=10

#-----
#-      The number of seconds to wait for a per-client ScipuffServer to -
#-      be launched successfully. On a typical machine, this launch -
#-      should occur within a few seconds, so about 30 seconds should -
#-      be more than enough. However, not allowing enough time for the -
#-      server to launch will result in very unhappy users. -
#-----
scipuffserver.ServerLookupWaitSeconds=60

#-----
#-      scipuffserver properties -
#-      (Certain sciupffserver properties moved here instead of -
#-      appearing as system properties on the command line to reduce -
#-      the length of the command line for Win9x's needs.) -
#-      Each of the scipuffserver.xxxDir properties is a full URL -
#-      specification of a directory to write to the Hpac.INI file. -
#-----
scipuffserver.SciDataDir=file:///c:/hpac4/server/data/scipuff
scipuffserver.SfcClimoDir=file:///c:/hpac4/server/data/sfcclimo,\
file:///D:/sfcclimo

```

```

scipuffserver.UaClimoDir=file:///c:/hpac4/server/data/uaclimo, \
file:///D:/uaclimo
scipuffserver.PoplibDataDir=file:///c:/hpac4/client/shared/populate, \
file:///c:/hpac4/server/data/populate, \
file:///e:/populate, \
file:///D:/populate
scipuffserver.LandUseDataFile= \
file:///c:/hpac4/server/data/scipuff/landuse.dat

```

3.2 LAUNCHING HPAC SERVICES

The HPAC-4.0.1 distribution does not include a facility for launching HPAC services in a separate process. Listed below are a Windows NT/2000/XP Command Prompt batch file for launching HPAC services and Unix/Linux scripts for service launch and termination. (The termination script finds the process list command for Linux and Sun Solaris. It should be modified as necessary for other Unix variants.) Services are terminated under Windows by typing **Ctrl-C** in the Command Prompt window from which the batch file is executed.

3.2.1 Windows Command Prompt launch batch file

```

@echo on
rem -----
rem -   NAME:           server.run.bat
rem -   PURPOSE:
rem -           Hazard Prediction and Assessment Capability (HPAC)
rem -           Version 4.0
rem -           January 2001
rem -----
rem This DOS script launches HPAC server in remote mode.

rem -----
rem -   Read environment, start in server/ subdirectory
rem -----
call ..\setpaths
cd %HPAC_server_DIR%

rem -----
rem -   Environment Input
rem -----
set path=%HPAC_server_DIR%\bin;%JRE_HOME%\bin;%path%
set ExtPath=%HPAC_server_DIR%\ext;%HPAC_shared_DIR%\ext;%JRE_HOME%\lib\ext

set ServerHost=172.16.24.2
rem set ServerHost=134.167.10.72
set ServerPort=1410

```

| Property | Req? | Description |
|------------------------------------|-------------|---|
| <i>hpac.logLevel</i> | N | specifies log verbosity level where higher numbers indicate more log messages (defaults to 0) |
| <i>hpac.stderr</i> | N | specifies path to file to hold server launcher log messages |
| <i>hpacserver.dataDir</i> | Y | Filesystem path to the server data directory |
| <i>hpacserver.dataURL</i> | Y | URL to the server data directory |
| <i>hpacserver.libDir</i> | Y | Path to directory containing shared objects and DLLs |
| <i>hpacserver.projectRootDir</i> | Y | Path to root directory containing per-user directories for server project files |
| <i>hpacserver.propsURL</i> | Y | URL for the server properties file (e.g., <i>hpacserver.properties</i>) |
| <i>hpacserver.rootURL</i> | Y | URL to the server root directory |
| <i>java.naming.factory.initial</i> | Y | “com.sun.jndi.cosnaming.CNCtxFactory” |
| <i>java.naming.provider.url</i> | Y | “iiop://ServerHost:ServerPort”, specifies naming service |
| <i>scipuffserver.stderr</i> | N | Path to file for log messages from per-client T&D calculations |
| <i>scipuffserver.stdout</i> | N | Path to file for non-log messages from per-client T&D calculations |
| <i>scipuffserver.verbose</i> | N | Turns on log messages for per-client T&D calculations |
| <i>scipuffserver.verboseTiming</i> | N | Adds timing log messages for per-client T&D calculations |

Table 3.1: Server system properties

```

setlocal

rem -----
rem - Run CORBA naming service
rem -----
echo Starting CORBA Naming Service...
start "CORBA Naming Service" tnameserv -ORBInitialPort %ServerPort%      ->
-ORBServerHost %ServerHost%


rem -----
rem - Launch NWPN strikeserver
rem -----
cd %HPAC_DIR%\strikeserver

set classpath=strikeserver.jar;nwpnglobal.jar

set S01=-Djava.rmi.server.hostname=%ServerHost%
set S02=-Djava.rmi.server.codebase=%HPAC_URL%/strikeserver/strikeserver.jar
set S03=-Djava.security.policy=strikeserver.policy

set S99=mil.dtra.hpac.models.nwpn.strikeserver.StrikeServerFactory

start "NWPN Strikeserver" java %S01% %S02% %S03% %S99%
set classpath=

rem -----
rem - If you have cygwin installed or sleep command, wait for the
rem - naming service to launch
rem -----
sleep 5


rem -----
rem - Run server launcher
rem -----
echo Starting Server Launcher

cd %HPAC_server_DIR%

set S01=-Djava.ext.dirs=%ExtPath%
set S02=-Dhpacserver.dataDir=%HPAC_server_DIR%\data
set S03=-Dhpacserver.dataURL=%HPAC_server_URL%/data
set S04=-Dhpacserver.libDir=%HPAC_server_DIR%\bin
set S05=-Dhpacserver.projectRootDir=%HPAC_server_DIR%\users
set S06=-Dhpacserver.propsURL=%HPAC_server_URL%/hpacserver.properties
set S07=-Dhpacserver.rootURL=%HPAC_server_URL%
set S08=-Djava.naming.factory.initial=com.sun.jndi.cosnaming.CNCtxFactory
set S09=-Djava.naming.provider.url=iiop://%ServerHost%:%ServerPort%/
set S09a=-Djava.rmi.server.hostname=%ServerHost%
set S10=-Djava.security.policy=%HPAC_server_URL%/java.policy

```

```

set S11=-Dscipuffserver.verbose=true -Dscipuffserver.verbosetiming=false
set S12=-Dhpac.stderr=server.log -Dhpac.logLevel=5

set S99=mil.dtra.hpac.server.impl.ServerLauncher

set logfile=-Dscipuffserver.stdout=scipuffserver.out
set errfile=-Dscipuffserver.stderr=scipuffserver.err

rem -----
java %S01% %S02% %S03% %S04% %S05% %S06% %S07% %S08% %S09% %S09a%
%S10% %S11% %S12% %logfile% %errfile% %S99%           ->

endlocal

```

3.2.2 Unix/Linux Bourne shell launch script

```

#!/bin/sh -a
#-----
#-      NAME:          HPACServer.start
#-      PURPOSE:
#-                  Launch the HPAC server processes
#-----

#-----
#-      Environment Input
#-----
#JRE_HOME=/usr/local/jdk1.3/jre
JRE_HOME=/usr/local/j2sdk1.4.0/jre
HPAC_CD_Dir=/mnt/cdrom/

ServerDir='dirname $0'
if [ "$ServerDir" = "." ]; then
  ServerDir='pwd'
fi

ServerHost='hostname'
ServerPort=1400

#-----
#-      Derived Environment
#-----
RootDir="$ServerDir/.."
ServerURL="file:///$ServerDir"
PATH="$ServerDir/bin:$JRE_HOME/bin:$PATH"
LD_LIBRARY_PATH="$ServerDir/lib:${LD_LIBRARY_PATH:-}"

ExtDirs="$ServerDir/ext:$RootDir/shared/ext:$JRE_HOME/lib/ext"

#-----
#-      Run CORBA Naming Service
#-----

```

```

#-----
cp /dev/null server.pids

echo "Starting CORBA Naming Service..." 
tnameserv -ORBInitialPort $ServerPort -ORBServerHost $ServerHost &
echo "$!" >> server.pids
sleep 5

#-----
#-      Launch NWPN StrikeServer
#-----
echo "Starting StrikeServer..." 
cd $RootDir/strikeserver

export CLASSPATH
CLASSPATH='pwd'/strikeserver.jar:'pwd'/nwpnglobal.jar

java \
-Djava.rmi.server.hostname=$ServerHost \
-Djava.rmi.server.codebase=$RootDir/strikeserver/strikeserver.jar \
-Djava.security.policy=strikeserver.policy \
mil.dtra.hpac.models.nwpn.strikeserver.StrikeServerFactory &
echo "$!" >> server.pids

#-----
#-      Run Server Launcher
#-----
echo "Starting Server Launcher..." 
cd $ServerDir
java \
-Djava.ext.dirs="$ExtDirs" \
-Dhpacserver.dataDir="$ServerDir/data" \
-Dhpacserver.dataURL="$ServerURL/data" \
-Dhpacserver.installRoot="$HPAC_CD_Dir" \
-Dhpacserver.libDir="$ServerDir/lib" \
-Dhpacserver.projectRootDir="$ServerDir/users" \
-Dhpacserver.propsURL="$ServerURL/hpacserver.properties" \
-Dhpacserver.rootURL="$ServerURL" \
-Djava.naming.factory.initial=com.sun.jndi.cosnaming.CNCtxFactory \
-Djava.naming.provider.url=iiop://{$ServerHost}:{$ServerPort}/ \
-Djava.security.policy=$ServerURL/java.policy \
-Dscipuffserver.verbose=true \
-Dscipuffserver.verbosetiming=false \
-Dhpac.logLevel=5 \
-Dscipuffserver.stdout=scipuffserver.out \
-Dscipuffserver.stderr=scipuffserver.err \
mil.dtra.hpac.server.impl.ServerLauncher $* &

echo "$!" >> server.pids

echo "Server Launched"

```

3.2.3 Unix/Linux Bourne shell terminate script

```
#!/bin/sh -a
#-----
#-      NAME:          HPACServer.stop
#-      PURPOSE:        -
#-                  Stop all HPAC services
#-----

#-----
#-      Determine process list Command
#-----
System='uname'
if [ "$System" = "Linux" ]; then
    PsCommand="ps ax"

elif [ "$System" = "SunOS" ]; then
    case `uname -r` in
        4*)
            PsCommand="ps ax"
            ;;
        5*)
            PsCommand="ps -e"
    esac

else
    PsCommand="ps ax"
fi

#-----
#-      Kill Process, Remove PID File
#-----
Pids=`$PsCommand | fgrep -f server.pids | awk '{print $1}'`''
kill $Pids
/bin/rm -f server.pids
```

3.3 SERVICES OVERVIEW

There are two types of HPAC service objects. First are those which register with the naming service and thus can be accessed by name by clients. They are *singletons* in that only one of them ever exists. Some of this first kind provide a specific capability, such as the CORBA MaterialServer, RadFileMerger, and StcalcServer and RMI FileServer. The remaining singleton server objects are incident source model factory servers that create per-client server objects as illustrated in Section 1.1.2.

These per-client servers are the second type of HPAC service objects. They exist solely to provide services to a specific client and are relatively short-lived. They are not registered in the naming service and can only be obtained via their corresponding factory server, which creates them upon request. When the requesting client completes a session of activity with its per-client server, the server is released. Thus, no state is maintained between client request for a per-client server. Figure 3.1 depicts the available HPAC server objects.

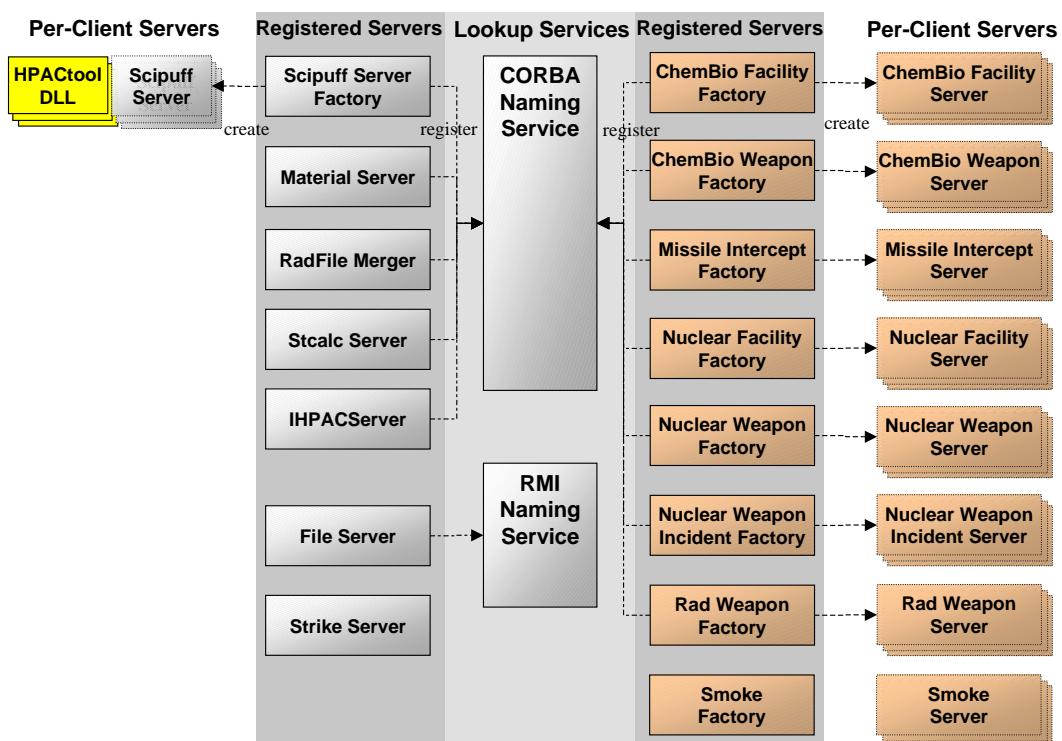


Figure 3.1: Registered and per-client servers

3.3.1 Naming services

Before HPAC services are started, a CORBA Naming Service process is launched, as illustrated in the scripts of Section 3.2. Specifically, the `tnameserv` executable in the J2SE distribution is launched on the machine on which the script is run with the port specified as the `-ORBInitialPort` command line option. Note the `-ORBServerHost` option is used only to distinguish between multiple network interfaces on the host.

When the `ServerLauncher` class is loaded and executed, it creates an RMI registry on the next port after that used for the CORBA Naming Service. By default, these are ports 1400 and 1401, respectively. All client access to factory and other registered objects occurs via these services.

3.3.2 Incident Source Models

Each incident source model provides a factory and a per-client server. Factory servers may provide additional services, but their primary purpose is to create per-client server objects on demand. Model computations are performed by per-client servers.

The purpose of an incident source model is to take an operational description of an incident and produce one or more detailed releases suitable for input to the T&D calculation engine, represented by the `ScipuffServer` object. Data objects describing an incident, its releases, and the material for each release are specified in Section 3.4.

Described more fully in Chapter 6 is the framework for incident source models. The super interfaces for model server and model factory interfaces are defined in the `server` IDL module listed in Section 3.4.5. Its `IncidentModelServerFactory` interface defines a single method, `getInstance()`, as required for all factory objects. Source models are free to add additional methods, but many do not and thus do not extend this interface.

The `IncidentModelServer` interface is the methods a source model server must implement or inherit from an abstract base class implementation provided in HPAC. Some of the defined methods are for calls to the model from HPACtool through `ScipuffServer`. Methods by which clients collaborate with model servers are described in the following sequence of steps.

1. Obtain a reference to the model factory object by looking it up in the naming service. Model IDL modules should provide a constant giving the name by which the factory is registered. If such a constant is not provided, use the name specified in the `hpacserver.properties` file.
2. Obtain a model server reference by calling the `getInstance()` method of the factory object.
3. Call the `initIncident()` method to obtain a default incident description and releases for this model. Note this method returns a standard `IncidentT` object as well as a model-specific object which should be defined in the model's IDL module. Note the entire state of the incident is contained in these two objects.
4. After modifying the objects describing the incident, call `updateIncident()` to have the model regenerate the releases. Calling this method is optional but likely necessary to compute anything of interest.

| Model | Description |
|-----------------------|--|
| ChemBioFacility | Conventional weapon attack against a chemical or biological facility |
| ChemBioWeapon | Chemical or biological delivery weapons |
| MissileIntercept | In-flight intercept of missile with chemical or biological payload |
| NuclearWeaponIncident | Accident or incident involving a nuclear device |
| NuclearFacility | Accident or incident at a nuclear facility |
| NuclearWeapon | Nuclear weapon or strike |
| RadiologicalWeapon | Radiological device or weapon |
| SmokeMunition | Smoke and obscurants |

Table 3.2: Incident source models

5. When no more update calls will be made, call the `terminate()` method and the standard CORBA object `_release()` method to release the resources associated with the model server.

Each incident source model is specified in sections below. Available source models are identified in Table 3.2.

3.3.3 Interface Definition Language

With the exception of the FileServer, all HPAC services are specified in CORBA IDL.

3.4 DATA OBJECTS

There are a plethora of IDL structures representing data which must be provided to the T&D calculation engine (ScipuffServer object) for an HPAC scenario to be executed. IDL files defining the structures and constants are listed below. Detailed HPAC services use structures for inputs instead of interfaces in order to reduce the amount of communication between the client and server. A client gathers and assigns the input structures and then makes a single server method call. Assignment to structure fields replaces calls to a server object accessor methods to set attribute values.

Note: In version 4.0.1 of HPAC, ScipuffServer does not expose the `HPACDefaultInput()` function available in HPACtool. This leaves the responsibility for providing values for all the inputs to ScipuffServer calculation methods. Examples of default values for the various structures can be found in the example programs in Appendix B.

This is rectified in version 4.0.2. Further, IHPACServer objects provide a higher level interface which assumes default values for most of the ScipuffServer inputs.

3.4.1 files.idl

```
#ifndef __files__
#define __files__
//-----
//      NAME:          files.idl
//-----

//-----
//      MODULE:         mil.dtra.hpac.server.files
//*****-
* Definition of <tt>FileReferenceT</tt> for handling client-to-server
* file uploads and references.  Packaged in <tt>fileutils.jar</tt>.
*
* <h2>FileReferenceT</h2>
* <p>
* This structure/class is an encapsulation of all the ways data may
* be passed from the client to the server.  There are four distinct
* states with different meanings on the client and server.  On the client
* an instance is created with a tag indicating a client path, server
* path, or a URL.  At resolution time, server path and URL are passed
* as is to the server.  A client path is handled in one of three
* ways.  In standalone mode, it is converted to a server path without
* change.  In remote mode, a file whose contents are no bigger than
* the maximum content size (<tt>MAX_CONTENTS_SIZE</tt>) is read, and
* its contents are passed as an octet sequence and converted
* to a contents state.  If the file is larger,
* it is uploaded and converted to a server path.
* </p>
*
* <p>The five states or tags are:</p>
* <p><dl>
*   <p><dt>Client path
*   <dd>Only valid on the client, this state will result in a conversion
*       to either server path or contents.  In standalone mode, the
*       server path conversion is a simple copy of the client path.  In
*       remote mode, a file whose contents are large than
*       <tt>MAX_CONTENTS_SIZE</tt> is uploaded to the server and converted
*       to a server path.  If the file is within the max size, it is
*       read, and its contents are stored in an octet sequence with a
*       contents tag.
*   </p>
*   <p><dt>Contents
*   <dd>A client path may be converted to this tag just before being
*       passed to the server.  The file contents are stored in an
*       octet sequence.
*   </p>
*   <p><dt>Deferred
*   <dd>Represents a file reference which must deferred until a later
```

```

*      time, such as availability of met in a model server.
*  </p>
*  <p><dt>Server path
*  <dd>A client may tag a file reference as a server path originally,
*  indicating a file that resides permanently on the server. An
*  original client path may be converted to a server path as either
*  an uploaded file or, in standalone mode, a copy
*  of a client path.
*  </p>
*  <p><dt>URL
*  <dd>URLs are assumed to be accessible to both client and server and
*  are passed as-is to the server. Note that "file" protocol URLs
*  will not work.
*  </p>
*  </dl></p>
*
*  <p>
*  The <tt>mil.dtra.hpac.server.fileutils.FileReferenceTMgr</tt> class serves
*  as manager for objects of this type. It provides factory methods
*  for creating instances of this structure and the
*  <tt>getServerFile()</tt> method
*  with which ScipuffServer can obtain the file content in a path
*  local to the server.
*  </p>
*
*  @see mil.dtra.hpac.server.fileutils.FileReference
*  @see mil.dtra.hpac.server.fileutils.FileReferenceTMgr
*****module mil { module dtra { module hpac { module server {
module files
{
    /**
     * Maximum size of a file for content
     * transfer via the sequence<octet> field
     * (262144)
     */
    const long          MAX_CONTENTS_SIZE = 262144;

//-----
//  File Reference Discriminator Values
//-----
    /**
     * File reference tag indicating a
     * deferred file specification
     */
    const long          FR_DEFERRED = -1;

    /**
     * File reference tag for a path local
     * to the client
     */
    const long          FR_CLIENT_PATH = 0;

```

```

        /**
         * File reference tag indicating the actual
         * file contents
         */
const long          FR_CONTENTS = 1;

        /**
         * File reference tag for a path on the
         * server for pre-existing files
         */
const long          FR_SERVER_PATH = 2;

        /**
         * File reference tag for data residing
         * at a URL
         */
const long          FR_URL = 3;

//-----
//      TYPE:          FileContentsT
//*****
struct FileContentsT
{
    string           fClientPath;
    sequence<octet> fContents;
}; // FileContentsT

//-----
//      TYPE:          FileServerPathT
//*****
* Representation of a server path. If the fClientPath field is
* empty (""), this specifies the path of a file originating on the
* server.
struct FileServerPathT
{
    string           fClientPath;
    string           fServerPath;
}; // FileServerPathT

//-----
//      TYPE:          FileReferenceT
//*****
* This structure/class is an encapsulation of all the ways data may
* be passed from the client to the server. There are four distinct
* states with different meanings on the client and server. On the client
* an instance is created with a tag indicating a client path, server
* path, or a URL. At resolution time, server path and URL are passed

```

```

* as is to the server. A client path is handled in one of three
* ways. In standalone mode, it is converted to a server path without
* change. In remote mode, a file whose contents are no bigger than
* the maximum content size (<tt>MAX_CONTENTS_SIZE</tt>) is read, and
* its contents are passed as an octet sequence and converted
* to a contents state. If the file is larger,
* it is uploaded and converted to a server path.
*
* <p>The five states or tags are:</p>
* <p><dl>
*   <p><dt>Client path
*     <dd>Only valid on the client, this state will result in a conversion
*       to either server path or contents. In standalone mode, the
*       server path conversion is a simple copy of the client path. In
*       remote mode, a file whose contents are large than
*       <tt>MAX_CONTENTS_SIZE</tt> is uploaded to the server and converted
*       to a server path. If the file is within the max size, it is
*       read, and its contents are stored in an octet sequence with a
*       contents tag.
*   </p>
*   <p><dt>Contents
*     <dd>A client path may be converted to this tag just before being
*       passed to the server. The file contents are stored in an
*       octet sequence.
*   </p>
*   <p><dt>Deferred
*     <dd>Represents a file reference which must deferred until a later
*       time, such as availability of met in a model server.
*   </p>
*   <p><dt>Server path
*     <dd>A client may tag a file reference as a server path originally,
*       indicating a file that resides permanently on the server. An
*       original client path may be converted to a server path as either
*       an uploaded file or, in standalone mode, a copy
*       of a client path.
*   </p>
*   <p><dt>URL
*     <dd>URLs are assumed to be accessible to both client and server and
*       are passed as-is to the server. Note that "file" protocol URLs
*       will not work.
*   </p>
* </dl></p>
*
* <p>
* The <tt>mil.dtra.hpac.server.fileutils.FileReferenceTMgr</tt> class serves
* as manager for objects of this type. It provides factory methods
* for creating instances of this structure and the
* <tt>getServerFile()</tt> method
* with which ScipuffServer can obtain the file content in a path
* local to the server.
* </p>
*
* @see mil.dtra.hpac.server.fileutils.FileReference

```

```

* @see mil.dtra.hpac.server.fileutils.FileReferenceTMgr
***** /
union FileReferenceT switch( long )
{
    case FR_CLIENT_PATH:
        string fClientPath;

    case FR_CONTENTS:
        FileContentsT fContents;

    case FR_DEFERRED:
        string fDeferred;

    case FR_SERVER_PATH:
        FileServerPathT fServerPath;

    case FR_URL:
        string fURL;
}; // FileReferenceT
}; // files
};

};

#endif

```

3.4.2 material.idl

```

#ifndef __material__
#define __material__
-----
//      NAME:          material.idl
//      -
//      -
//      -
//      MODULE:         mil.dtra.hpac.material.server
//      -
//      -
module mil { module dtra { module hpac { module material {
module server
{
//      -
//      EXCEPTION:      MaterialException
//      -
exception MaterialException
{
    string fMaterialName;
    string fMessage;
};

//      -
//      Material Constants
//      -

```

```

//-----
const long          GROUP_DEP_MASK = (0x01 << 0);
const long          GROUP_DOS_MASK = (0x01 << 1);
const long          TOTAL_DEP_MASK = (0x01 << 2);
const long          TOTAL_DOS_MASK = (0x01 << 3);

const long          HS_MAX_MATERIAL_BIN_SIZE = 50;

//-----
// Plot Type Constants
//-----
const string        HPAC_RADIATION_DOSE = "HPAC radiation dose";
const string        SURFACE_DEP = "surface deposition";
const string        SURFACE_DOSE = "surface dose";

//-----
// TYPE:           FloatArrayT
//-----
typedef sequence<float>  FloatArrayT;

//-----
// TYPE:           StringArrayT
//-----
typedef sequence<string>  StringArrayT;

//-----
// TYPE:           ContourDefT
//-----
struct ContourDefT
{
    string            fPlotType;
    string            fMaterialName;

    long              fCount;
    FloatArrayT       fValues;
    StringArrayT      fLabels;

    float             fScale;
    string            fUnits;
}; // ContourDefT

//-----
// TYPE:           ContourDefListT
//-----
typedef sequence<ContourDefT>  ContourDefListT;

//-----

```

```

//      TYPE:          HazardAreaDefT
//-----
struct HazardAreaDefT
{
    string          fPlotType;
    string          fMaterialName;

    long            fCount;
    FloatArrayT     fProbability;
    FloatArrayT     fExceeds;
    StringArrayT   fLabels;

    float           fScale;
    string          fUnits;
}; // HazardAreaDefT

//-----
//      TYPE:          HazardAreaDefListT
//-----
typedef sequence<HazardAreaDefT>  HazardAreaDefListT;

//-----
//      TYPE:          AerosolMaterialT
//-----
struct AerosolMaterialT
{
    float           fMinConcentration;
    float           fMaxDayDecay;
    float           fMinNightDecay;
    float           fNWPNDelay;
    long            fSavedFields;
    float           fGasDeposition;
    float           fLiquidDensity;
    float           fAntoineCoeffs[ 3 ];
    float           fMolecularWeight;
    float           fLiquidSpecificHeat;
    float           fGasSpecificHeat;
}; // AerosolMaterialT

//-----
//      TYPE:          GasMaterialT
//-----
struct GasMaterialT
{
    float           fMinConcentration;
    float           fMaxDayDecay;
    float           fMinNightDecay;
    float           fNWPNDelay;
    long            fSavedFields;
    float           fGasDeposition;
}

```

```

float fGasDensity;
}; // GasMaterialT

//-----
//      TYPE:          LiquidMaterialT
//-----
struct LiquidMaterialT
{
    float fMinConcentration;
    float fMaxDayDecay;
    float fMinNightDecay;
    float fNWPNDelay;
    long fSavedFields;
    float fGasDeposition;
    float fGasDensity;
    float fLiquidDensity[ 2 ];
    float fAntoineCoeffs[ 3 ];
    float fMolecularWeight;
    float fSurfaceTension;
    float fSpreadFactor;
    long fBinCount;
    FloatArrayT fBinBounds;
    float fViscosity;
}; // LiquidMaterialT

//-----
//      TYPE:          ParticleMaterialT
//-----
struct ParticleMaterialT
{
    float fMinConcentration;
    float fMaxDayDecay;
    float fMinNightDecay;
    float fNWPNDelay;
    long fSavedFields;
    float fDensity;
    long fBinCount;
    FloatArrayT fBinBounds;
    float fParticleSystemSize;
}; // ParticleMaterialT

// General Material
//

//-----
//      Material Type Constants
//-----
const long HM_GAS      = (0x01 << 0);
const long HM_PARTICLE = (0x01 << 1);

```

```

const long          HM_LIQUID      = ( 0x01 << 2 );
const long          HM_AEROSOL     = ( 0x01 << 3 );
const long          HM_WETPARTICLE = ( 0x01 << 4 );

const long          HM_2NDEVAP    = ( 0x01 << 20 );
const long          HM_NWPN       = ( 0x01 << 21 );
const long          HM_NFAC       = ( 0x01 << 22 );
const long          HM_HAZARD     = ( 0x01 << 23 );
const long          HM_MULTICOMP   = ( 0x01 << 24 );

//-----
//      TYPE:      MaterialDataT
//-----
union MaterialDataT
{
    switch( long )
    {
        case HM_AEROSOL:
            AerosolMaterialT fAerosol;

        case HM_GAS:
            GasMaterialT fGas;

        case HM_LIQUID:
            LiquidMaterialT fLiquid;

        case HM_PARTICLE:
            ParticleMaterialT fParticle;
    }; // MaterialDataT

//-----
//      TYPE: SupplementalPropsT
//-----
struct SupplementalPropsT
{
    string fKey;
    string fValue;
};

typedef sequence<SupplementalPropsT>
    SupplementalPropsListT;

typedef sequence<SupplementalPropsListT>
    SupplementalPropsListGroupT;

//-----
//      TYPE:      MaterialT
//-----
struct MaterialT

```

```

{
    string          fID;
    long           fTypeMask;
    string          fName;
    string          fLongName;
    string          fUnits;
    boolean         fMultiComp;
    long           fEffAvail;
    long           fEffClass;

    string          fAgentType;
    MaterialDataT   fMaterialData;
    ContourDefListT fContourDefs;
    HazardAreaDefListT fHazardAreaDefs;
    SupplementalPropsListGroupT fSupplementalProps;
}; // MaterialT

//-----
//      TYPE:          MaterialListT
//-----
typedef sequence<MaterialT>  MaterialListT;

//-----
//      Material Server Service Name
//-----
const string        MATERIAL_SERVICE_NAME = "MaterialServer";

//-----
//      ByteArray definition in IDL
//-----
typedef sequence<octet>    ByteArray;

//-----
//      INTERFACE:      MaterialServer
//-----
interface MaterialServer
{
//-----
//      METHOD:          getMaterial()
//      PURPOSE:
//-----
    MaterialT
    getMaterial( in string material_name )
        raises ( MaterialException );

//-----
//      METHOD:          getMaterialList ()
//      PURPOSE:         retrieve list of materials available through CORBA
//-----

```

```

//-----
StringArrayT
getMaterialList ()
    raises (MaterialException);

//-----
// METHOD:           gePropertiesByKey ()
// PURPOSE:        retrieve list of materials available through CORBA -
//-----
ByteArray
getPropertiesByKey( in string filename, in string prefix );
}; // interface MaterialServer

}; // material
}; }; }; };

#endif

```

3.4.3 project.idl

```

#ifndef __project__
#define __project__
//-----
//      NAME:          project.idl
//-----

//-----
//      MODULE:         mil.dtra.hpac.server.project
//-----
module mil { module dtra { module hpac { module server {
/*****************/
* Definitions of constants and structures from the HPACTool API related
* to projects.
* Model servers should not need these classes, so we put them in
* <tt>scipuffdefs.jar</tt>.
/*****************/
module project
{

//-----
//      Value Constants
//-----
        /**
         * Maximum latitude allowed in SCIPUFF
         */
const float      MAX_LATITUDE = 84.0;

        /**
         * Minimum latitude allowed in SCIPUFF
         */

```

```

const float MIN_LATITUDE = -80.0;

//-----
// Coordinate Mode Constants
//-----
/***
 * Lat-lon coordinate mode constant
 */
const long HD_LATLON = 1;
/***
 * Cartesian coordinate mode constant
 */
const long HD_CARTESIAN = 2;
/***
 * UTM coordinate mode constant
 */
const long HD_UTM = 3;
/***
 * ?? coordinate mode constant
 */
const long HD_METERS = 4;

//-----
// Scipuff Method constants
//-----
/***
 * SCIPUFF <em>dynamic</em> method bit mask
 */
const long HF_DYNAMIC = 0x01 << 0;
/***
 * SCIPUFF <em>dense gas</em> method bit mask
 */
const long HF_DENSE = 0x01 << 1;
/***
 * SCIPUFF <em>static</em> method bit mask
 */
const long HF_STATIC = 0x01 << 2;
/***
 * SCIPUFF <em>multipcomp</em> method bit mask
 */
const long HF_MULTICOMP = 0x01 << 3;

//-----
// Scipuff Mode constants
//-----
/***
 * SCIPUFF <em>fast</em> mode bit mask;
 * also passed as a mode value to
 * <tt>IncidentModelServer.updateRelease()</tt>
 */

```

```

const long          HF_FAST           = 0x01 << 0;
{
    /**
     * SCIPUFF <em>hazard</em> mode bit mask
    */
}
const long          HF_HAZARD        = 0x01 << 1;
{
    /**
     * SCIPUFF <em>dual</em> mode bit mask
    */
}
const long          HF_DUAL          = 0x01 << 2;

//-----
//      TYPE:          AuditT          -
//*****
/* Definition of project audit information.
*****/
struct AuditT
{
    string            fProjectTitle;
    string            fAnalyst;
    string            fClassification;
    string            fHPACVersion;
    string            fDate;
}; // struct AuditT

//-----
//      TYPE:          FlagsT          -
//*****
/* Flags for the calculation.
*****/
struct FlagsT
{
    /**
     * Flags restart of previous calculation
    */
    boolean           fRestart;
    /**
     * Method id (<tt>HF_DYNAMIC</tt>,
     * <tt>HF_DENSE</tt>, <tt>HF_STATIC</tt>)
    */
    long              fScipuffMethod;
    /**
     * Mode id (<tt>HF_FAST</tt>,
     * <tt>HF_HAZARD</tt>, <tt>HF_DUAL</tt>)
    */
    long              fScipuffMode;
    /**
     * Audit info
    */
    AuditT           fAudit;
}; // struct FlagsT

```

```

//-----
//      TYPE:          LimitT
//*****
* Definition of computational limits.
*****/
struct LimitT
{
    /**
     * Max number of puffs
     */
    long        fMaxPuffs;
    /**
     * Max number of grid cells per surface field
     */
    long        fMaxGridCellsPerSurface;
    /**
     * Max size in either direction of horizontal
     * met arrays
     */
    long        fMaxMetHorizSize;
}; // struct LimitT

//-----
//      TYPE:          OptionsT
//*****
* Options for the dispersion calculation.
*****/
struct OptionsT
{
    long        fVertGridTurbBL;
    long        fGridResolution;
    long        fSubstrateIndex;
    float       fTurbDiffAvgTime;
    float       fMinPuffMass;
    float       fMinAdaptiveGridSize;
    float       fTropVertVelVariance;
    float       fTropAvgDissRate;
    float       fTropVertScalelength;
    float       fCalmTurb;
    float       fCalmScaleLength;
    float       fDosageCalcHeight;
    float       fSamplerOutputInterval;
    string      fSamplerFile;
}; // struct OptionsT

//-----
//      TYPE:          SpatialDomainT
//*****
* Client view of a spatial domain. Set fComputeDefault true to tell the
* server to compute the domain from the releases.
*****/

```

```
***** */
struct SpatialDomainT
{
    boolean           fComputeDefault;
    float             fMaxLatitude;
    float             fMinLatitude;
    float             fMaxLongitude;
    float             fMinLongitude;
    float             fMaxHeight;
    float             fHorzResolution;
    float             fVertResolution;
}; // SpatialDomainT

//-----  

//      TYPE:          TimeT
//***** */
* Definition of a UTC time as defined in the HPACTool API.
***** */

struct TimeT
{
    short             fYear;
    short             fMonth;
    short             fDay;
    float             fHour;
};

//-----  

//      TYPE:          TemporalDomainT
//***** */
* Client view of a temporal domain. Times are absolute. Set fComputeDefault
* true to tell the server to compute the time domain from the releases.
***** */

struct TemporalDomainT
{
    boolean           fComputeDefault;
    TimeT             fStartTime;
    TimeT             fEndTime;
}; // TemporalDomainT
}; // project

}; }; };

#endif
```

3.4.4 release.idl

```
#ifndef __release__
#define __release__
//-----  

//      NAME:          release.idl
//-----
```

```

//-----
#include "files.idl"
#include "material.idl"
#include "project.idl"

//-----
//      MODULE:          mil.dtra.hpac.server.release      -
//      NOTE:           -
//                  We don't build with -pkgPrefix, because this file is      -
//                  included in other IDLs.                                -
//-----
module mil { module dtra { module hpac { module server {
/****** */
* Definitions of constants and structures from the HPACTool API related
* to releases.
* Model servers need these classes, so we put them in
* <tt>hpacshared.jar</tt>.
******/
module release
{
typedef mil::dtra::hpac::server::files::FileReferenceT
        FileReferenceT;

typedef mil::dtra::hpac::material::server::MaterialT
        MaterialT;

typedef mil::dtra::hpac::server::project::TimeT
        TimeT;

//-----
//      General Constants
//-----
        /**
         * Release distribution constant value
         */
const long      HD_LOGNORM = -1;

//-----
//      Release Status Value Constants
//-----
        /**
         * Invalid release status value
         */
const long      RS_INVALID = 0;
        /**
         * Valid release status bit mask
         */
const long      RS_VALID = 0x01 << 0;
        /**
         * Not customizable release status bit mask

```

```

        */
const long      RS_NOT_CUSTOMIZABLE = 0x01 << 1;
                /**
                 * Customized release status bit mask
                 */
const long      RS_CUSTOM = 0x01 << 2;
                /**
                 * Empty release status bit mask
                 */
const long      RS_EMPTY = 0x01 << 3;

                /**
                 * Number of status array values common to
                 * all release types
                 */
const long      RSI_STATUS_COUNT = 9;

// Continuous Release
//


//-----
//    Continuous Release Status Index Constants
//-----
                /**
                 * <em>buoyancy</em> status array index
                 * for a Continuous release
                 */
const long      RSI_CONT_BUOYANCY = RSI_STATUS_COUNT + 0;
                /**
                 * <em>distribution</em> status array index
                 * for a Continuous release
                 */
const long      RSI_CONT_DISTRIBUTION = RSI_STATUS_COUNT + 1;
                /**
                 * <em>dryMassFraction</em> status array index
                 * for a Continuous release
                 */
const long      RSI_CONT_DRY_MASS_FRACTION = RSI_STATUS_COUNT + 2;
                /**
                 * <em>duration</em> status array index
                 * for a Continuous release
                 */
const long      RSI_CONT_DURATION = RSI_STATUS_COUNT + 3;
                /**
                 * <em>MMD</em> status array index
                 * for a Continuous release
                 */
const long      RSI_CONT_MMD = RSI_STATUS_COUNT + 4;
                /**
                 * <em>massRate</em> status array index
                 * for a Continuous release
                 */

```

```

const long          RSI_CONT_MASS_RATE = RSI_STATUS_COUNT + 5;
/*  

 * <em>massSigma</em> status array index  

 * for a Continuous release  

 */
const long          RSI_CONT_MASS_SIGMA = RSI_STATUS_COUNT + 6;
/*  

 * <em>momentum</em> status array index  

 * for a Continuous release  

 */
const long          RSI_CONT_MOMENTUM = RSI_STATUS_COUNT + 7;
/*  

 * <em>sigmaY</em> status array index  

 * for a Continuous release  

 */
const long          RSI_CONT_SIGY = RSI_STATUS_COUNT + 8;
/*  

 * <em>sigmaZ</em> status array index  

 * for a Continuous release  

 */
const long          RSI_CONT_SIGZ = RSI_STATUS_COUNT + 9;
/*  

 * Total number of status array values  

 * for a Continuous release  

 */
const long          RSI_CONT_STATUS_COUNT = RSI_STATUS_COUNT + 10;

//-----  

//      TYPE:          ContinuousReleaseT  

//*****  

* Encapsulation of properties specific to continuous releases  

*****/  

struct ContinuousReleaseT
{
    /*  

     * Buoyance value in C-m3/sec  

     */
    float      fBuoyancy;

    /*  

     * Distribution of release mass across the  

     * available material particle or droplet  

     * bin sizes. A value of 0 means all mass  

     * is released as the vapor phase and is valid  

     * for liquid materials only. A value of  

     * <tt>HD_LOGNORM</tt> selects a lognormal  

     * distribution across all bin s defined  

     * by the massMeanDiameter and massSigma.  

     * A value > 0 selects the bin with that  

     * index.  

     */
}

```

```

long fDistribution;

    /**
     * Mass fraction of dry agent in wet particle
     * (slurry) releases.
    */
float fDryMassFraction;

    /**
     * Length of time for the release in hrs
    */
float fDuration;

    /**
     * Mass mean diameter of the particle or
     * liquid (in m), used if the distribution is
     * <tt>HD_LOGNORM</tt>
    */
float fMassMeanDiameter;

    /**
     * Mass release rate in mass units/sec, where
     * the mass units are taken from the
     * material
    */
float fMassRate;

    /**
     * Standard deviation of the particle or
     * liquid droplet distribution, used
     * if the distribution is <tt>HD_LOGNORM</tt>
    */
float fMassSigma;

    /**
     * Momentum of the affluent in m4/s2. Applies
     * only if the release is of a gas material
     * or the vapor phase of a liquid material,
     * and used only if the project flags include
     * <tt>HF_DYNAMIC</tt>
    */
float fMomentum;

    /**
     * Lateral Gaussian spread of the release
     * source in m
    */
float fSigmaY;

    /**
     * Vertical Gaussian spread of the release
     * source in m
    */

```

```

float           fSigmaZ;
}; // ContinuousReleaseT

// File Release
//

//-----  

//      File Release Status Index Constants
//-----  

/*
 * <em>randomCount</em> status array index
 * for a File release
 */
const long      RSI_FILE_RANDOM_COUNT = RSI_STATUS_COUNT + 0;
/***
 * <em>randomSeed</em> status array index
 * for a File release
 ***/
const long      RSI_FILE_RANDOM_SEED = RSI_STATUS_COUNT + 1;
/***
 * <em>randomSpread</em> status array index
 * for a File release
 ***/
const long      RSI_FILE_RANDOM_SPREAD = RSI_STATUS_COUNT + 2;

/**
 * Total number of status array values
 * for a File release
 */
const long      RSI_FILE_STATUS_COUNT = RSI_STATUS_COUNT + 3;

//-----  

//      TYPE:          FileReleaseT
//*****  

* Encapsulation of properties specific to file releases
*****/  

struct FileReleaseT
{
    long           fRandomCount;
    long           fRandomSeed;
    float          fRandomSpread;
    FileReferenceT fFile;
}; // FileReleaseT

// Instantaneous Release
//

//-----  

//      Instantaneous Release Status Index Constants
//-----  


```

```

    /**
     * <em>buoyancy</em> status array index
     * for an Instantaneous release
     */
const long RSI_INST_BUOYANCY = RSI_STATUS_COUNT + 0;
    /**
     * <em>distribution</em> status array index
     * for an Instantaneous release
     */
const long RSI_INST_DISTRIBUTION = RSI_STATUS_COUNT + 1;
    /**
     * <em>dryMassFraction</em> status array index
     * for an Instantaneous release
     */
const long RSI_INST_DRY_MASS_FRACTION = RSI_STATUS_COUNT + 2;
    /**
     * <em>mass</em> status array index
     * for an Instantaneous release
     */
const long RSI_INST_MASS = RSI_STATUS_COUNT + 3;
    /**
     * <em>MMD</em> status array index
     * for an Instantaneous release
     */
const long RSI_INST_MMD = RSI_STATUS_COUNT + 4;
    /**
     * <em>massSigma</em> status array index
     * for an Instantaneous release
     */
const long RSI_INST_MASS_SIGMA = RSI_STATUS_COUNT + 5;
    /**
     * <em>momentum</em> status array index
     * for an Instantaneous release
     */
const long RSI_INST_MOMENTUM = RSI_STATUS_COUNT + 6;
    /**
     * <em>randomCount</em> status array index
     * for an Instantaneous release
     */
const long RSI_INST_RANDOM_COUNT = RSI_STATUS_COUNT + 7;
    /**
     * <em>randomSeed</em> status array index
     * for an Instantaneous release
     */
const long RSI_INST_RANDOM_SEED = RSI_STATUS_COUNT + 8;
    /**
     * <em>randomSpread</em> status array index
     * for an Instantaneous release
     */
const long RSI_INST_RANDOM_SPREAD = RSI_STATUS_COUNT + 9;
    /**
     * <em>sigmaX</em> status array index
     * for an Instantaneous release
     */

```

```

        */
const long      RSI_INST_SIGX = RSI_STATUS_COUNT + 10;
                /**
                 * <em>sigmaY</em> status array index
                 * for an Instantaneous release
                 */
const long      RSI_INST_SIGY = RSI_STATUS_COUNT + 11;
                /**
                 * <em>sigmaZ</em> status array index
                 * for an Instantaneous release
                 */
const long      RSI_INST_SIGZ = RSI_STATUS_COUNT + 12;

                /**
                 * Total number of status array values
                 * for an Instantaneous release
                 */
const long      RSI_INST_STATUS_COUNT = RSI_STATUS_COUNT + 13;

//-----
//      TYPE:           InstantaneousReleaseT
//***** Encapsulation of properties specific to instantaneous releases
***** struct InstantaneousReleaseT
{
    /**
     * @see ContinuousReleaseT
     */
    float         fBuoyancy;

    /**
     * @see ContinuousReleaseT
     */
    long          fDistribution;

    /**
     * Mass fraction of dry agent in wet particle
     * (slurry) releases.
     */
    float         fDryMassFraction;

    /**
     * Total mass in mass units as defined in
     * the material
     */
    float         fMass;

    /**
     * @see ContinuousReleaseT
     */
    float         fMassMeanDiameter;

```

```

        /**
         * @see ContinuousReleaseT
         */
float fMassSigma;

        /**
         * @see ContinuousReleaseT
         */
float fMomentum;

        /**
         * Number of release locations to be
         * randomly generated
         */
long fRandomCount;

        /**
         * Random generator seed (should be a large,
         * positive, odd integer)
         */
long fRandomSeed;

        /**
         * Radius of the circular region (in m)
         * containing the randomly distributed
         * sources
         */
float fRandomSpread;

        /**
         * Lateral Gaussian spread (in m) of the release
         * source in the horizontal X direction
         */
float fSigmaX;

        /**
         * Lateral Gaussian spread (in m) of the release
         * source in the horizontal Y direction
         */
float fSigmaY;

        /**
         * Vertical Gaussian spread of the release
         * source in m
         */
float fSigmaZ;
}; // InstantaneousReleaseT

// Moving Release
//

```

```

//-----
//      Moving Release Status Index Constants
//-----
/*
 * <em>buoyancy</em> status array index
 * for a Moving release
 */
const long RSI_MOVING_BUOYANCY = RSI_STATUS_COUNT + 0;
/*
 * <em>distribution</em> status array index
 * for a Moving release
 */
const long RSI_MOVING_DISTRIBUTION = RSI_STATUS_COUNT + 1;
/*
 * <em>dryMassFraction</em> status array index
 * for a Moving release
 */
const long RSI_MOVING_DRY_MASS_FRACTION = RSI_STATUS_COUNT + 2;
/*
 * <em>duration</em> status array index
 * for a Moving release
 */
const long RSI_MOVING_DURATION = RSI_STATUS_COUNT + 3;
/*
 * <em>MMD</em> status array index
 * for a Moving release
 */
const long RSI_MOVING_MMD = RSI_STATUS_COUNT + 4;
/*
 * <em>massRate</em> status array index
 * for a Moving release
 */
const long RSI_MOVING_MASS_RATE = RSI_STATUS_COUNT + 5;
/*
 * <em>massSigma</em> status array index
 * for a Moving release
 */
const long RSI_MOVING_MASS_SIGMA = RSI_STATUS_COUNT + 6;
/*
 * <em>momentum</em> status array index
 * for a Moving release
 */
const long RSI_MOVING_MOMENTUM = RSI_STATUS_COUNT + 7;
/*
 * <em>sigmaY</em> status array index
 * for a Moving release
 */
const long RSI_MOVING_SIGY = RSI_STATUS_COUNT + 8;
/*
 * <em>sigmaz</em> status array index
 * for a Moving release
 */
const long RSI_MOVING_SIGZ = RSI_STATUS_COUNT + 9;

```

```

        /**
         * <em>velocityX</em> status array index
         * for a Moving release
         */
const long RSI_MOVING_VELX = RSI_STATUS_COUNT + 10;
        /**
         * <em>velocityY</em> status array index
         * for a Moving release
         */
const long RSI_MOVING_VELY = RSI_STATUS_COUNT + 11;
        /**
         * <em>velocityZ</em> status array index
         * for a Moving release
         */
const long RSI_MOVING_VELZ = RSI_STATUS_COUNT + 12;

        /**
         * Total number of status array values
         * for a Moving release
         */
const long RSI_MOVING_STATUS_COUNT = RSI_STATUS_COUNT + 13;

//-----
//      TYPE:          MovingReleaseT
// ****
* Encapsulation of properties specific to moving releases
****

struct MovingReleaseT
{
    /**
     * @see ContinuousReleaseT
     */
    float fBuoyancy;

    /**
     * @see ContinuousReleaseT
     */
    long fDistribution;

    /**
     * Mass fraction of dry agent in wet particle
     * (slurry) releases.
     */
    float fDryMassFraction;

    /**
     * @see ContinuousReleaseT
     */
    float fDuration;

    /**
     * @see ContinuousReleaseT
     */

```

```

        */
float fMassMeanDiameter;

        /**
 * @see ContinuousReleaseT
 */
float fMassRate;

        /**
 * @see ContinuousReleaseT
 */
float fMassSigma;

        /**
 * @see ContinuousReleaseT
 */
float fMomentum;

        /**
 * @see ContinuousReleaseT
 */
float fSigmaY;

        /**
 * @see ContinuousReleaseT
 */
float fSigmaZ;

        /**
 * Source velocity component in the X
 * direction (m/s)
 */
float fVelocityX;

        /**
 * Source velocity component in the Y
 * direction (m/s)
 */
float fVelocityY;

        /**
 * Source velocity component in the Z
 * direction (m/s)
 */
float fVelocityZ;
};

// MovingReleaseT

// Pool Release
//



-----  

//      Pool Release Status Index Constants
-
```

```

//-----
/* 
 * <em>mass</em> status array index
 * for a Pool release
 */
const long RSI_POOL_MASS = RSI_STATUS_COUNT + 0;
/** 
 * <em>sizeX</em> status array index
 * for a Pool release
 */
const long RSI_POOL_SIZEX = RSI_STATUS_COUNT + 1;
/** 
 * <em>sizey</em> status array index
 * for a Pool release
 */
const long RSI_POOL_SIZEY = RSI_STATUS_COUNT + 2;

/** 
 * Total number of status array values
 * for a Pool release
 */
const long RSI_POOL_STATUS_COUNT = RSI_STATUS_COUNT + 3;

//-----
//      TYPE:          PoolReleaseT
//*****
* Encapsulation of properties specific to pool releases
*****/
struct PoolReleaseT
{
    /**
     * Total mass in mass units as defined in
     * the material
     */
    float fMass;

    /**
     * Radius of the liquid pool in the horizontal
     * X direction (m)
     */
    float fSizeX;

    /**
     * Radius of the liquid pool in the horizontal
     * Y direction (m)
     */
    float fSizeY;
}; // PoolReleaseT

// Stack Release
//

```

```

//-----
//      Stack Release Status Index Constants
//-----
/*
 * <em>diameter</em> status array index
 * for a Stack release
 */
const long          RSI_STACK_DIAMETER = RSI_STATUS_COUNT + 0;
/*
 * <em>distribution</em> status array index
 * for a Stack release
 */
const long          RSI_STACK_DISTRIBUTION = RSI_STATUS_COUNT + 1;
/*
 * <em>duration</em> status array index
 * for a Stack release
 */
const long          RSI_STACK_DURATION = RSI_STATUS_COUNT + 2;
/*
 * <em>exitTemperature</em> status array index
 * for a Stack release
 */
const long          RSI_STACK_EXITTEMP = RSI_STATUS_COUNT + 3;
/*
 * <em>exitVelocity</em> status array index
 * for a Stack release
 */
const long          RSI_STACK_EXITVEL = RSI_STATUS_COUNT + 4;
/*
 * <em>MMD</em> status array index
 * for a Stack release
 */
const long          RSI_STACK_MMD = RSI_STATUS_COUNT + 5;
/*
 * <em>massRate</em> status array index
 * for a Stack release
 */
const long          RSI_STACK_MASS_RATE = RSI_STATUS_COUNT + 6;
/*
 * <em>massSigma</em> status array index
 * for a Stack release
 */
const long          RSI_STACK_MASS_SIGMA = RSI_STATUS_COUNT + 7;

/*
 * Total number of status array values
 * for a Stack release
 */
const long          RSI_STACK_STATUS_COUNT = RSI_STATUS_COUNT + 8;

//-----

```

```

//      TYPE:          StackReleaseT
//*********************************************************************
* Encapsulation of properties specific to stack releases
//********************************************************************/
struct StackReleaseT
{
    /**
     * Diameter of the stack in m
     */
    float fDiameter;

    /**
     * @see ContinuousReleaseT
     */
    long fDistribution;

    /**
     * @see ContinuousReleaseT
     */
    float fDuration;

    /**
     * Excess exit temperature of the effluent
     * in C. Applies only if the release is of a
     * gas material or the vapor phase of a liquid
     * material. Used only if the project flags
     * include <tt>HF_DYNAMIC</tt>.
     */
    float fExitTemp;

    /**
     * Exit velocity of the effluent
     * in m/s. Applies only if the release is of a
     * gas material or the vapor phase of a liquid
     * material. Used only if the project flags
     * include <tt>HF_DYNAMIC</tt>.
     */
    float fExitVelocity;

    /**
     * @see ContinuousReleaseT
     */
    float fMassMeanDiameter;

    /**
     * @see ContinuousReleaseT
     */
    float fMassRate;

    /**
     * @see ContinuousReleaseT
     */
    float fMassSigma;

```

```

}; // StackReleaseT

        // General Release
        //

//-----  

//      Release Status Index Constants  

//-----  

        /**
         * Maximum size of a status array
         * (not used)
         */
const long      RELEASE_STATUS_LENGTH = 32;

        /**
         * <em>release</em> status array index
         * (status of the entire release)
         */
const long      RSI_RELEASE_STATUS = 0;
        /**
         * <em>horzSize</em> status array index
         * for any release
         */
const long      RSI_HORZ_SIZE = 1;
        /**
         * <em>horzUncertainty</em> status array index
         * for any release
         */
const long      RSI_HORZ_UNCERTAINTY = 2;
        /**
         * <em>location</em> status array index
         * for any release
         */
const long      RSI_LOCATION = 3;
        /**
         * <em>material</em> status array index
         * for any release
         */
const long      RSI_MATERIAL = 4;
        /**
         * <em>puffDuration</em> status array index
         * for any release
         */
const long      RSI_PUFF_DURATION = 5;
        /**
         * <em>startTime</em> status array index
         * for any release
         */
const long      RSI_START_TIME = 6;
        /**
         * <em>vertSize</em> status array index
         * for any release
         */

```

```

        */
const long      RSI_VERT_SIZE = 7;
                /**
                 * <em>vertUncertainty</em> status array index
                 * for any release
                */
const long      RSI_VERT_UNCERTAINTY = 8;

//-----
//    Release Type Constants
//-----
/* 
 * Type value for an Instantaneous release
 */
const long      HR_INSTANTANEOUS = (0x01 << 0);
                /**
                 * Type value for a Continuous release
                */
const long      HR_CONTINUOUS = (0x01 << 1);
                /**
                 * Type value for a File release
                */
const long      HR_FILE = (0x01 << 20) + HR_INSTANTANEOUS;
                /**
                 * Type value for a Moving release
                */
const long      HR_MOVING = (0x01 << 21) + HR_CONTINUOUS;
                /**
                 * Type value for a Pool release
                */
const long      HR_POOL = (0x01 << 22) + HR_CONTINUOUS;
                /**
                 * Type value for a Stack release
                */
const long      HR_STACK = (0x01 << 23) + HR_CONTINUOUS;

//-----
//    TYPE:          ReleaseDataT
//*****
* Defined union for all release subtypes.
*****/
union ReleaseDataT  switch( long )
{
    case HR_CONTINUOUS:
        ContinuousReleaseT      fContinuous;

    case HR_FILE:
        FileReleaseT            fFile;

    case HR_INSTANTANEOUS:
        InstantaneousReleaseT   fInstantaneous;
}

```

```

case HR_MOVING:
    MovingReleaseT           fMoving;

case HR_POOL:
    PoolReleaseT            fPool;

case HR_STACK:
    StackReleaseT           fStack;
} // ReleaseDataT

//-----
//      TYPE:          ReleaseT
//      NOTES:
//          The type is the ReleaseDataT discriminator.
/*****************************************/
* HPAC Tool or Server view of a release. Note that the type of the
* release is indicated by the <em>fReleaseData</em> property.
/****************************************/
struct ReleaseT
{
    /**
     * Bit mask composed of the defined
     * <tt>RE_</tt> constants
     */
    //long        fAvailableEffects;

    /**
     * Spatial reach of this release on the
     * earth's surface in km
     */
    float       fHorzSize;

    /**
     * Uncertainty in m
     */
    float       fHorzUncertainty;

    /**
     * Unique release ID.
     * <tt>ReleaseUtils.createID()</tt> will create
     * a unique ID.
     */
    string      fID;

    /**
     * ID of the notional incident to which this
     * release belongs
     */
    string      fIncidentID;

    /**

```

```

        * At this level, only lat-lon-alt values
        * are stored. The coordinate system chosen
        * by the user is represented in the
        * location object property of the
        * corresponding <tt>Release</tt> object
        */
float fLLALocation[ 3 ];

        /**
        * Co-located releases should be tagged with
        * the same value for this property. Releases
        * which need not be co-located must have
        * different values for this property. Values
        * should start with 0 and increment by 1,
        * but this is not required.
        */
long fLocationGroup;

        /**
        * The material object for this release
        */
MaterialT fMaterial;

        /**
        * Flag indicating whether this is a standard
        * or customized material
        */
boolean fMaterialCustomized;

        /**
        * Estimated duration of the puffs related
        * to this release, in hrs
        */
float fPuffDuration;

        /**
        * Union for descriptions of the different
        * kinds of releases
        */
ReleaseDataT fReleaseData;

        /**
        * This is the absolute start time for this
        * release in UTC. The incident time will
        * also be stored in UTC.
        */
TimeT fStartTime;

        /**
        * Status flags for various fields, including
        * those defined for this structure (common
        * to all release types) and the ones
        * defined for the type of release data

```

```

        */
sequence<long>      fStatus;

        /**
         * Vertical spatial reach of this release in m
         */
float          fVertSize;

        /**
         * Vertical uncertainty in m
         */
float          fVertUncertainty;
}; // ReleaseT

//-----
//      TYPE:           ReleaseListT
//***** Definition of a sequence or array of <tt>ReleaseT</tt> objects.
***** /*****
typedef sequence<ReleaseT>
                    ReleaseListT;
}; // release

}; }; }; };

#endif

```

3.4.5 server.idl

```

#ifndef __server__
#define __server__
//-----
//      NAME:           server.idl
//-----
#include "release.idl"

//-----
//      MODULE:          mil.dtra.hpac.server
//      NOTE:
//                  We don't build with -pkgPrefix, because this file is
//                  included in model IDLs.
//-----
module mil { module dtra { module hpac {
//***** Definition of constants and structures related to HPAC incident model
* Definitions of constants and structures related to HPAC incident model
* servers. Packaged in <tt>hpacshared.jar</tt>.
***** /*****
module server
{
    typedef mil::dtra::hpac::server::project::TimeT

```

```

TimeT;

typedef mil::dtra::hpac::server::release::ReleaseT
    ReleaseT;

typedef mil::dtra::hpac::server::release::ReleaseListT
    ReleaseListT;

//-----
//      Special Value Constants
//-----
/***
 * Constant for a default real value
 */
const float      DEFAULT_FLOAT = 1.0e36;

/***
 * Constant for a deferred real value;
 * deferred values are dependent upon met
 */
const float      DEFERRED_FLOAT = 2.0e36;

/***
 * Constant for an empty real value
 */
const float      EMPTY_FLOAT = -1.0 * DEFAULT_FLOAT;

/***
 * Constant for an empty real value,
 * equivalent to <tt>EMPTY_FLOAT</tt>
 */
const float      NOT_SET = -1.0 * DEFAULT_FLOAT;
/***
 * Constant value indicating a property
 * is not supported
 */
const long       NOT_SUPPORTED = -1;

/***
 * Constant for a default integer value
 */
const long       DEFAULT_LONG = 65535;
/***
 * Constant for a deferred integer value;
 * deferred values are dependent upon met
 */
const long       DEFERRED_LONG = DEFAULT_LONG - 1;
/***
 * Constant for an empty integer value
 */
const long       EMPTY_LONG = -DEFAULT_LONG;
/***

```

```

        * Constant for an empty string
        * (not used)
        */
const string          EMPTY_STRING = "<empty>";

//-----
//      EXCEPTION:      ModelException
//*****
* Definition of an exception in model server side processing.
*****exception ModelException
{
    /**
     * name of the model in which the
     * exception occurred
     */
string          fModelName;

    /**
     * exception message
     */
string          fMessage;
};

//-----
//      TYPE:          ByteArray
//*****
* Array of bytes.
*****typedef sequence<octet>
                    ByteArray;

//-----
//      TYPE:          IncidentLocation
//*****
*****typedef float
                    IncidentLocation[ 3 ];

//-----
//      TYPE:          PointEnvironmentT
//*****
* Definition derived from the HPACTool API.
*****struct PointEnvironmentT
{
    float          fHumidity;
    float          fPotentialTemp;
    float          fPressure;
    float          fWindUComponent; // x
}

```

```

float          fWindVComponent; // y
float          fWindWComponent; // z
float          fZ;
}; // PointEnvironmentT

//-----
//      TYPE:           EnvironmentT
//*****
* Definition derived from the HPACTool API.
*****/
struct EnvironmentT
{
    float          fMixingLayerHeight;
    sequence<PointEnvironmentT> fSamples;
    short          fSampleCount;
    float          fSurfaceElevation;
    float          fSurfacePressure;
}; // EnvironmentT

//-----
//      TYPE:           EnviroBLT
//*****
* Definition derived from the HPACTool API.
*****/
struct EnviroBLT
{
    float          fSurfaceRoughness;
    float          fCanopyHeight;
    float          fCanopyFlowIndex;
    float          fMoninObukovLength;
    float          fSurfaceLayerHeight;
    float          fBLMixingHeight;
    float          fSurfaceHeatFlux;
    float          fHorizVelocityScale;
    float          fVertVelocityScale;
    float          fTemperatureGradient;
}; // EnviroBLT

//-----
//      Mode constants for updateIncident
//-----
/***
 * Request mask value for
 * <tt>IncidentModelServer.updateIncident()</tt>
 * indicating the last call before
 * calculation begins
 */
const long      HU_LASTCHANCE = 0x01 << 20;

```

```

//-----
// Mode constants for updateRelease
//-----
/*
 * Mode value for
 * <tt>IncidentModelServer.updateRelease()</tt>
 * indicating updated environment data is
 * available
 */
const long HU_ENVIRONMENT = 0x01 << 21;

//-----
// TYPE: TimeMode
//*****
* Time mode enumeration as defined in the HPACTool API.
*****/enum TimeMode
{
    HT_UTC,
    HT_LOCAL
}; // TimeMode

//-----
// TYPE: IncidentT
//*****
* Encapsulates the incident abstraction as seen by ScipuffServer.
*****/struct IncidentT
{
    /**
     * Absolute time reference for the beginning
     * of the incident
     */
    TimeT fAbsoluteTime;

    /**
     * Bit mask composed of the defined
     * <tt>EM_</tt> constants in effects.idl
     */
    long fAvailableEffects;

    /**
     * True if the incident has releases with
     * custom materials
     * (does anyone use this?)
     */
    boolean fHasCustomMaterials;

    /**
     * True if the incident has customized

```

```

        * releases
        * (does anyone use this?)
        */
boolean fHasCustomReleases;

        /**
         * Assigned, unique identifier for the
         * incident object
         */
string fID;

        /**
         * Logical location of the incident
         * in LLA coordinates
         */
float fLLALocation[ 3 ];

        /**
         * Name of the source model that generated
         * the incident
         */
string fmodelName;

        /**
         * Name in the naming or lookup service of
         * the factory server object which can
         * produce an <tt>IncidentModelServer</tt>
         * instance for processing this object
         */
string fModelServiceName;

        /**
         * Name assigned to this incident by the
         * user
         */
string fName;

        /**
         * List of release objects comprising the
         * incident
         */
ReleaseListT fReleaseList;
}; // IncidentT

//-----
// CONSTANT:      NO_SERVICE_NAME
*****-
* Constant for specifying no service for a model which has no
* incident model server and, therefore, does not update releases.
*****/
const string NO_SERVICE_NAME = "no service";

```

```

//-----
// CONSTANT:      VERBOSE_PROP_NAME
//****************************************************************************
* Name of boolean system property indicating verbose/debug log output.
//****************************************************************************
const string      VERBOSE_PROP_NAME = "scipuffserver.verbose";

//-----
// INTERFACE:      IncidentModelServer
//****************************************************************************
* Defines the common interface for all incident source model server
* objects.
*
* <p>We adopt a stateless, connectionless communication paradigm from
* client to server. Each time a client needs a model server to process
* an incident, the client must obtain a per-client instance of an
* <tt>IncidentModelServer</tt> from the corresponding
* <tt>IncidentModelServerFactory</tt>, which is a singleton object
* registered under a well known name in the naming service.
*
* @see IncidentModelServerFactory
//****************************************************************************
interface IncidentModelServer
{
//-----
// METHOD:         closeIncident()
//****************************************************************************
* Gives the incident model server an opportunity to clean up any resources
* created.
* <em>[Will be called at the end of a run, when the incident model
* server will be no longer needed?]</em>
*
* @param incident the incident to close
//****************************************************************************
void
closeIncident( in IncidentT incident )
    raises( ModelException );
}

//-----
// METHOD:         computeEffect()
//****************************************************************************
* Computes effects associated with an incident.
* <p><table>
*   <tr>
*     <th>effect_id</th> <th>effect_in</th> <th>effect_out</th>
*   </tr>
*   <tr align="center">
*     <td><tt>REID_BLAST_CIRCLE</tt></td>
*     <td> N/A </td>
*     <td> <tt>EffectsCircleT</tt> structure (currently undefined) </td>

```

```

*      </tr>
*      <tr align="center">
*          <td><tt>REID_THERMAL_CIRCLE</tt></td>
*          <td> N/A </td>
*          <td> <tt>EffectsCircleT</tt> structure (currently undefined) </td>
*      </tr>
*      <tr align="center">
*          <td><tt>REID_PROMPT_CIRCLE</tt></td>
*          <td> N/A </td>
*          <td> <tt>EffectsCircleT</tt> structure (currently undefined) </td>
*      </tr>
*      <tr bgcolor="lightyellow"> <td colspan=4> &nbsp; </td> </tr>
*      <tr align="center">
*          <td><tt>REID_INIT_PROMPCAS</tt></td>
*          <td> <tt>NWPNIInputPCEInitT</tt> structure </td>
*          <td> <tt>NWPNOOutputPCEInitT</tt> structure </td>
*      </tr>
*      <tr align="center">
*          <td><tt>REID_PROMPT_CASUALTY</tt></td>
*          <td> <tt>NWPNIInputPCEComputeT</tt> structure </td>
*          <td> <tt>NWPNOOutputPCEComputeT</tt> structure </td>
*      </tr>
*      <tr align="center">
*          <td><tt>REID_PROMPT_CASUALTY</tt></td>
*          <td> N/A </td>
*          <td> N/A </td>
*      </tr>
*      </table>
*
* @param request identifies the request as defined in effects.idl,
*                  one of <tt>EID_INIT</tt>, <tt>EID_COMP</tt>, or
*                  (<tt>EID_EXIT</tt>
* @param effect_id identifies the effect to be processed, as per
*                  effect ID constants defined in effects.idl, one of
*                  <tt>EID_CIRCLE_BLAST</tt>, <tt>EID_CIRCLE_THERM</tt>,
*                  <tt>EID_CIRCLE_PROMPT</tt>, or
*                  <tt>EID_CASUALTY_PROMPT</tt>
* @param effect_in effect-dependent input data
* @param effect_out effect-dependent output data
* @param incident_time incident time (<tt>IncidentT.fAbsoluteTime</tt>)
* @param incident_location incident location
*                                (<tt>IncidentT.fLLALocation</tt>)
* @param model_incident model-specific data updated on return
* @return non-zero indicating a problem computing the effect,
*                  <tt>NOT_SUPPORTED.value</tt> if the computation
*                  is not supported by this model, or zero if
*                  the function succeeded
***** */
long
computeEffect(
    in long           request,
    in long           effect_id,
    in any            effect_in,

```

```

        out any          effect_out,
        in TimeT         incident_time,
        in IncidentLocation incident_location,
        //in IncidentT    incident,
        inout any         model_incident
    )
    raises ( ModelException );

//-----
//  METHOD:           initIncident()          -
//*****
* Initializes the model incident data, the incident object and its
* corresponding release list. This method is part of the services
* provided by the IncidentModelServer to <b>any</b> client needing
* initialized <tt>Incident</tt> (and thus <tt>Release</tt>s)
* and <tt>ModelIncident</tt> objects. This avoids requiring clients
* to be knowledgeable of how to initialize objects for a particular
* incident model.
*
* <p>
* HPAC 4 model beans make this call in their
* <tt>createIncidentObjects()</tt> methods. The incident is initialized
* with a unique ID and a default name and anything else that may be
* documented with the model server implementation.
* </p>
*
* @param incident  default incident object returned
* @param model_incident  default model-specific data
*****/
void
initIncident(
    inout IncidentT incident,
    inout any         model_incident
)
    raises ( ModelException );

//-----
//  METHOD:           restoreCustomizedProperties()      -
//*****
* Restores customized properties as saved in the <em>from</em> parameter.
*
* @param to          release object to update
* @param from        release object from which to read customized props
*****/
void
restoreCustomizedProperties(
    inout ReleaseT to,
    in ReleaseT   from
);

```

```

//-----
// METHOD:          terminate()           -
/* Hook for the server implementation to do things before shutting
* down a per-client instance. Clients must call this method before
* releasing the remote server object.
*****/
void
terminate();

//-----
// METHOD:          updateIncident()        -
/* Based on the properties of the incident and model incident objects,
* updates and/or recomputes the incident object and its release list.
*
* <p>Called by the model bean when the user <em>Apply</em> or
* <em>OK</em>s changes to the operational model parameters.
*
* <p>Called by ScipuffServer with the <tt>HU_LASTCHANCE</tt>
* bit set in request_mask right before a call to the Tool Engine/SCIPUFF.
*
* @param request_mask mask of flags indicating the type of request;
*                      currently the only flag is <tt>HU_LASTCHANCE</tt>
* @param incident     incident object updated on return
* @param model_incident model-specific data updated on return
*****/
void
updateIncident(
    in long          request_mask,
    inout IncidentT incident,
    inout any         model_incident
)
    raises ( ModelException );

//-----
// METHOD:          updateRelease()         -
/* This is the model server's opportunity to update release parameters
* based on environment data computed by the Tool Engine.
*
* <p>This method is implemented in <tt>IncidentModelServerImpl</tt>
* and should not normally be overridden by incident model servers.
* Instead, this method calls <tt>updateReleaseData()</tt> while
* preserving user customizations of release properties. Subclasses
* should override <tt>updateReleaseData()</tt>.
*
* @param mode       mode for the request;
*                      currently the only defined modes are 0 (meaning none),
*                      <tt>HU_ENVIRONMENT</tt>, and
*                      <tt>HF_FAST</tt>

```

```

* @param current_time current time of the update request in hours from
*                     the project start time
* @param next_update time of the next anticipated update request in hours
*                     from the project start time
* @param environ computed environment data to use for update
* @param incident incident associated with the release
* @param model_incident model-specific data updated on return
* @param release release object to update on return
*****/
void
updateRelease(
    in long          mode,
    in float         current_time,
    in float         next_update,
    in EnvironmentT environ,
    in IncidentT    incident,
    inout any        model_incident,
    inout ReleaseT   release
)
raises ( ModelException );

//-----
// METHOD:           updateReleaseData() -
//*****
* Called by <tt>updateRelease()</tt> to perform model-specific updates
* to release properties based on the environment.
*
* <p>Model servers must implement this method.
*
* @param mode      mode for the request;
*                  currently the only defined modes are 0 (meaning none),
*                  <tt>HU_ENVIRONMENT</tt>, and
*                  <tt>HF_FAST</tt>
* @param current_time current time of the update request in hours from
*                     the project start time
* @param next_update time of the next anticipated update request in hours
*                     from the project start time
* @param environ   computed environment data to use for update
* @param incident  incident associated with the release
* @param model_incident model-specific data updated on return
* @param release   release object to update on return
*****/
void
updateReleaseData(
    in long          mode,
    in float         current_time,
    in float         next_update,
    in EnvironmentT environ,
    in IncidentT    incident,
    inout any        model_incident,
    inout ReleaseT   release
)

```

```

        raises ( ModelException );
    } // IncidentModelServer

//-----
// INTERFACE: IncidentModelServerFactory -
//*****
* Defines the common interface for all incident source model server
* factory objects.
*
* <p>We adopt a stateless, connectionless communication paradigm from
* client to server. Each time a client needs a model server to process
* an incident, the client must obtain a per-client instance of an
* <tt>IncidentModelServer</tt> from the corresponding
* <tt>IncidentModelServerFactory</tt>, which is a singleton object
* registered under a well known name in the naming service.
*
* @see IncidentModelServer
*****/
interface IncidentModelServerFactory
{
//-----
// METHOD: getInstance() -
//*****
* Instantiates and returns an instance of the model server.
*
* @param user_name user name
* @param project_name project name
* @param incident_id incident identifier
* @return model server instance
*****/
IncidentModelServer
getInstance(
    in string      user_name,
    in string      project_name,
    in string      incident_id
)
    raises ( ModelException );
}; // IncidentModelServerFactory
}; // server
};

};

#endif

```

3.4.6 weatherT.idl

```

#ifndef __weatherT__
#define __weatherT__
//-----
// NAME: weatherT.idl -
//-----

```

```
#include "files.idl"

//-----
//      MODULE:          mil.dtra.weather.data.weatherT
//-----

module mil { module dtra { module weather { module shared { module data {
module weatherT
{

typedef sequence<string> StringSeq;
typedef sequence<float>  FloatSeq;
typedef long    longArray[2];
typedef float   floatArray[2];
typedef mil::dtra::hpac::server::files::FileReferenceT FileReferenceT;

//-----
// doHazard constants
//-----

const long    HFB_HAZARD = 1;
const long    HFB_DUAL = 2;

//-----
// time reference constants
//-----

const long    HT_LOCAL = 1;
const long    HT_UTC = 0;

//-----
// Weather type constants
//-----

const long    HW_METFIX      = 1;
const long    HW_METSRF      = 0x01 << 1;
const long    HW_METPRF      = 0x01 << 2;
const long    HW_METMRF      = 0x01 << 3;
const long    HW_METMED      = 0x01 << 4;
const long    HW_METSCLI     = 0x01 << 5;
const long    HW_METPCLI     = 0x01 << 6;
const long    HW_METOPER     = 0x01 << 7;
const long    HW_METHIST     = 0x01 << 8;
const long    HW_METFCST     = 0x01 << 9;
const long    HW_METCURRE    = 0x01 << 10;

//-----
// Met output type constants
//-----

const long    HO_OUTPUT      = 1;
const long    HO_OUTMET      = 0x01 << 1;
```

```

const long    HO_ASCII      = 0x01 << 2;
const long    HO_2D         = 0x01 << 3;
const long    HO_3D         = 0x01 << 4;

//-----
//      Units
//-----

const long    HU_DEG       = 1;
const long    HU_DMS       = 2;
const long    HU_MPS       = 1;
const long    HU_KTS       = 2;
const long    HU MPH      = 3;
const long    HU_KPH       = 4;
const long    HU_FPS       = 5;

//-----
//      Boundary layer types
//-----

const long    HB_NONE      = 0;
const long    HB_BLSIMPLE  = 0x01 << 0;
const long    HB_BLCALC    = 0x01 << 1;
const long    HB_BLPRF     = 0x01 << 2;
const long    HB_BLOBS     = 0x01 << 3;
const long    HB_BLMED     = 0x01 << 4;
const long    HB_BLOPER    = 0x01 << 7;

//-----
//      Surface moisture types
//-----

const long    MST_DRY      = 1;
const long    MST_NORMAL   = 2;
const long    MST_WET      = 3;

//-----
//      LSV types
//-----

const long    HL_NONE      = 0;
const long    HL_LSVINP    = 0x01 << 0;
const long    HL_LSVMOD    = 0x01 << 1;
const long    HL_LSVOBS    = 0x01 << 3;
const long    HL_LSVOPER   = 0x01 << 7;
const long    HL_LSVOPER_20 = 0x01 << 7 + 18;

//-----
//      Precipitation types
//-----

const long    HP_NONE      = 0;
const long    HP_PRCPINP   = HP_NONE;

```

```

const long      HP_PRCPMET      = 0x01 << 0;
const long      HP_RAIN        = 0x01 << 1;
const long      HP_SNOW         = 0x01 << 2;
const long      HP_LIGHT        = 0x01 << 3;
const long      HP_MODERATE    = 0x01 << 4;
const long      HP_HEAVY        = HP_MODERATE + HP_LIGHT;

//-----
//      Mass consistency types
//-----

const long      HT_NONE         = 0;
const long      HT_SWIFT        = 0x01 << 0;
const long      HT SCIPUFF       = 0x01 << 1;
const long      HT_AVAIL_T      = 0x01 << 9;
const long      HT_USE_T        = 0x01 << 10;
const long      HT_AVAIL_L      = 0x01 << 13;
const long      HT_USE_L        = 0x01 << 14;
const long      HT_CATEGORY     = 0x01 << 15;

//-----
//      Structures used in the weather
//-----


struct MetFlagsT {
    long          fTimeReference;
    boolean       fDoMassConsistency;
    long          fDoOutput;
    long          fDoHazard;
    float         fOutputInterval;
    float         fUncertaintyScaleLength;
};

struct MetMetT {
    long          fMetMethod;
    long          fMaxProfileLocations;
    long          fMaxSurfaceLocations;
    float         fTimeBin;
    long          fNumberSurfClimoStations;
    long          fUACDMonthDay;
    long          fWindSpeedUnit;
    long          fWindDirectionUnit;
    float         fFixedWindsSpeed;
    float         fFixedWindsDirection;
    StringSeq     fSurfClimoStations;
    FileReferenceT fMetFile1;
    FileReferenceT fMetFile2;
};

struct MetBLT {
    long          fBLMethod;
    long          fSurfaceMoisture;
    float         fDayMaxInversionHeight;
};

```

```

float      fNightMaxInversionHeight;
float      fDayMaxSurfaceHeatFlux;
float      fNightMaxSurfaceHeatFlux;
float      fCanopyHeight;
float      fSurfaceRoughness;
float      fSurfaceAlbedo;
float      fSurfaceBowenRatio;
float      fCloudFraction;
float      fCanopyFlowIndex;
}; // MetBLT

struct MetLSVT {
    long      fLSVMMethod;
    float     fLSVTurb;
    float     fLSVLengthScale;
}; // MetLSVT

struct MetPrecipT {
    long      fPrecipType;
    long      fPrecipClass;
}; // MetPrecipT

struct TerrainMCT {
    float     fSwiftDeltaT;
    longArray fMaxIterations;
    floatArray fErrorCriteria;
    floatArray fVerticalAdjustment;
    long      fNumberVerticalPoints;
    FloatSeq   fVerticalGrid;
}; // TerrainMCT

struct TerrainT {
    long      fMCType;
    TerrainMCT fMassConsistency;
    FileReferenceT fTerrainFile;
}; // TerrainT

//-----
//      TYPE: WeatherT
//      should implement Serializable and PropsSerializer
//-----
struct WeatherT {
    MetFlagsT   fMetFlags;
    MetMetT     fMeteorology;
    MetBLT      fBoundaryLayer;
    MetLSVT     fLSVariability;
    MetPrecipT   fPrecipitation;
    TerrainT    fTerrain;
}; // struct WeatherT

interface WeatherAccessors {

```

```

void setMetFlagsT(
    in long      fTimeReference,
    in boolean   doMassConsistency,
    in long      doOutput,
    in long      doHazard,
    in float     outputInterval,
    in float     uncertaintyScaleLength
);

void setMetMetT (
    in long      metMethod,
    in long      maxProfileLocations,
    in long      maxSurfaceLocations,
    in float     timeBin,
    in long      numberSurfClimoStations,
    in long      UACDMonthDay,
    in long      windSpeedUnit,
    in long      windDirectionUnit,
    in float     fixedWindsSpeed,
    in float     fixedWindsDirection,
    in StringSeq SurfClimoStations,
    in FileReferenceT metFile1,
    in FileReferenceT metFile2
);

void setMetBLT (
    in long      BLMETHOD,
    in long      surfaceMoisture,
    in float     dayMaxInversionHeight,
    in float     nightMaxInversionHeight,
    in float     dayMaxSurfaceHeatFlux,
    in float     nightMaxSurfaceHeatFlux,
    in float     canopyHeight,
    in float     surfaceRoughness,
    in float     surfaceAlbedo,
    in float     surfaceBowenRatio,
    in float     cloudFraction,
    in float     canopyFlowIndex
);

void setMetLSVT (
    in long      LSVMethod,
    in float     LSVTurb,
    in float     LSVLengthScale
);

void setMetPrecipT (
    in long      precipType,
    in long      precipClass
);

void setTerrainMCT (
    in float      swiftDeltaT,

```

```

        in longArray maxIterationsArray,
        in floatArray errorCriteriaArray,
        in floatArray verticalAdjustmentArray,
        in long      numberVerticalPoints,
        in FloatSeq   verticalGrid
    );

    void setTerrainT (
        in long      MCType,
        in TerrainMCT massConsistency,
        in FileReferenceT terrainFile
    );

    MetFlagsT getMetFlagsT();
    MetMetT getMetMetT ();
    MetBLT getMetBLT ();
    MetLSVT getMetLSVT ();
    MetPrecipT getMetPrecipT ();
    TerrainMCT getTerrainMCT ();
    TerrainT getTerrainT ();
} // interface WeatherAccessors

}; // module weatherT
}; }; }; };

#endif

```

3.5 SCIPUFFFACTORY AND SCIPUFFSERVER

ScipuffServer is a per-client server object wrapping HPACtool, the T&D engine. The IDL files listed below define the interface to ScipuffServer and ScipuffServerFactory.

As HPACtool is non-reentrant, each per-client ScipuffServer object must run in a separate process. ScipuffFactory takes care of spawning a new JVM for each client's ScipuffServer.

3.5.1 ScipuffServer Collaborations

A client interacts with ScipuffServer in a sequence of steps described below. The example programs in Appendix B illustrate the collaboration between a client and ScipuffServer.

1. Look up the ScipuffServerFactory object via the CORBA naming service (refer to Section 3.3.1). The `SCIPUFF_FACTORY_SERVICE_NAME` constant defined in the `server` IDL module provides the lookup name. The object returned from the naming service is of type `ScipuffServerFactory` as defined in the `scipuff` IDL module.
2. Call the `getInstance()` method of the factory object to obtain a per-client ScipuffServer object implementing the `ScipuffServer` interface. The factory is no longer needed and may be released via the CORBA object `_release()` method.

3. Begin polling for messages in a separate thread or process. Polls are made by calling the `poll()` `ScipuffServer` `poll()` method. When the server issues a message of type `HM_REPLY` it will block until a reply is sent from the client via call to the `reply()` method of `ScipuffServer`.
4. Call the `getDispersionCalculator()` method of the `ScipuffServer` object to initialize the T&D engine for calculation and retrieve a `DispersionCalculator` reference.
5. Call one of the calculation methods of the `DispersionCalculator` object, such as `calculate2()`.
6. Continue to poll until the server responds with a `HM_DISPERSION_END` message. If the message value (`fMessageValue` field of the `MessageT` structure) is `HM_SUCCESS`, the calculation succeeded and output or plot data may be requested. Otherwise, the calculation failed, and `HM_ERROR` messages returned from `poll()` calls will reveal the source of the error.

At this point the T&D engine may be released from calculation mode by calling the `terminateDispersionCalculator()` method on the `DispersionCalculator` object. The CORBA object method `_release()` should also be called. If no output will be retrieved, the server should be released via calls to `terminate()` (defined in the `ScipuffServer` interface) and then `_release()`.

Output is obtained via the following steps.

1. If necessary, obtain a factory object via the naming service and call its `getInstance()` method to obtain a `ScipuffServer` object, as described above.
2. Call the `getPlotGenerator()` method of the `ScipuffServer` object to initialize the T&D engine for output and return a `PlotGenerator` reference.
3. Retrieve available plot information with calls to `PlotGenerator` methods, `numPlotClasses()`, `numPlotTimes()`, `getPlotClasses()`, and `getPlotTimes()`. The objects filled in describe the plot classes, choices, kinds, and times available.
4. Retrieve data for a particular kind of plot or output with other `PlotGenerator` methods. First, an adaptive grid field must be created via the `createField()` method.
5. A typical sequence of calls for generating output for various available plot classes, choices, and kinds would be `contourCount()`, and `contourField()`.
6. When the plots for a particular field are complete, call the `deleteField()` method.
7. When output retrieval is complete, call the `terminatePlotGenerator()` and `_release()` methods.

3.5.2 effects.idl

```
#ifndef __effects__
#define __effects__
//-----
//      NAME:          effects.idl
-
```

```

//-----

//-----  

//      MODULE:          mil.dtra.hpac.server.effects  

//      NOTE:  

//          We don't build with -pkgPrefix, because this file is  

//          included in other IDLs.  

//-----  

module mil { module dtra { module hpac { module server {  

*****  

* Definitions of constants and structures from the HPACTool API related  

* to effects processing.  

* Model servers may need these classes, so we put them in  

* <tt>hpacshared.jar</tt>.  

*****  

module effects  

{  

    typedef sequence<float>  

        FloatArray;  

//-----  

//      Effect ID and Mask Contants  

//      EID_     - effect ID for computeEffect() calls  

//      EM_     - effect mask for IncidentT.fAvailableEffects  

//-----  

        /**  

         * Blast circle effect ID value in calls to  

         * <tt>IncidentModelServer.computeEffect()</tt>  

         */  

const long          EID_CIRCLE_BLAST = 1;  

        /**  

         * Blast circle bit mask;  

         * blast circle is the  

         * NWPN 1 PSI overpressure blast radius  

         */  

const long          EM_CIRCLE_BLAST = 0x01 << (EID_CIRCLE_BLAST - 1);  

        /**  

         * Thermal circle effect ID value in calls to  

         * <tt>IncidentModelServer.computeEffect()</tt>  

         */  

const long          EID_CIRCLE_THERM = 2;  

        /**  

         * Thermal circle bit mask;  

         * thermal circle is the  

         * NWPN 2 cal/cm2 thermal radius  

         */  

const long          EM_CIRCLE_THERM = 0x01 << (EID_CIRCLE_THERM - 1);  

        /**  

         * Prompt circle effect ID value in calls to

```

```

        * <tt>IncidentModelServer.computeEffect()</tt>
        */
const long EID_CIRCLE_PROMPT = 3;
        /**
         * Prompt circle bit mask;
         * prompt circle is the
         * NWPN 50 rad prompt radius
         */
const long EM_CIRCLE_PROMPT = 0x01 << (EID_CIRCLE_PROMPT - 1);

        /**
         * Casualty prompt effect ID value in calls to
         * <tt>IncidentModelServer.computeEffect()</tt>
         */
const long EID_CASUALTY_PROMPT = 4;
        /**
         * Casualty prompt bit mask;
         * this is the prompt casualty calculation
         */
const long EM_CASUALTY_PROMPT = 0x01 << (EID_CASUALTY_PROMPT - 1);

//-----
//      Effect Request Constants
//-----
/***
 * Request value passed to
 * <tt>IncidentModelServer.computeEffect()</tt>
 * specifying initialization of an effect
 * computation
 */
const long EID_INIT = 1;

        /**
         * Request value passed to
         * <tt>IncidentModelServer.computeEffect()</tt>
         * specifying computation of an effect
         */
const long EID_COMP = 2;

        /**
         * Request value passed to
         * <tt>IncidentModelServer.computeEffect()</tt>
         * specifying the end of an effect
         * computation
         */
const long EID_EXIT = 3;

//-----
//      Old NWPN effects array sizes
//      CPE_ refers to casualty prompt effect
//-----

```

```

const long          CPE_MAX_RADII = 23;
const long          CPE_MAX_SETTINGS = 2;
const long          CPE_MAX_VNTK = 6;

//-----
//      TYPE:          CasualtyPromptInitInT
//*****
*****/
struct CasualtyPromptInitInT
{
    long          fNumberVNTKs;
    sequence<long>    fVNTKArray;
}; // CasualtyPromptInitInT

//-----
//      TYPE:          CasualtyPromptInitOutT
//*****
*****/
struct CasualtyPromptInitOutT
{
    long          fNumberWeapons;
    sequence<FloatArray>    fProtectionFactors;
    FloatArray        fProbabilityFactors;
}; // CasualtyPromptInitOutT

//-----
//      TYPE:          CasualtyPromptCompInT
//*****
*****/
struct CasualtyPromptCompInT
{
    long          fNumberWeapons;
}; // CasualtyPromptCompInT

//-----
//      TYPE:          CasualtyPromptCompOutT
//*****
*****/
struct CasualtyPromptCompOutT
{
    float         fWeaponLocation[3];
    float         fWeaponBurstTime;
    sequence<FloatArray>    fCasualtyProbabilityRadii;
    sequence<FloatArray>    fFatalityProbabilityRadii;
    sequence<float>        fWeaponCasualtyRadii;
    sequence<float>        fWeaponFatalityRadii;
}; // CasualtyPromptCompOutT

```

```

//-----
//      TYPE:          CasualtyPromptCompOutTArray
//*****
*****/ 
typedef sequence<CasualtyPromptCompOutT> CasualtyPromptCompOutTArray;

//-----
//      TYPE:          CircleInitOutT
//*****
*****/ 
struct CircleInitOutT
{
    long      fNumberWeapons;
}; // CircleInitOutT

//-----
//      TYPE:          CircleCompInT
//*****
*****/ 
struct CircleCompInT
{
    long      fNumberWeapons;
}; // CircleCompInT

//-----
//      TYPE:          CircleCompOutT
//*****
*****/ 
struct CircleCompOutT
{
    float     fLocation[ 3 ];
    float     fTime;
    float     fRadius;
}; // CircleCompOutT

//-----
//      TYPE:          CircleCompOutTArray
//*****
*****/ 
typedef sequence<CircleCompOutT> CircleCompOutTArray;
}; // effects

}; }; }; };

#endif

```

3.5.3 plot.idl

```

#ifndef __plot__
#define __plot__
//-----
//      NAME:          plot.idl
//-----
#include "project.idl"

//-----
//      MODULE:        mil.dtra.hpac.server.plot
//-----
module mil { module dtra { module hpac { module server {
//****************************************************************************
* Definitions of constants and structures from the HPACTool API related
* to plotting.
* Model servers should not need these classes, so we put them in
* <tt>scipuffdefs.jar</tt>.
*****}
module plot
{
    typedef mil::dtra::hpac::server::project::TimeT
        TimeT;

//-----
//      Contour Type IDs
//-----
        /**
         * Left hand contours id
         */
const long        HP_LEFTHAND = -1;
        /**
         * Open contours id
         */
const long        HP_OPEN = 0;
        /**
         * Right hand contours id
         */
const long        HP_RIGHHAND = 1;

//-----
//      Output Type IDs
//-----
        /**
         * Effect output id
         */
const long        HP_EFFECT = -1;
        /**
         * Concentrations output id
         */
const long        HP_CONC = 0;

```

```

        /**
         * Met output id
         */
const long      HP_MET = 1;
                /**
                 * Deposition output id
                 */
const long      HP_DEP = 2;
                /**
                 * Dose output id
                 */
const long      HP_DOS = 3;

//-----
// Plot Operations?
//-----
        /**
         * Open contour id (0)
         */
const long      OPEN_CONTOUR = 0;
                /**
                 * Close contour id (2^11)
                 */
const long      CLOSE_CONTOUR = 0x01 << 11;
                /**
                 * Lat-lon output id (2^12)
                 */
const long      LATLON_OUTPUT = 0x01 << 12;

//-----
// Plot Category Constants
//-----
        /**
         * Number of plot categories
         */
const long      HP_NUMCAT = 6;
                /**
                 * ??
                 */
const long      HP_CATTYPE = 0;
                /**
                 * Surface (Fortran) index (1)
                 */
const long      HP_SURF = 1;
                /**
                 * Surface Slice (Fortran) index (2)
                 */
const long      HP_SSLICE = 2;
                /**
                 * Horizontal Slice (Fortran) index (3)
                 */

```

```

const long          HP_HSLICE = 3;
                   /**
 * Vertically Integrated Slice
 * (Fortran) index (4)
 */
const long          HP_VINT = 4;
                   /**
 * Vertical Slice (Fortran) index (5)
 */
const long          HP_VSLICE = 5;
                   /**
 * Tabular Output (Fortran) index (6)
 */
const long          HP_TABLE = 6;

//-----
// Plot Time Indexes
//-----
/***
 * No plot times (Fortran) index (0)
 */
const long          HP_NOTIME = 0;
                   /**
 * Puff times (Fortran) index (1)
 */
const long          HP_PUFFTIME = 1;
                   /**
 * Surface times (Fortran) index (2)
 */
const long          HP_SRFTIME = 2;
                   /**
 * Met times (Fortran) index (3)
 */
const long          HP_METTIME = 3;
                   /**
 * Rad times (Fortran) index (4)
 */
const long          HP_RADTIME = 4;

//-----
// Population Mask Values
//-----
/***
 * No area/population calculations
 */
const long          HP_OFF = 0;
                   /**
 * Perform population calculations
 */
const long          HP_ON = 0x01 << 0;
                   /**

```

```

        * Area instead of population calculations
        */
const long      HP_AREA = 0x01 << 1;
                /**
                 * Expected area/population calculations
                 * instead of actual
                 */
const long      HP_EXPECT = 0x01 << 2;

//-----
//    Miscellaneous Constants
//-----

        /**
         * ???
         */
const float     HP_SPV = -1.0e-36;

//-----
//    Plot Value Constants
//-----

const long      PLOT_NULL = 0;
const long      PLOT_ON = 1;
const long      PLOT_OFF = -1;

const long      PLOT_LOG = 1;
const long      PLOT_LIN = 2;
const long      PLOT_USER = 0;

//-----
//    TYPE:          HPACCategoryClassT
//***** Record describing a category-class combination.
//***** struct HPACCategoryClassT
{
    /**
     * Flags availability of the
     * category-class combination
     */
boolean        fIsAvailable;

    /**
     * Flags whether a plot type must
     * be selected
     */
boolean        fIsTypeRequired;
}; // HPACCategoryClassT

//-----

```

```

//      TYPE:          HPACCATEGORYCLASSLIST
//*****
*****typedef sequence<HPACCATEGORYCLASST>          HPACCATEGORYCLASSLIST;
*****
```

```

//-----  

//      TYPE:          HPACCLASSCHOICET
//*****  

/* Record describing a class-choice combination.  

*****
```

```

struct HPACCLASSCHOICET
{
    /**
     * Flags availability of the
     * class-choice combination
     */
    boolean        fIsAvailable;

    /**
     * Flags whether this class-choice combo
     * supports class kind selection
     */
    boolean        fSupportsKindSelection;

    /**
     * Index (1-based) into the kind array for
     * kinds applicable to this class-choice combo
     */
    long          fKindIndex;

    /**
     * Count of kinds applicable to this
     * class-choice combo
     */
    long          fNumberKinds;

    /**
     * ID of plot times which apply to this
     * class-choice combo
     */
    long          fTimeId;

    /**
     * Flags whether this class-choice combo
     * supports user time entry
     */
    boolean        fSupportsUserTime;
}; // HPACCLASSCHOICET
```

```

//-----  

//      TYPE:          HPACCLASSCHOICELIST
//*****
```

```

/*****-
*typedef sequence<HPACClassChoiceT>          HPACClassChoiceTList;
*****-

//-----
//      TYPE:          HPACContourElementT
//*****-
* Definition of a contour element.
*****-
struct HPACContourElementT
{
    float           fContourValue;
    float           fAssociatedValue;
    string          fContourLabel;
    float           fArea;
    float           fPopulation;
}; // HPACContourElementT

//-----
//      TYPE:          HPACContourElementTList
//*****-
*****-
typedef sequence<HPACContourElementT>          HPACContourElementTList;
*****-

//-----
//      TYPE:          HPACContourHeaderT
//*****-
* Definition of a contour header.
*****-
struct HPACContourHeaderT
{
    long            fNumber;
    float           fScaleFactor;
    long            fLabelMode;
    long            fDrawMode;
    string          fUnits;
}; // HPACContourHeaderT

//-----
//      TYPE:          HPACPointT
//*****-
* Point/location defintion.
*****-
struct HPACPointT
{
    /**
     * X coordinate
     */
    float           fx;

```

```

        /**
         * Y coordinate
         */
float          fY;
}; // HPACPointT

//-----
//      TYPE:          HPACPointTList
//*****
*****typedef sequence<HPACPointT>           HPACPointTList;

//-----
//      TYPE:          HPACSliceT
//*****
/* Definition of a plot slice.
*****struct HPACSliceT
{
    long             fNumberPointsInLine;
    HPACPointT       fStartPoint;
    HPACPointT       fEndPoint;
}; // HPACSliceT

//-----
//      TYPE:          ReferenceT
//*****
/* Location reference.
*****struct ReferenceT
{
    float           fX;
    float           fY;
    float           fLatitude;
    float           fLongitude;
}; // ReferenceT

//-----
//      TYPE:          HPACFieldCoordinateT
//*****
/* Definition of a coordinate for a field.
*****struct HPACFieldCoordinateT
{
    /**
     * Coordinate mode id
     * (<tt>HD_LATLON</tt>, <tt>HD_CARTESIAN</tt>,
     * <tt>HD_UTM</tt>, <tt>HD_METERS</tt>)
     */

```

```

long          fCoordinateMode;
             /**
              * UTM zone
              */
long          fUTMReferenceZone;
             /**
              * Reference b/w lat-lon and UTM
              */
ReferenceT    fLatLonReference;
             /**
              * Slice
              */
HPACSliceT    fSliceBaseLine;
};

// HPACFieldCoordinateT

//-----
//      TYPE:          HPACLineT
//*****
struct HPACLineT
{
    long          fContourIndex;
    long          fstartIndex;
    long          fNumberLocationPoints;
    long          fLineSense;
};

// HPACLineT

//-----
//      TYPE:          HPACLineTList
//*****
typedef sequence<HPACLineT>          HPACLineTList;

//-----
//      TYPE:          HPACPlotFieldT
//*****
struct HPACPlotFieldT
{
    long          fFieldCategory;
    long          fFieldClass;
    long          fFieldChoice;
    long          fFieldKind;
    long          fTimeIndex;
    float         fUserTime;
    long          fHazard;
    long          fMaxCells;
    long          fMaxRefinementLevels;
    long          fActualRefinementLevels;
    float         fResolution;
};

```

```

long fInterpolationType;
HPACFieldCoordinateT
{
    fCoordinates;
    string fUnits;
    string fProjectName;
    //string fProjectPath;
}; // HPACPlotFieldT

//-----
// TYPE:          HPACPlotFieldNodeT
// ****
struct HPACPlotFieldNodeT
{
    long fID;
    float fX;
    float fY;
    float fZ;
    float fXRes;
    float fYRes;
    float fValue;
}; // HPACPlotFieldNodeT

//-----
// TYPE:          HPACPlotFieldNodeTList
// ****
typedef sequence<HPACPlotFieldNodeT> HPACPlotFieldNodeTList;

//-----
// TYPE:          HPACPlotFieldTriangleT
// ****
struct HPACPlotFieldTriangleT
{
    long fID;
    long fNodeIdentA;
    long fNodeIdentB;
    long fNodeIdentC;
}; // HPACPlotFieldTriangleT

//-----
// TYPE:          HPACPlotFieldTriangleTList
// ****
typedef sequence<HPACPlotFieldTriangleT> HPACPlotFieldTriangleTList;

//-----

```

```

//      TYPE:          HPACPlotTypeT
//*****
*****struct HPACPlotTypeT
{
    long           fPlotType;
    float          fData;
    long           fAreaMode;
}; // HPACPlotTypeT

//-----
//      TYPE:          HPACTimeT
//*****
*****struct HPACTimeT
{
    TimeT          fTime;
    long            fNumberItems;
    string          fTimeDisplay;
}; // HPACTimeT

//-----
//      TYPE:          HPACTimeTList
//*****
*****typedef sequence<HPACTimeT>          HPACTimeTList;
}; // plot

}; }; };

#endif

```

3.5.4 scipuff.idl

```

#ifndef __scipuff__
#define __scipuff__
//-----
//      NAME:          scipuff.idl
//-----
#include "project.idl"
#include "plot.idl"
#include "server.idl"
#include "effects.idl"
#include "weatherT.idl"

//-----
//      MODULE:         mil.dtra.hpac.server.scipuff
//-----
module mil { module dtra { module hpac { module server {

```

```

module scipuff
{
    typedef mil::dtra::hpac::server::effects::FloatArray
        FloatArray;

    typedef mil::dtra::hpac::server::EnviroBLT
        EnviroBLT;

    typedef mil::dtra::hpac::server::project::AuditT
        AuditT;

    typedef mil::dtra::hpac::server::project::FlagsT
        FlagsT;

    typedef mil::dtra::hpac::server::plot::HPACCATEGORYCLASSLIST
        HPACCATEGORYCLASSLIST;

    typedef mil::dtra::hpac::server::plot::HPACCLASSCHOICETLIST
        HPACCLASSCHOICETLIST;

    typedef mil::dtra::hpac::server::plot::HPACCONTOURELEMENTT
        HPACCONTOURELEMENTT;

    typedef mil::dtra::hpac::server::plot::HPACCONTOURELEMENTLIST
        HPACCONTOURELEMENTLIST;

    typedef mil::dtra::hpac::server::plot::HPACCONTOURHEADERT
        HPACCONTOURHEADERT;

    typedef mil::dtra::hpac::server::plot::HPACLINELIST
        HPACLINELIST;

    typedef mil::dtra::hpac::server::plot::HPACFIELDCOORDINATET
        HPACFIELDCOORDINATET;

    typedef mil::dtra::hpac::server::plot::HPACPLOTFIELDNODET
        HPACPLOTFIELDNODET;

    typedef mil::dtra::hpac::server::plot::HPACPLOTFIELDNODELIST
        HPACPLOTFIELDNODELIST;

    typedef mil::dtra::hpac::server::plot::HPACPLOTFIELDTRIANGLET
        HPACPLOTFIELDTRIANGLET;

    typedef mil::dtra::hpac::server::plot::HPACPLOTFIELDTRIANGLELIST
        HPACPLOTFIELDTRIANGLELIST;

    typedef mil::dtra::hpac::server::plot::HPACPLOTFIELDT
        HPACPLOTFIELDT;

    typedef mil::dtra::hpac::server::plot::HPACPLOTTYPET
        HPACPLOTTYPET;
}

```

```

typedef mil::dtra::hpac::server::plot::HPACPointT
    HPACPointT;

typedef mil::dtra::hpac::server::plot::HPACPointTList
    HPACPointTList;

typedef mil::dtra::hpac::server::plot::HPACTimeTList
    HPACTimeTList;

typedef mil::dtra::hpac::server::project::LimitT
    LimitT;

typedef mil::dtra::hpac::server::project::OptionsT
    OptionsT;

typedef mil::dtra::hpac::server::IncidentT
    IncidentT;
typedef sequence<IncidentT> IncidentTList;
typedef sequence<any> AnyList;

typedef mil::dtra::hpac::server::project::SpatialDomainT
    SpatialDomainT;

typedef mil::dtra::hpac::server::project::TemporalDomainT
    TemporalDomainT;

typedef mil::dtra::hpac::server::release::TimeT
    TimeT;

typedef mil::dtra::weather::shared::data::weatherT::WeatherT
    WeatherT;

//-----
//      Scipuff Factory Service Name
//*****
*****const string          SCIPUFF_FACTORY_SERVICE_NAME = "ScipuffServerFactory";

//-----
//      HPAC Success and Failure constants
//-----
const long      HPAC_SUCCESS = 1;
const long      HPAC_FAILURE = -1;

//-----
//      HPAC message type constants used in MessageT.fMessageType
//-----
const long      HM_CHECK      = 0;
const long      HM_SETWAIT    = 1;
const long      HM_RELEASEWAIT = 2;
const long      HM_SETCLOCK   = 3;

```

```

const long    HM_STEPCLOCK      = 4;                                // HM_STRUCTURE = 256
const long    HM_STOPCLOCK      = 5;

const long    HM_RELEASE        = 256 + 1;                            // HM_STRUCTURE + 1
//cont long   HM_UPDATEREL     = 256 + 2;                            // HM_STRUCTURE + 2
//cont long   HM_COMPUTEEFF     = 256 + 32;                           // HM_STRUCTURE + 32

const long    HM_SYNC           = 128 + 1;                            // HM_ARRAY = 128
//const long   HM_INITEFF       = 128 + 32;                           // HM_ARRAY + 32
//const long   HM_HASEFF        = 128 + 33;                           // HM_ARRAY + 33

const long    HM_START          = 512 + 1;                            // HM_INTEGER = 512
const long    HM_PROGRESSBAR    = 512 + 3;                            // HM_INTEGER + 3
const long    HM_BUTTONSTATE    = 512 + 5;                            // HM_INTEGER + 5

const long    HM_PROGRESSMSG    = 1024;                               // HM_MESSAGE = 1024
const long    HM_INFO           = 1024 + 1;                           // HM_MESSAGE + 1
const long    HM_ERROR          = 1024 + 2;                           // HM_MESSAGE + 2
const long    HM_REPLY          = 1024 + 3;                           // HM_MESSAGE + 3
const long    HM_STOP           = 1024 + 4;                           // HM_MESSAGE + 4
const long    HM_BTONTAG        = 1024 + 5;                           // HM_MESSAGE + 5

const long    HM_NO_NEW_MESSAGE = -1;
const long    HM_EMPTY_MESSAGE  = -2;

const long    HM_DISPERSION_END = -99;

//-----
//      HPAC reply constants. Used as parameter values in the reply() method.
//-----

const long    HPAC_AFFIRMATIVE_REPLY = 1;
const long    HPAC_NEGATIVE_REPLY  = -1;

//-----
//      HPAC button click constants
//-----

const long    HC_NO_BUTTON      = 0;
const long    HC_LEFT_BUTTON    = 201;
const long    HC_MIDDLE_BUTTON  = 202;
const long    HC_RIGHT_BUTTON   = 203;

//-----
//      TYPE: MessageT
*****-
*****-
struct MessageT

```

```

{
long      fMessageType;
long      fMessageValue;
long      fMessageParameter;
string    fMessageString1;
string    fMessageString2;
string    fMessageString3;
string    fOriginator;
}; // struct MessageT

typedef sequence<MessageT> MessageTList;

//-----
//   TYPE:   CircleEffectInputT
// ****
***** / 
union CircleEffectInputT switch (long)
{
case mil::dtra::hpac::server::effects::EID_INIT:
    long                                fCircleInitIn;
case mil::dtra::hpac::server::effects::EID_COMP:
    mil::dtra::hpac::server::effects::CircleCompInT  fCircleCompIn;
case mil::dtra::hpac::server::effects::EID_EXIT:
    long                                fCircleExitIn;
}; // CircleEffectInputT

//-----
//   TYPE:   CircleEffectOutputT
// ****
***** / 
union CircleEffectOutputT switch (long)
{
case mil::dtra::hpac::server::effects::EID_INIT:
    mil::dtra::hpac::server::effects::CircleInitOutT      fCircleInitOut;
case mil::dtra::hpac::server::effects::EID_COMP:
    mil::dtra::hpac::server::effects::CircleCompOutTArray fCircleCompOut;
case mil::dtra::hpac::server::effects::EID_EXIT:
    long                                fCircleExitOut;
}; // CircleEffectOutputT

//-----
//   Typedef's of array types
//----- 

typedef sequence<octet>          ByteArray;
typedef sequence<long>            IntArray;
typedef sequence<string>          StringList;
typedef sequence<HPACCClassChoiceTList> ClassChoice2D;
typedef sequence<HPACCCategoryClassTList> CategoryClass2D;
typedef sequence<IntArray>        I2D;
typedef sequence<I2D>             I3D;

```

```

//-----
//      EXCEPTION:      ScipuffException
//*****
***** exception ScipuffException
{
    string          fMessage;
    MessageT        fOptionalMessageT;
};

//-----
//      EXCEPTION:      PlotException
//*****
***** exception PlotException
{
    string          fMessage;
    MessageT        fOptionalMessageT;
};

//-----
//      EXCEPTION:      CircleEffectException
//*****
***** exception CircleEffectException
{
    string          fMessage;
};

//-----
//      INTERFACE:      DispersionCalculator
//*****
***** interface DispersionCalculator
{
//-----
//      METHOD:      addIncident()
//*****
* Adds incidents and their model incident data to the project. Should
* be called once for each <tt>IncidentT</tt> in the project. The incident
* and model_incident parameters are IDL inouts, making them appear as
* <tt>Holder</tt> objects in Java. These objects will not be changed
* directly by <tt>addIncident</tt> but they will be passed on to the
* model server's <tt>updateIncident</tt> method which may change them.
* After all incidents are added to the project, the client should call
* <tt>calculate</tt>.
*
* @param incident          (the holder for) the <tt>IncidentT</tt> struct

```

```

*
* @param model_incident      to be added
*                            (the holder for) the model-specific data (aka
*                            "private block") for this incident
*
* @param flags                parameters used by the tool engine to define a
*                            project's type of dispersion calculation. Must
*                            be the same for each incident added and the
*                            same as passed to calculate. Or else. No check
*                            is made.
*
* @throws ScipuffException    if already calculating or if any internal
*                            error occurs
*
* @see #calculate calculate
***** */
void
addIncident(
    inout IncidentT           incident,
    inout any                  model_incident,
    in FlagsT                 flags
)
    raises (ScipuffException);

//-----
//  METHOD:          calculate()          -
//*****
* Initializes the HPAC Tool Engine, calls each model's updateIncident()
* with the "last chance" flag set, and starts the HPAC Tool Engine
* running. This will asynchronously return after the tool engine is
* successfully started. Future interaction with the tool engine must
* occur via the <tt>poll</tt> method.
*
* @param weather            the weather
* @param limits              array size limits, etc. for the tool engine
* @param options             more parameters used by the tool engine
* @param flags               even more such parameters
* @param temporal_domain    specification of project start and end times
* @param spatial_domain     specification of project spatial domain
* @param max_time_step       maximum time step in seconds allowed for SCIPUFF
* @param output_interval    time interval in seconds for SCIPUFF output
*
* @return                   true if project was converted to UTM, false
*                          if not
* @throws ScipuffException  if already calculating, if any errors occurred
*                          during the update incident threads started by
*                          addIncident(), if HPAC Tool Engine cannot be
*                          initialized, if a lat/lon-to-UTM coordinate
*                          conversion error occurs, or if any internal
*                          error occurs
***** */
boolean
calculate(
    in WeatherT               weather,

```

```

    in LimitT           limits,
    in OptionsT         options,
    in FlagsT          flags,
    inout TemporalDomainT temporal_domain,
    inout SpatialDomainT spatial_domain,
    in float            max_time_step,
    in float            output_interval
)
raises (ScipuffException);

//-----
// METHOD:           calculate2() -
// *****
* Performs a dispersion calculation using HPACtool. The following steps
* are performed: calls updateIncident() with the "last chance" flag set
* for each <tt>IncidentT</tt> provided, builds the input needed by HPACtool
* from the parameters provided, converts the project to UTM coordinates if
* required, initializes HPACtool, calls the HPACtool function HPACNewProject
* to create HPACtool project files, and, assuming HPACNewProject was
* successful, calls the HPACtool function HPACRunProject to start SCIPUFF
* running. Will return asynchronously after the tool engine is successfully
* started. Future interaction with the tool engine must occur via the <tt>
* poll</tt> method.
*
* @param incidents      array of the <tt>IncidentT</tt> objects for
*                       this project
* @param model_incidents array of the model-specific data (aka "private
*                       blocks") corresponding to the <tt>IncidentT</tt>
*                       's of the project. Must be provided as an array
*                       of CORBA "any" objects, each element of which is
*                       an "any" containing the model-specific data.
* @param weather         the weather (duh)
* @param limits           array size limits, etc. for the tool engine
* @param options          more parameters used by the tool engine
* @param flags            even more such parameters
* @param temporal_domain specification of project start and end times
* @param spatial_domain   specification of project spatial domain
* @param max_time_step    maximum time step in seconds allowed for SCIPUFF
* @param output_interval  time interval in seconds for SCIPUFF output
*
* @return                 true if project was converted to UTM, false
*                       if not
* @throws ScipuffException
* 
* 
* 
* 
*****/
boolean
calculate2(
    in IncidentTList     incidents,

```

```

in AnyList           model_incidents,
in WeatherT          weather,
in LimitT            limits,
in OptionsT           options,
in FlagST             flags,
inout TemporalDomainT temporal_domain,
inout SpatialDomainT spatial_domain,
in float               max_time_step,
in float               output_interval
) raises
(ScipuffException);

boolean
calculateWithUrban(
    in IncidentTList      incidents,
    in AnyList             model_incidents,
    in WeatherT            weather,
    in LimitT              limits,
    in OptionsT             options,
    in FlagST               flags,
    inout TemporalDomainT temporal_domain,
    inout SpatialDomainT   spatial_domain,
    in float                 max_time_step,
    in float                 output_interval,
    in string                udm_params,
    in string                uwm_params
)
raises (ScipuffException);

//-----
// METHOD:           currentTerrain()
// ****
float
currentTerrain( in HPACPointT location )
    raises (ScipuffException);

//-----
// METHOD:           currentWeather()
// ****
void
currentWeather(
    in HPACPointT          location,
    inout EnvironmentT       environment,
    out EnviroBLT            boundary_layer
)
raises (ScipuffException);

```

```

//-----
//  METHOD:          getSubstrates()           -
//*****                                                 *****
* Retrieves the list of available substrates for the seconday
* evaporation model.
*
* @return    a <tt>String</tt> array of the substrate list. If no
*            substrates are available, returns a zero-length array.
*
* @throws   ScipuffException if the native call to HPACGetSubstrates
*           fails
*****                                                 *****/
StringList
getSubstrates()
    raises (ScipuffException);

//-----
//  METHOD:          processButtonClick()        -
//*****                                                 *****
* Notifies the server that a button (left, middle, or right) was
* clicked on the user interface.
*****                                                 *****/
void
processButtonClick( in long click )
    raises (ScipuffException);

//-----
//  METHOD:          restartFromPrevious()       -
//*****                                                 *****
* Creates a new project from an existing one and runs the new project.
* All of the parameters must be provided because ScipuffServer maintains
* no state from the previous run. However, each parameter labeled MUST
* NOT CHANGE must contain values identical to the corresponding parameters
* in the previous run. No check is performed, but attempting to restart
* with changed parameters is not supported and may produce difficult-to-
* understand errors.
*
* @param previous_username           username of the previous project from
*                                   which to restart
* @param previous_projectname        name of the previous project from which
*                                   to restart
* @param time_to_restart_from       output time of the previous project
*                                   from which to restart
* @param previous_incident_ts       array of the <tt>IncidentT</tt> structs
*                                   from the previous project. MUST NOT
*                                   CHANGE.
* @param previous_model_incidents  array of the model-specific data (aka
*                                   "private blocks") corresponding to the
* <tt>IncidentT</tt>'s of the previous
* project. These must be provided as an
* array of CORBA "any" objects, each

```

```

*
*
* @param previous_scipuff_mode
* @param new_weather_t
* @param new_limit_t
* @param new_options_t
* @param new_tm_domain
* @param new_sp_domain
* @param new_max_time_step
* @param new_output_interval
* @param audit
*
* @return
* @throws ScipuffException
******/
```

element of which is an "any" containing a model-specific data. MUST NOT CHANGE. scipuff calculation mode flag -- either HF_FAST (1), HF_HAZARD(2), or HF_DUAL(4) -- used in previous project. MUST NOT CHANGE.

the weather (duh). May change.

array size limits, etc. for the tool engine. May change but values should only be increased.

more parameters used by the tool engine. May change.

specified temporal domain of the project. Only the <tt>fEndTime</tt> field may change.

specified spatial domain of the project. May change.

max time step for SCIPUFF. May change.

time interval in seconds for SCIPUFF output. May change.

AuditT structure specifying analyst, etc.

true if project was converted to UTM, false if not

if any error occurs.

```

boolean
restartFromPrevious(
    in string
    in string
    in float
    in IncidentTList
    in AnyList
    in long
    in AuditT
    in WeatherT
    in LimitT
    in OptionsT
    inout TemporalDomainT
    inout SpatialDomainT
    in float
    in float
)
raises (ScipuffException);
```

```

boolean
restartFromPreviousWithUrban(
    in string
    in string
    in float
    in IncidentTList
    in AnyList
    in long
    in AuditT
)
raises (ScipuffException);
```

```

    in WeatherT           new_weather_t,
    in LimitT             new_limit_t,
    in OptionsT            new_options_t,
    inout TemporalDomainT new_tm_domain,
    inout SpatialDomainT  new_sp_domain,
    in float               new_max_time_step,
    in float               new_output_interval,
    in string              udm_params,
    in string              uwm_params
)
raises (ScipuffException);

//-----
//  METHOD:          resumeCalculation()  -
// ****
* Resumes calculation of a previously "Stopped" or "Halted" run. This
* will re-initialize the HPAC Tool Engine and call HPACRunProject to
* resume the previous run where it had stopped. Like calculate(), this
* will asynchronously return after the tool engine is successfully
* started. Future interaction with the tool engine must occur via the
* <tt>poll</tt> method. All of the parameters labeled MUST NOT CHANGE
* must contain values identical to the corresponding parameters in the
* previously stopped or halted run. No check is performed, but attempting
* to resume with changed parameters is unsupported and may produce
* difficult-to-understand errors.
*
* @param incidents      array of <tt>IncidentT</tt> objects used in the
*                       run to be resumed. MUST NOT CHANGE.
* @param model_incidents array of CORBA any objects containing the
*                         model-specific data for each <tt>IncidentT</tt>
*                         in the <tt>incidents</tt> array. MUST NOT
*                         CHANGE.
* @param weather         the weather (duh). MUST NOT CHANGE.
* @param limits           array size limits, etc. for the tool engine. May change
*                         but values should only be increased.
* @param temporal_domain specification of project start and end times.
*                         Only the <tt>fEndTime</tt> field may change.
* @param spatial_domain   specification of project spatial domain. MUST
*                         NOT CHANGE.
* @param max_time_step    maximum time step in seconds allowed for
*                         SCIPUFF. May change.
* @param output_interval  time interval in seconds for SCIPUFF output.
*                         May change.
* @param audit            AuditT structure specifying analyst, etc.
*
* @return                 true if project was converted to UTM, false
*                         if not
* @throws ScipuffException if already calculating, if HPAC Tool Engine
*                         cannot be initialized, if a lat/lon-to-UTM
*                         coordinate conversion error occurs, or if any
*                         internal error occurs
* ****
/

```

```

boolean
resumeCalculation(
    in IncidentTList           incidents,
    in AnyList                  model_incidents,
    in WeatherT                weather,
    in LimitT                  limits,
    inout TemporalDomainT     temporal_domain,
    inout SpatialDomainT      spatial_domain,
    in float                   max_time_step,
    in float                   output_interval
)
raises (ScipuffException);

//-----
// METHOD:          terminateDispersionCalculator()          -
//*****************************************************************************
* Should be called by the client when finished with the server. This is
* a hook for the server to clean up memory allocations, unload shared
* objects, etc. This <b>must</b> be called before the server
* process can be re-used by another client or by the same client for
* plot generation.
//*****************************************************************************
void
terminateDispersionCalculator()
    raises (ScipuffException);
}; // interface DispersionCalculator

//-----
// INTERFACE:       PlotGenerator                           -
//*****************************************************************************
interface PlotGenerator
{
//-----
// METHOD:          computeCircleEffects()                 -
//*****************************************************************************
void
computeCircleEffects(
    in long           incident_index,
    in long           effect_id,
    in long           request,
    in CircleEffectInputT ceit,
    out CircleEffectOutputT ceot
)
raises (CircleEffectException);

//-----
// METHOD:          contourCount()                         -
//*****************************************************************************

```

```

* Counts the number of contour lines and contour points to be created
* by <tt>contourField</tt>.
*
* @param sag_field_id      the SAG field identifier
* @param plot_field        describes the field used to generate the
8          SAG field specified by <tt>sage_field_id</tt>
* @param plot_type         describes the type of plot to be generated
*                          from the field data
* @param contour_header    describes the array of contour elements in
* @param contour_list      <tt>contour_list</tt>
* @param mode               array of contour elements
*                           contour generation mode. One of
*                           <tt>OPEN_CONTOUR, CLOSE_CONTOUR</tt>, or
*                           <tt>LATLON_OUTPUT</tt>
* @param num_lines          upon return contains the number of lines to be
*                           generated
* @param num_points         upon return contains the number of points
*                           generated
*
* @throws PlotException     if any error occurs
*****void
contourCount(
    in long                  sag_field_id,
    in HPACPlotFieldT        plot_field,
    in HPACPlotTypeT         plot_type,
    in HPACContourHeaderT   contour_header,
    in HPACContourElementTList contour_list,
    in long                  mode,
    out long                 num_lines,
    out long                 num_points
)
    raises (PlotException);

//-----
//  METHOD:           contourField()  -
//*****
* @throws PlotException     if any error occurs
*****void
contourField(
    in long                  sag_field_id,
    in HPACPlotFieldT        plot_field,
    in HPACPlotTypeT         plot_type,
    in HPACContourHeaderT   contour_header,
    inout HPACContourElementTList contour_list,
    in long                  mode,
    out HPACLineTList        lines,
    out HPACPointTList       points
)
    raises (PlotException);

```

```

//-----
//  METHOD:          createField()           -
/* ****
* Creates an HPAC plot field.
*
* @param plot_field describes the plot field to be created
* @param class_data additional plot field class data as required by the <tt>
*                      fFieldCategory</tt> and <tt>fFieldClass</tt> elements of
*                      the <tt>field</tt>
*
* @return            the SAG grid identifier
*
* @throws PlotException    if any error occurs
***** */

long
createField(
    inout HPACPPlotFieldT      plot_field,
    in  FloatArray              class_data
)
    raises (PlotException);

//-----
//  METHOD:          deleteField()           -
/* ****
* @throws PlotException    if any error occurs
***** */

void
deleteField( in long sag_field_id )
    raises (PlotException);

//-----
//  METHOD:          exitCircleEffects()      -
/* ****
***** */

void
exitCircleEffects()
    raises (CircleEffectException);

//-----
//  METHOD:          getField()               -
/* ****
* @throws PlotException    if any error occurs
***** */

void
getField(
    in long                  sag_field_id,
    in HPACPPlotFieldT       plot_field,
    in HPACPPlotTypeT        plot_type,
    out HPACPPlotFieldNodeTList plot_nodes,
)
    raises (PlotException);

```

```

    out HPACPlotFieldTriangleTList plot_triangles
)
raises (PlotException);

//-----
// METHOD:           getFieldDomain() -
//*****************************************************************************
* @throws PlotException      if any error occurs
*****/
void
getFieldDomain(
    in long          sag_field_id,
    out long         nx,
    out long         ny,
    out float        xMin,
    out float        yMin,
    out float        dx,
    out float        dy
)
raises (PlotException);

//-----
// METHOD:           getFieldMinMax() -
//*****************************************************************************
* @throws PlotException      if any error occurs
*****/
void
getFieldMinMax(
    in long          sag_field_id,
    in HPACPlotTypeT plot_type,
    out float         mean_min,
    out float         mean_max,
    out float         var_min,
    out float         var_max,
    out float         plot_min,
    out float         plot_max
)
raises (PlotException);

//-----
// METHOD:           getFieldSize()
//*****************************************************************************
* @throws PlotException      if any error occurs
*****/
void
getFieldSize(
    in long          sag_field_id,
    in HPACPlotFieldT plot_field,
    in HPACPlotTypeT plot_type,
    out long         number_nodes,

```

```

        out long           number_triangles
    )
    raises (PlotException);

//-----
//  METHOD:          getFieldTable( )           -
//***** *****
* @throws PlotException      if any error occurs
***** *****
void
getFieldTable(
    in HPACPlotFieldT      plot_field,
    in FloatArray           class_data,
    out StringList          titles,
    out StringList          columns,
    out StringList          rows,
    out I3D                 table_3d
)
raises (PlotException);

//-----
//  METHOD:          getFieldTableSize()         -
//***** *****
* @throws PlotException      if any error occurs
***** *****
void
getFieldTableSize(
    in HPACPlotFieldT      plot_field,
    in FloatArray           class_data,
    out long                ntables,
    out long                ncolumns,
    out long                nrows
)
raises (PlotException);

//-----
//  METHOD:          getFieldValue(             -
//***** *****
* @throws PlotException      if any error occurs
***** *****
void
getFieldValue(
    in long                 sag_field_id,
    in HPACPlotTypeT        plot_type,
    in float                x_point,
    in float                y_point,
    out float               v_point
)
raises (PlotException);

```

```

//-----
//  METHOD:      getFieldValues(          -
// **** @throws PlotException      if any error occurs
// ****
void
getFieldValues(
    in long           sag_field_id,
    in HPACPlotTypeT plot_type,
    in long           n_points,
    in FloatArray     x_points,
    in FloatArray     y_points,
    out FloatArray    v_points
)
    raises (PlotException);

//-----
//  METHOD:      getPlotClasses()          -
// **** @throws PlotException      if any error occurs
// ****
void
getPlotClasses(
    out StringList    classes,
    out StringList    choices,
    out StringList    kinds,
    out CategoryClass2D category_classes,
    out ClassChoice2D class_choices,
    out HPACFieldCoordinateT project_coordinates
)
    raises (PlotException);

//-----
//  METHOD:      getPlotTimes()          -
// **** @throws PlotException      if any error occurs
// ****
void
getPlotTimes(
    out HPACTimeTList puff_times,
    out HPACTimeTList surf_times,
    out HPACTimeTList met_times,
    out long          neff_times
)
    raises (PlotException);

//-----
//  METHOD:      initCircleEffects()      -
// ****

```

```
*****
long
initCircleEffects()
    raises (CircleEffectException);

//-----
//  METHOD:          numPlotClasses()           -
/*****
* @throws PlotException      if any error occurs
*****/
void
numPlotClasses(
    out long          nclasses,
    out long          nchoices,
    out long          nkinds
)
    raises (PlotException);

//-----
//  METHOD:          numPlotTimes()            -
/*****
* @throws PlotException      if any error occurs
*****/
void
numPlotTimes(
    out long          npuff_times,
    out long          nsurf_times,
    out long          nmet_times,
    out long          neff_times
)
    raises (PlotException);

//-----
//  METHOD:          popAreaField()            -
/*****
* @throws PlotException      if any error occurs
*****/
void
popAreaField(
    in long          sag_field_id,
    in HPACPlotFieldT plot_field,
    in HPACPlotTypeT plot_type,
    in HPACContourHeaderT contour_header,
    out HPACContourElementTList contour_list
)
    raises (PlotException);

//-----
//  METHOD:          queryCircleEffects()       -

```

```

/********************* /
boolean
queryCircleEffects(
    in long      incident_index,
    in long      effect_id
)
raises (CircleEffectException);

//-----
// METHOD:          terminatePlotGenerator() -
/********************* /
* Should be called by the client when finished with the server. This is
* a hook for the server to clean up memory allocations, unload shared
* objects, etc. This <bold>must</bold> be called before the server
* process can be re-used by another client or by the same client for
* plot generation.
/********************* /
void
terminatePlotGenerator()
    raises (PlotException);
} // interface PlotGenerator

//-----
// INTERFACE:      ScipuffServer -
/********************* /
interface ScipuffServer
{
//-----
// METHOD:          deleteProjectFiles() -
/********************* /
void
deleteProjectFiles(
    in string      username,
    in string      projectname,
    in long       request
)
raises (ScipuffException);

//-----
// METHOD:          getDispersionCalculator() -
/********************* /
* Gets and activates a CORBA dispersion calculation interface, which
* is required for making any DispersionCalculator method calls.
*
* @throws ScipuffException if any problem occurs during activation
/********************* /
DispersionCalculator

```

```

getDispersionCalculator()
    raises (ScipuffException);

//-----
//  METHOD:          getPlotGenerator()           -
// *****
* Gets and activates a PlotGenerator CORBA interface, which is required
* for making any PlotGenerator method calls.
*
* @param max_grid_cells_per_surface the maximum number of grid cells
*                                     per surface field. Used for creating
*                                     horizontal/vertical slices. May be
*                                     0 in which case a default value will
*                                     be assigned by the HPAC Tool Engine.
* @param incidents             list of <tt>IncidentT</tt>'s that can do prompt
*                               effects. Must include at least all the incidents
*                               that can do prompt effects, but may optionally
*                               include other incidents that do not support
*                               prompt effects.
* @param model_incidents       array of CORBA any objects containing the
*                               model-specific data for each <tt>IncidentT</tt>
*                               in the <tt>incidents</tt> array.
*
* @throws PlotException        if any problem occurs during activation for plot
*                             generation
*****/
PlotGenerator
getPlotGenerator(
    in long                 max_grid_cells_per_surface,
    in IncidentTList        incidents,
    in AnyList               model_incidents
) raises (PlotException);

//-----
//  METHOD:          poll()                    -
// *****
* Obtains current status information from the running tool engine.
*****/
MessageTList
poll()
    raises (ScipuffException);

//-----
//  METHOD:          reply()                  -
// *****
* Provides a reply to the server when a poll message indicated that a
* reply was required.
*
* @param reply         one of HPAC_AFFIRMATIVE_REPLY or HPAC_NEGATIVE_REPLY.
*                      Other values will be ignored.

```

```
*****
void
reply( in long reply )
    raises (ScipuffException);

//-----
// METHOD:           shutdown() -
/*****
* Used by ScipuffServerFactory to shut down a per-client ScipuffServer
* JVM process. Not intended to be called by any remote client. For
* internal use only. If CORBA had access modifiers, this would be private.
* See the shutdown() method on ScipuffServerFactory for clients to request
* shutdown of inactive servers.
*
* @param immediate  if true, the shutdown will be fast; if false, the
*                     shutdown will wait scipuffserver.JVMkilldelay
*                     seconds before shutting down, allowing time for the
*                     use to read the final messages, if any, directed to
*                     the server's window, if showing. The property
*                     scipuffserver.JVMkilldelay is read from
*                     hpacserver.properties.
*
* @throws ScipuffException if any problem occurs during shutdown
*****
void
shutdown( in boolean immediate )
    raises (ScipuffException);

//-----
// METHOD:           terminate() -
/*****
* Terminates and cleans up a ScipuffServer instance. Should be called by
* the client when finished with the server. This is a hook for the server
* to clean up memory allocations, unload shared objects, etc.
*
* @throws ScipuffException if any problem occurs. Can probably be ignored.
*****
void
terminate()
    raises (ScipuffException);
}; // interface ScipuffServer

//-----
// INTERFACE:        ScipuffAll -
/*****
*****
```

```
interface ScipuffAll :
    ScipuffServer, DispersionCalculator, PlotGenerator
{
```

```

// METHOD:          activate() -
//****************************************************************************
* Activates and initializes an inactive server. Not intended to be
* called by a remote client. For use by the server factory only. If
* CORBA had access modifiers, this would be private.
*****-
void
activate( in string user, in string project )
    raises (ScipuffException);

//-----
// METHOD:          isAvailable() -
//****************************************************************************
* Returns whether or not the server is available for activation. Not
* intended to be called by a remote client. For use by the server
* factory only. If CORBA had access modifiers, this would be private.
*
* @return true if the server is available, false if not
*****-
boolean
isAvailable();
}; // interface ScipuffAll

//-----
// INTERFACE:      ScipuffServerFactory -
//****************************************************************************
*****-
interface ScipuffServerFactory
{
//-----
// METHOD:          deactivated() -
//****************************************************************************
* Used to notify ScipuffServerFactory that ScipuffServer is being
* deactivated. Not intended to be called by any remote client. For
* internal use only. If CORBA had access modifiers, this would be private.
*****-
void
deactivated(
    in ScipuffAll server,
    in string server_name
)
    raises (ScipuffException);

//-----
// METHOD:          deleteProject() -
//****************************************************************************
*****-
void
deleteProject(
    in string      username,

```

```

        in string      projectname
    )
    raises (ScipuffException);

//-----
//  METHOD:          exportProject()           -
//*****                                                 *****/
ByteArray
exportProject(
    in string      username,
    in string      projectname
)
raises (ScipuffException);

//-----
//  METHOD:          getInstance()            -
//*****                                                 *****/
* Returns an instance of ScipuffServer
*****                                                 *****/
ScipuffServer
getInstance(
    in string user_name,
    in string project_name
)
raises (ScipuffException);

//-----
//  METHOD:          getSubstrates()          -
//*****                                                 *****/
* Retrieves the list of available substrates for the secondary evaporation
* model. This list is read directly from the land use data file instead of
* asking HPACtool for the list. There is a similar <tt>getSubstrates</tt>
* method in the <tt>DispersionCalculator</tt> interface, but it requires
* that HPACtool be already initialized. This method provides the list of
* substrates without requiring initialization of HPACtool. The location of
* the land use data file is obtained from the scipuffserver.LandUseDataFile
* property in the hpacserver.propsURL properties file.
*
* @return a <tt>String</tt> array of the substrate list. If no
* substrates are available, returns a zero-length array.
*
* @throws ScipuffException if hpacserver.propsURL is not specified or
* cannot be read, if scipuffserver.LandUseDataFile is not
* specified or cannot be read, if the land use data file contains
* no substrate data, or if the data is improperly formatted.
*
*****                                                 *****/
StringList
getSubstrates()

```

```

        raises (ScipuffException);

//-----
//  METHOD:          importProject()
//*****-
*****-
void
importProject(
    in string      username,
    in string      projectname,
    in ByteArray   zipfile
)
raises (ScipuffException);

//-----
//  METHOD:          shutdown()
//*****-
/* Shuts down all per-client ScipuffServer objects in the inactive
 * server pool.
*****-
void
shutdown()
    raises (ScipuffException);
}; // interface ScipuffServerFactory
}; // scipuff

}; }; };

#endif

```

3.5.5 ScipuffServer System Properties

There are a number of system properties which may be set via -D arguments to the java command to control ScipuffServer and ScipuffServerFactory settings. Refer to Section 3.2.1 for a listing of the Windows launch script. These properties are described in tables 3.3 and 3.4, respectively.

3.6 MATERIAL SERVER

The MaterialServer IDL interface is defined in the *material.idl* file listed in Section 3.4.2. Three methods are defined. The *getMaterial()* method creates a material object containing properties and attributes for the specified standard material in the HPAC database. Each material is stored in a file with a *.mtl* extension in the *server/ data/ materials/* subdirectory under the root directory of the HPAC installation.

A complete list of all material names in the database may be retrieved via the *getMaterialList()* method. The *getPropertiesByKey()* method is used by the HPAC client application to retrieve default plot settings for a material and is not generally useful to other clients.

| Property | Type | Description |
|--|---------|---|
| <i>scipuffserver.verbose</i> | boolean | if true, diagnostics are logged directly to the terminal |
| <i>scipuffserver.verbosecallbacks</i> | boolean | turns on diagnostics for every HPACtool call-back |
| <i>scipuffserver.verbosetoolinputs</i> | boolean | turns on verbose printing of inputs to many HPACtool calls, particularly releases and materials |
| <i>scipuffserver.verbosepollmessages</i> | boolean | turns on verbose printing of every message returned during poll calls from the client |
| <i>scipuffserver.verboseplotmessages</i> | boolean | turns on verbose printing of input and output for HPACtool plotting functions |
| <i>scipuffserver.verbosecircleeffects</i> | boolean | turns on verbose printing of I/O for circle effects |
| <i>scipuffserver.verbosecomputeeffects</i> | boolean | turns on verbose printing of I/O for compute effects calls |
| <i>scipuffserver.verbosetiming</i> | boolean | turns on verbose printing of timing information |
| <i>scipuffserver.verbosewrapper</i> | boolean | turns on verbose wrapper diagnostics |
| <i>scipuffserver.updateIncidentMaxWaitSecs</i> | int | sets the maximum amount of time Scipuff-Server will wait on a model server to return from an updateLastChance() call |

Table 3.3: ScipuffServer system properties

| Property | Type | Description |
|--|--------|---|
| <i>ScipuffServerFactory.shutdownPassword</i> | string | key that must be supplied by a client given permission to call the shutdownServer-VM() method to terminate the server process |
| <i>scipuffserver.maxIdleServerSeconds</i> | int | specifies the maximum amount of time in seconds an idle ScipuffServer instance will be allowed to live. Defaults to 600 seconds or 10 minutes. One idle ScipuffServer will always be kept for faster server startup. Additional ScipuffServers will be created as demanded by multiple clients, if any. |
| <i>scipuffserver.initialServersInPool</i> | int | specifies the initial number of ScipuffServers to create in the server pool at startup. Defaults to 1. |

Table 3.4: ScipuffServerFactory system properties

3.7 RADFILE MERGER

The RadFileMerger object exists to merge "rad files" generated by the Nuclear Facility, Nuclear Weapon Incident, and Radiological Weapon source models. Each rad file must be merged into a project rad file for input to the T&D engine. All of this happens behind the scene for HPAC clients. Thus, it is unlikely a client will ever need to access RadFileMerger directly. It is a candidate for obsolescence in a future HPAC version.

3.7.1 radfile.idl

```
#ifndef __radfile__
#define __radfile__
//-----
//      NAME:          radfile.idl
//-----
//#include "scipuff.idl"

//-----
//      MODULE:        mil.dtra.hpac.server.radfile
//-----
module mil { module dtra { module hpac { module server {
***** * Definition of the interfaces for rad file merger services.
* Packaged in <tt>hpacshared.jar</tt>.
***** */
module radfile
```

```

{
typedef sequence<string>      StringList;

//-----
//  CONSTANT:      RAD_FILE_MERGER_SERVICE_NAME
//*****
* Default name under which the rad file merger service is bound.
*****/
const string      RAD_FILE_MERGER_SERVICE_NAME = "RadFileMerger";

//-----
//  EXCEPTION:      RadFileNotFoundException
//*****
* Definition of an exception in model server side processing.
*****/
exception RadFileNotFoundException
{
    string      fMessage;
};

//-----
//  INTERFACE:      RadFileMerger
//*****
* Services merging radfiles into the single rad file for a project.
*****/
interface RadFileMerger
{

//-----
//  METHOD:          getRadFileContents()
//*****
* Returns the contents of the rad file managed for the project
* identified by the user_name and project_name. This method is to
* support <tt>ScipuffServer</tt> in obtaining the rad file in order to
* hand it to <tt>HPACtool</tt> and works if the <tt>RadFileMerger</tt>
* object is running on a separate machine.
*
* @param user_name user name used to identify the project
* @param project_name project name used to identify the project
* @return      contents of the project rad file
* @exception RadFileNotFoundException on error accessing the rad file
*****/
string
getRadFileContents(
    in string      user_name,
    in string      project_name
)
raises ( RadFileNotFoundException );
}

```

```

//-----
//  METHOD:          getRadFileURL()           -
//*****
* Builds a URL for accessing the rad file managed for the project
* identified by the user_name and project_name. This method is to
* support <tt>ScipuffServer</tt> in obtaining the rad file. The
* URL could use a protocol such as HTTP if the <tt>RadFileMerger</tt>
* server object is running on a machine with a co-resident Web
* server, or it could be a "file" URL if it has been established
* a priori that the <tt>RadFileMerger</tt> lives on the same machine
* as the <tt>ScipuffServer</tt> instance.
*
* @param user_name user name used to identify the project
* @param project_name project name used to identify the project
* @return          URL to the project rad file
* @exception RadFileNotFoundException on error accessing the rad file
*****/
string
getRadFileURL(
    in string      user_name,
    in string      project_name
)
    raises ( RadFileNotFoundException );

```



```

//-----
//  METHOD:          merge()           -
//*****
* Merges the provided rad file into the current project master rad file,
* which is identified by user_name and project_name. If the project master
* rad file does not yet exist, it is created with the provided contents.
* Note that as of this writing, the merger implementation will convert
* all material names to UPPERCASE! These case conversions will not be
* reported in the returned array of name changes.
* <p>
* This method is to be called by model servers and clients after creating
* a rad file.
* </p>
*
* @param user_name user name used to identify the project
* @param project_name project name used to identify the project
* @param rad_file_contents contents of the rad file to merge into the
*                           project rad file
* @param model_prefix the prefix representing the model that created the
*                     rad file to be merged. The material names in the rad
*                     file must begin with the same prefix. In order to avoid
*                     material namespace conflicts, this prefix should be
*                     chosen to be unique to the model. For example, for a
*                     material named "mat" and model "mod", the <tt>
*                     model_prefix</tt> could be "mod" and the material name
*                     used in the rad file would be "mod-mat". An exception
*                     will be generated if the material name(s) in the rad

```

```

*           file do not begin with <tt>model_prefix</tt>.
* @param  rename_materials_flag  true if the materials in the rad
*                               file are to be renamed, false otherwise
* @return an array of material name changes in the sequence old-name,
*        new-name, old-name, new-name, etc. -- one pair for each material
*        in the original rad file that had its name changed. If the length
*        of the returned array is zero, then no names were changed.
* @exception RadFileException  on error processing the rad files
*****StringList
merge(
    in string      user_name,
    in string      project_name,
    in string      rad_file_contents,
    in string      model_prefix,
    in boolean     rename_materials_flag
)
raises ( RadFileException );
}; // RadFileMerger
}; // radfile
};

};

#endif

```

3.8 STCALC SERVER

StcalcServer takes descriptions of radiological releases and produces a "rad file". Currently, it is only used by the Nuclear Facility Model and exists as a server solely to isolate it's native Fortran code on the server. It is also a candidate for obsolescence in a future HPAC version.

3.8.1 stcalc.idl

```

#ifndef __stcalc__
#define __stcalc__
-----
//      NAME:          stcalc.idl
//      PURPOSE:        -
//      -
//include "server.idl"

-----
//      MODULE:        mil.dtra.hpac.server.stcalc
//      -
module mil { module dtra { module hpac { module server {
module stcalc
{
//      -
//      Staclc Service Name
//      -
}}
```

```

const string           STCALC_SERVICE_NAME = "StcalcServer";

//-----
//   TYPE:          StcalcException
//   PURPOSE:       -
//-----
exception StcalcException
{
    string      message;
    long        errno;
}; // exception StcalcException

//-----
//   TYPE:          STReleaseMaterialT
//   PURPOSE:       -
//-----
struct STReleaseMaterialT
{
    string      fMaterialName;
    float       fStartTime;
    string      fStartTimeUnits;
    float       fDuration;
    string      fDurationUnits;
}; // struct STReleaseMaterialT

//-----
//   TYPE:          STReleaseMaterialListT
//   PURPOSE:       -
//-----
typedef sequence<STReleaseMaterialT>
                STReleaseMaterialListT;

//-----
//   TYPE:          STReleaseListT
//   PURPOSE:       -
//-----
struct STReleaseListT
{
    long        fNumberMaterials;
    float       fTotalDuration;
    STReleaseMaterialListT   fReleaseMaterialList;
}; // struct STReleaseList

//-----
//   INTERFACE:     StcalcServer
//   PURPOSE:       -
//-----
interface StcalcServer
{
    //-----
    //   METHOD:       createRadFile()
    //   PURPOSE:      -
    //-----

```

```

//           Creates a rad file from the input file provided and      -
//           returns a string containing the contents of the created   -
//           rad file.
//-----
string createRadFile (
    in string username,
    in string projectname,
    in string inputfile,
    in string caller
) raises (StcalcException);

//-----
// METHOD:          getDisplayableSourceTerm()                      -
// PURPOSE:         Creates a source term file from the provided input file -
//                  and returns a string containing the source term contents-
//                  in a human-readable form.                           -
//-----
string getDisplayableSourceTerm (
    in string username,
    in string projectname,
    in string inputfile,
    in string caller
) raises (StcalcException);

//-----
// METHOD:          createReleaseList()                                -
// PURPOSE:         Creates a release list from the rad file contents   -
//                  provided and returns the list as a ReleaseList struct.  -
//-----
STReleaseListT createReleaseList (
    in string username,
    in string projectname,
    in string rad_file_contents
) raises (StcalcException);
}; // interface StcalcServer
}; // module stcalc

}; }; };
#endif

```

3.9 CHEMICAL-BIOLOGICAL FACILITY

The Chemical-Biological Facility source model processes incidents describing conventional weapon attacks on chemical or biological facilities. It includes submodels for computing the interactions of weapons and facilities. The submodels provide alternatives for calculation time versus results integrity and are exposed as three server objects in addition to the standard HPAC incident model server.

3.9.1 Services

Simplest of the submodels is Damage Category, specified in the *DamCat.idl* file. Only the quantity and type of agent involved are required as inputs. The Simple Weapons Facility Interaction (*SWFI.idl*) and Detailed Weapons Facility Interaction (*DWFI.idl*) models require detailed inputs. Additional servers provided with this model are a material classification and agent property server, specified in *AgentsList.idl* and a weapon properties server, specified in *WeaponServer.idl*.

The Chemical-Biological Facility model provides the common HPAC source model collaborations described in Section 3.3.2. Clients may use the default incidents and make minor changes in agent types and quantities. Alternatively, a more sophisticated client may exercise the full gamut of services to specify an incident in the following sequence of steps.

1. Obtain the agent name list from the material classification server for the type of submodel (detailed or simple) desired using the `AgentsList` service and the `getSWFIAgentsList()` and/or `getDWFIAgentsList()` methods.
2. Obtain the warhead name list from the `WeaponList` service's `getWeaponsList()` method.
3. Retrieve specific data or properties for the selected agent via the `getSWFIMatValues()` or `getDWFIMatValues()` method of the `AgentsList` service.
4. Retrieve properties of the selected warhead via the `getWeaponValues()` method of the `WeaponList` service.
5. Provide inputs for the chosen submodel, `InputsDC` for Damage Category, `InputsSWFI` for Simple Weapon Facility Interaction, or `Inputs` for Detailed Weapon Facility Interaction.
6. Run the submodel calculation by calling the `DWFI.Update()`, `SWFI.UpdateSWFI()`, or `DamCat.UpdateDC()`.
7. Create releases suitable for input to the T&D calculation by calling the `updateIncident()` method of the `CBFacWeaponServer` service.

The naming service binding name for the Chemical-Biological Weapon model factory service is the only one specified in the IDL files. Names of all the services are listed in Table 3.5.

Note also that all of the services identified above are singleton objects. With the exception of `CBFacServerFactory` (which produces per-client `CBFacWeaponServer` instances), none of them should be assumed thread-safe. That is, simultaneous access by multiple clients may yield unpredictable results.

3.9.2 AgentsList.idl

```
module mil { module dtra { module models { module CBFac {
  module AgentsList {
    struct DCAgents {
      sequence<string> agentName;
    };
    struct SWFIAgents {
      sequence<string> agentName;
    };
  }
}}}}
```

| Interface | Name |
|--------------------|--------------------|
| AgentsList | MTIMatServer |
| CBFacServerFactory | CBFacServerFactory |
| CloudTrans | CloudTransCalc |
| DamCat | DamCatCalc |
| DWFI | DWFICalc |
| SWFI | SWFICalc |
| WeaponList | WeaponServer |

Table 3.5: Chemical-Biological Weapon model services

```

{
    sequence<string> agentName;
};

struct DWFIAgents
{
    sequence<long> agentNum;
    sequence<string> agentName;
};

struct SWFIAgentMatValues
{
    long iChem;      // 1 if chem , 0 if bio
    long iGas;       // 1 if gas , 0 if liquid
    float liqDen[2];
    float antCoef[3];
    float bprep;
    float bprer;
    float e0puv;
    float eovrr;
    float eovrrp;
    float htcomb;
    float fastr;
    float htvap;
    float mwvap;
    float surfaceTension;
    float viscosityAgent;
};

struct DWFIAgentMatValues
{
    long iChem;      // 1 if chem , 0 if bio
    long iGas;       // 1 if gas , 0 if liquid
}

```

```

float dryBioDen;
float liqDen[2];
float surfaceTension;
float viscosityAgent;
long numBins;
float binBnd[50];
};

struct agentsIn
{
    sequence<string> agentsInName;
};
struct agentsOut
{
    sequence<string> agentsOutName;
};
interface Filter
{
    void Apply(in agentsIn agentsin,out agentsOut agentsout);
};

interface DamCatFilter : Filter
{
};
interface SWFIcatFilter : Filter
{
};
interface DWFIFilter : Filter
{
};

interface AgentsList
{
    attribute DCAGENTS dcagents;
    attribute SWFIAGENTS swfiagents;
    attribute string CBfacBaseDirectory;
    DCAGENTS getDamCatAgentsList();
    SWFIAGENTS getSWFIAGENTSList();
    DWFIAgents getDWFIAgentsList();
    SWFIAGENTMATVALUES getSWFIAGENTMATVALUES(in string sMat);
    DWFIAgentMATVALUES getDWFIAgentMATVALUES(in string sMat);
    long getDamCatMATVALUES(in string smat);
    string getLongName(in string smat);
    void setDWFIAgentsList();
    void setSWFIAGENTSList();
    void setDamCatAgentsList();
    // string getMatProp(in string sMat,in string sprop);
    void ApplyFilter(in Filter filter);
};

```

```
}; }; }; };
};
```

3.9.3 CBFactoHpac.idl

```
#ifndef __CBFactoHpac__
#define __CBFactoHpac__
-----
//      NAME:          CBFactoHpac.idl
//      PURPOSE: idl to wrap up CBFac stuff for HPAC
-----
#include "../../server/src/server.idl"
#include "../../Models/CBFac/DWFI/server/src/DWFI.idl"
#include "../../Models/CBFac/SWFI/server/src/SWFI.idl"
#include "../../Models/CBFac/DamCat/server/src/DamCat.idl"
#include "../../Models/CBFac/CloudTrans/server/src/CloudTrans.idl"

-----
//      MODULE:        server
-----
module mil { module dtra { module hpac { module models { module CBFac {
module server
{
    typedef mil::dtra::models::CBFac::DWFI::Inputs Inputs;
    typedef mil::dtra::models::CBFac::DWFI::Results Results;
    // structs for transferring data in the true procedural tradition
    struct DWFIData{
        Inputs inputs;
        Results results;
    }; // total data for the DWFI calculation
    typedef mil::dtra::models::CBFac::SWFI::InputsSWFI InputssSWFI;
    typedef mil::dtra::models::CBFac::SWFI::ResultsSWFI ResultsSWFI;
    struct SWFIData{
        InputssSWFI inputs;
        ResultsSWFI results;
    }; // total data for the SWFI calculation
    typedef mil::dtra::models::CBFac::DamCat::InputsDC InputsDC;
    typedef mil::dtra::models::CBFac::DamCat::ResultsDC ResultsDC;
    struct DamCatData{
        InputsDC inputs;
        ResultsDC results;
    }; // total data for the DamCat calculation
    typedef mil::dtra::models::CBFac::CloudTrans::StateVarCT StateVarCT;
    struct CloudTransData{
        StateVarCT statevarct;
    }; // total data for the DamCat calculation
    typedef mil::dtra::hpac::server::IncidentModelServer IncidentModelServer;
interface CBFacWeaponServer : IncidentModelServer
{
}; // CBFacWeaponServer

-----
```

```

//      INTERFACE:      CBFacServerFactory
//-----
const string   CBFAC_FACTORY_SERVICE_NAME = "CBFacServerFactory";
interface CBFacServerFactory : mil::dtra::hpac::server::IncidentModelServerFactory
{
    };
    // CBFacServerFactory
} ; // CBFac
};};};};};

#endif

```

3.9.4 CloudTrans.idl

```

module mil { module dtra { module models { module CBFac {
module CloudTrans
{
    struct StateVarCT
    {
        long RandomCount;
        long RandomSeed;
        float RandomSpread;
        string URL;
    };
    interface CloudTrans
    {
        attribute StateVarCT statevarct;
        long getMatOK(in string smat);
    };
};};};};

};


```

3.9.5 DamCat.idl

```

module mil { module dtra { module models { module CBFac {
module DamCat
{
    struct StateVarDC
    {
        string agentName;
        long agentType;
        long igas;
    };
    struct InputsDC
    {
        long isever;
        float masfrc[4];
        long numfrc;
        float masinv;
        StateVarDC statevardc;
    };
    struct ResultsDC
    {
        long ierror;

```

```

        float masexp;
    };
interface DamCat
{
    attribute InputsDC inputdc;
    readonly attribute ResultsDC resultdc;
    boolean UpdateDC();
};
}; }; }; };
};

```

3.9.6 DWFI.idl

```

module mil { module dtra { module models { module CBFac {
module DWFI
{
    struct StateVarDWFI
    {
        string agentName;
        float dryBioDen;
        long agentType;
        long nonGenericFacilityType;
        long nonGenericConstructionType;
        long unconfinedContainerTypeChem;
        long unconfinedContainerTypeBio;
        long unconfinedContainerArrangement;
        long iInit;
        string MEAInfo_DBName;
        string MEAInfo_ModelID;
        string MEAInfo_AttackID;
        string MEAInfo_WeaponName;
        long MEAInfo_DamageLevel;
        string MEAInfo_UserName;
        string MEAInfo_Date;
        string MEAInfo_Time;
    };
    struct Facility
    {
        float FacVolume;
        boolean BRdataAvail;
        float BRVolume;
        float TotalVentArea;
        float BurstRoomVentArea;
        boolean SoftStructure;
        float AreaDoor;
        float shad;
        float shada;
        float cin;
        float abarmx;
        float range;
        float shadw;
        long iunder;
        float ddust;
    };
}
}
}
}
}

```

```

float avtlin;
float admdin;
float avrfin;
float avsdin;
float atotin;
float pqsin;
float mfbgin;
float avbrin;
float avbsin;
float t80in;
float t80ain;
};

struct WarHead
{
    long WeaponID;
    float MassBomb;
        float PenHoleDiameter;
        float MassDust;
    float FragVelocity;
};

struct Agent
{
    long AgentID;
    float MassInBurstRoom;
    float MassInAdjoiningRoom;
    float ContainerLengthOverDiam;
    float ContainerWallThickness;
    float ViscosityAgent;
    float SurfaceTension;
    float LiquidDensity[2];
    float bintop[50];
    long NumberBins;
    long igas;
    long iwet;
    float fliq;
};

struct RunConditions
{
    long iprob;
    float Temperature;
        float Pressure;
    long icode;
};

struct Type
{
    long Sub;
    long Data;
};

struct Inputs
{
    Facility facility;
    WarHead warHead;
    Agent agent;
}

```

```

RunConditions runconditions;
Type type;
StateVarDWFI statevardwfi;
};

struct Results
{
    long ierror;
    float mlexp[50];
    float mvexp;
    float mspill;
    float mvblxp;
    float mbexp;
    float mpexp;
    float mviabn[50];
    long icfnbk;
    long isftbk;
};

interface DWFI
{
    attribute Inputs input;
    attribute string CBfacBaseDirectory;
    readonly attribute Results result;
    boolean Update();
};

};

};

};

};
```

3.9.7 SWFI.idl

```

module mil { module dtra { module models { module CBFac {
module SWFI
{
    struct StateVarSWFI
    {
        string agentName;
        long agentType;
        long nonGenericFacilityType;
        long nonGenericConstructionType;
        long unconfinedContainerTypeChem;
        long unconfinedContainerTypeBio;
        long unconfinedContainerArrangement;
        long iInit;
        string MEAInfo_DBName;
        string MEAInfo_ModelID;
        string MEAInfo_AttackID;
        string MEAInfo_WeaponName;
        long MEAInfo_DamageLevel;
        string MEAInfo_UserName;
        string MEAInfo_Date;
        string MEAInfo_Time;
    };
}

struct FacilitySWFI
{
    string facilityName;
    long facilityType;
    long facilitySize;
    long facilityCapacity;
    long facilityStatus;
    string facilityAddress;
    string facilityContact;
    string facilityNotes;
}
```

```

{
    float FacVolume;
    boolean BRdataAvail;
    float BRVolume;
    float TotalVentArea;
    boolean SoftStructure;
    float AreaDoor;
    float shad;
    float shada;
    float cin;
    float abarmx;
    long itargt;
    long icano;
    long ncstot;
    long ncbase;
    long ncsht;
    float shadw;
    long iunder;
    float avtlin;
    float admdin;
    float avrfin;
    float avsdin;
    float atotin;
    float pqsin;
    float mfbgin;
    float avbrin;
    float avbsin;
    float t80in;
    float t80ain;
};

struct WarHeadSWFI
{
    long WeaponID;
    float MassBomb;
        float PenHoleDiameter;
        float MassDust;
    float range;
    float ebomb;
    float ddust;
};

struct AgentSWFI
{
    long igas;
    float MassInBurstRoom;
    float MassInAdjoiningRoom;
    float ContainerLengthOverDiam;
    float ContainerWallThickness;
    float ViscosityAgent;
    float SurfaceTension;
    float LiquidDensity[2];
    float antcoef[3];
    float e0puv;
    float mwvap;
}

```

```

        float fastr;
        float htvap;
        float htcomb;
        float faerbr;
        float faerar;
        float flkgbr;
        float flkgar;
        float eovrr;
        float bprer;
        float eovrrp;
        float bprep;
    };
    struct RunConditionsSWFI
    {
        long iprob;
        float Temperature;
        float Pressure;
        float constm;
        float constd;
        float efloat;
        long icode;
    };
    struct TypeSWFI
    {
        long Sub;
        long Data;
    };
    struct InputssWFI
    {
        FacilitySWFI facilityswfi;
        WarHeadSWFI warHeadsWFI;
        AgentSWFI agentswfi;
        RunConditionsSWFI runconditionsswfi;
        TypeSWFI typeswfi;
        StateVarSWFI statevarswfi;
    };
    struct ResultsSWFI
    {
        long ierror;
        float mlexp;
        float mvexp;
        float mspill;
        float mvblxp;
        float mbexp;
        float mpexp;
        float mmd;
        float sigma;
        float mspray;
        float rspray;
        float rfireb;
        float mvapfb;
        float mliqfb;
        float tempfb;
    };
}

```

```

        float mcontn;
        long icfnbk;
        long isftbk;
    };
    interface SWFI
    {
        attribute InputsSWFI input;
        attribute string CBFacBaseDirectory;
        readonly attribute ResultssSWFI result;
        boolean UpdateSWFI();
    };
}; }; }; };
};

```

3.9.8 WeaponServer.idl

```

module mil { module dtra { module models { module CBFac {
module WeaponServer
{
    struct WeaponNames
    {
        sequence<string> sWeaponName;
    };
    struct WeaponValues
    {
        long wid;
        long cvarID;
        string sMBLMName;
        float eBomb;
        float massBomb;
        float diamBomb;
        float frgVel;
        long iCode;
    };
    interface WeaponList
    {
        attribute string CBFacBaseDirectory;
        attribute WeaponNames weaponnames;
        attribute WeaponValues weaponvalues;
        WeaponNames getWeaponsList();
        WeaponValues getWeaponValues(in string sWeapon);
    };
};
} }; }; };
};

```

3.10 CHEMICAL-BIOLOGICAL WEAPON

3.10.1 ChemBioWeapon.idl

```

#ifndef __ChemBioWeapon__
#define __ChemBioWeapon__
//-----

```

```

//      NAME:          ChemBioWeapon.idl
//      HISTORY:
//      18 Aug 2000 dek@logicon.com
//      removed #pragma javapackage & replaced with module { ...
//      15 Mar 2000      babrice@cbdcom.apgea.army.mil
//      Converted from RadWeapon.idl
//              7 Mar 2000      dek@logicoan.com
//      Updated to remove units in dat structs--all assume default units
//              11 Feb 2000      leerw@ornl.gov
//      New data model, changes to base classes
//              20 Oct 1999      leerw@ornl.gov
//      PURPOSE:
//-----
#include "server.idl"

#pragma prefix "mil.dtra.hpac"

//-----
//      MODULE:          cbwpn
//-----
module mil { module dtra { module hpac { module models { module cbwpn {
module server
{

//-----
//      TYPE:          ChemBioWeaponIncidentT
//      PURPOSE:
//              The structure you defined to replace the private block
//-----
struct ChemBioWeaponIncidentT
{
    long      fModelType;
    string    fModelVersion;
    long      fSeed;
    float     fAgentPurity;
    float     fBuoyancy;
    float     fEfficiency;
    float     fFallAngle;
    float     fHeading;
    float     fHob;
    float     fHorzUncertainty;
    float     fISize;
    float     fLength;
    float     fLiquidDryFraction;
    float     fMass;
    float     fMmd;
    float     fMomentum;
    long      fNumber;
    float     fReleaseMass;
    float     fSigmaD;
}

```

```

float      fSpeed;
float      fSpread;
float      fVaporFraction;
float      fVertUncertainty;
string    fAgent;
string    fDeliverySystem;
string    fMunition;
};

// ChemBioWeaponIncidentT

// -----
// INTERFACE:      ChemBioWeaponServer
// -----
interface ChemBioWeaponServer : mil::dtra::hpac::server::IncidentModelServer
{
};

// ChemBioWeapon Server

// -----
// ChemBioWeapon Factory Service Name
// -----
const string      CHEMBIOWEAPON_FACTORY_SERVICE_NAME = "ChemBioWeaponFactory";

// -----
// INTERFACE:      ChemBioWeaponServerFactory
// -----

interface ChemBioWeaponServerFactory : mil::dtra::hpac::server::IncidentModelServerFactory
{
};

// ChemBioWeaponServerFactory
// server
// cbwpn
// models
// hpac
// dtra
// mil

#endif

```

3.11 MISSILE INTERCEPT

3.11.1 MissileIntercept.idl

```

#ifndef __MissileIntercept__
#define __MissileIntercept__

// -----
// NAME: MissileIntercept.idl
// -----

```

```

#include "server.idl"

#pragma prefix "mil.dtra.hpac"

//-----
// MODULE: mint
//-----

module mil { module dtra { module hpac { module models { module mint {
module server
{

//-----
// Constants
//-----

// The submodel settings are used as discriminators for the
// MissileInterceptIncidentT union.
const long SUBMODEL_LIVE = 0; // Live Session (a.k.a. RADAR scenario)
const long SUBMODEL_PLAN = 1; // Planning Session

// The damage level in the MissileInterceptIncidentT structure may be set to
// one of the following four constants. The specific settings for each level
// are used directly for control of a combo box, so care must be taken when
// changing the specific settings of each level.
const long DAMAGELEVEL_NONE = -1; // Indicates the level has yet to be set.
const long DAMAGELEVEL_LOW = DAMAGELEVEL_NONE + 1; // Low damage to target
const long DAMAGELEVEL_MODERATE = DAMAGELEVEL_LOW + 1; // Moderate damage
const long DAMAGELEVEL_HIGH = DAMAGELEVEL_MODERATE + 1; // High damage

// The threat type in the PairInfoT structure may be set to one of the
// following three constants. The setting of these constants was chosen to
// correspond with the values used by the PEELS database DLL.
const long THREATTTYPE_BULKTANK = 1;
const long THREATTTYPE_SUBMUNITION = 2;
const long THREATTTYPE_BOMBLET = 3;

//-----
// TYPE: AgentsListT
//-----
typedef sequence<string> AgentsListT;

//-----
// TYPE: InterceptLocationT
//-----
typedef sequence<float> InterceptLocationT;

//-----

```

```

// TYPE: PairInfoT
// PURPOSE: Provide initialization info required by the client from the server
//-----
struct PairInfoT
{
    long      fPairID;
    string    fPairName;
    long      fThreatType;        // set using THREATTYPE* constants
    float     fCloseSpeedMax;   // km/s
    float     fCloseSpeedMin;   // km/s
    float     fStrikeAngleMax;  // degrees
    float     fStrikeAngleMin;  // degrees
    float     fLookAngleMax;    // degrees
    float     fLookAngleMin;    // degrees
}; // PairInfoT

//-----
// TYPE: PairInfoListT
//-----
typedef sequence<PairInfoT> PairInfoListT;

//-----
// TYPE: LiveSessionDataT
//-----
struct LiveSessionDataT
{
    float     fThreatSpeed;     // 1000 <= value <= 3000 m/s
    float     fThreatAzimuth;   // 0 <= value < 360 degrees
    float     fThreatFPA;       // -90 <= value <= 90 degrees
    float     fInterceptorSpeed; // 1000 <= value <= 3000 m/s
    float     fInterceptorAzimuth; // 0 <= value < 360 degrees
    float     fInterceptorFPA;   // -90 <= value <= 90 degrees
}; // LiveSessionDataT

//-----
// TYPE: PlanSessionDataT
//-----
struct PlanSessionDataT
{
    float     fThreatLaunchLon;  // -180 <= value < 180 deg E
    float     fThreatLaunchLat;  // -90 <= value < 90 deg N
    float     fAimpointLon;     // -180 <= value < 180 deg E
    float     fAimpointLat;     // -90 <= value < 90 deg N
    float     fInterceptorLaunchLon; // -180 <= value < 180 deg E
    float     fInterceptorLaunchLat; // -90 <= value < 90 deg N
    float     fInterceptAltitude; // 0 <= value < 120,000 meters
}; // PlanSessionDataT

//-----

```

```

// TYPE: SubModelDataT
//-----
union SubModelDataT switch(long)
{
    case SUBMODEL_LIVE:
        LiveSessionDataT fLiveSession;
    case SUBMODEL_PLAN:
        PlanSessionDataT fPlanSession;
}; // SubModelDataT

//-----
// TYPE: MissileInterceptIncidentT
// PURPOSE: The structure you defined to replace the private block
//-----
struct MissileInterceptIncidentT
{
    long      fPairId;           // interceptor/target pair ID from PEELS
    string    fMaterial;         // hazardous material name
    long      fDamageLevel;      // set using DAMAGELEVEL* constants

    SubModelDataT fSubModel;     // either LiveSession or PlanSession

}; // MissileInterceptIncidentT

//-----
// INTERFACE: MissileInterceptServer
//-----
interface MissileInterceptServer : mil::dtra::hpac::server::IncidentModelServer
{
    void initClient(out PairInfoListT pairList,
                   out AgentsListT forSubmunitionList,
                   out AgentsListT forBombletList,
                   out AgentsListT forBulkList)
        raises (mil::dtra::hpac::server::ModelException);

    void planToLive(inout SubModelDataT scenario,
                   out InterceptLocationT location)
        raises (mil::dtra::hpac::server::ModelException);

}; // MissileInterceptServer

//-----
// MissileIntercept Factory Service Name
//-----
const string MISSILE_INTERCEPT_FACTORY_SERVICE_NAME =
    "MissileInterceptFactory";

//-----

```

```

// INTERFACE: MissileInterceptServerFactory
//-----
interface MissileInterceptServerFactory :
    mil::dtra::hpac::server::IncidentModelServerFactory
{
}; // MissileInterceptServerFactory

}; // server
}; // rwpn
}; // models
}; // hpac
}; // dtra
}; // mil

#endif

```

3.12 NUCLEAR WEAPON INCIDENT

3.12.1 nwi.idl

```

#ifndef __NWI__
#define __NWI__
//-----
//      NAME:          NWI.idlsrc
//      PURPOSE:        -
//-----
#include "server.idl"

#pragma prefix "mil.dtra.hpac"

//-----
//      MODULE:         nwi
//***** ****
module mil { module dtra { module hpac { module models {module nwi {
module server
{

//-----
//      Constants
//-----

// source choices
const long            EXPLOSIVE_DRIVEN_SOURCE = 0;
const long            FOSSIL_FIRE_DRIVEN_SOURCE = 1;
const long            PROPELLANT_DRIVEN_SOURCE = 2;
const long            USER_DEFINED_SOURCE = 3;

// user defined source submodel discriminators

```

```

// if source is USER_DEFINED_SOURCE
const long          AIRCRAFT_GROUND = 0;
const long          AIRCRAFT_AIR = 1;
const long          GROUND_TRANSPORT = 2;
const long          MISSILE_SILO = 3;

//-----
//      TYPE:          AircraftGroundDataT
//*****-
//*****-
//-----
```

```

struct AircraftGroundDataT
{
    // data specific to AircraftGroundDataT
    string   fAircraftType;
    string   fAircraftState;
    string   fFire;
    float    fDuration;
    float    fDistance;
    string   fWeaponCondition;

    float    fExplosionPercent;
    float    fBurnPercent;
    float    fMechPercent;
    float    fNonePercent;
    long     fDispersalMechanism;
}; // AircraftGroundDataT

//-----
//      TYPE:          AircraftAirDataT
//*****-
//*****-
//-----
```

```

struct AircraftAirDataT
{
    // data specific to AircraftAirDataT
    string   fAircraftType;
    float    fAircraftHeight;
    string   fSoilType;
    string   fFire;
    float    fDuration;
    float    fDistance;
    string   fWeaponCondition;

    float    fExplosionPercent;
    float    fBurnPercent;
    float    fMechPercent;
    float    fNonePercent;
    long     fDispersalMechanism;
}; // AircraftAirDataT

```

```

//-----
//      TYPE:          GroundTransportDataT
//*****
*****struct GroundTransportDataT
{
    // data specific to GroundTransportDataT
    string    fTransportType;
    string    fAnotherVehicle;
    string    fImpactType;
    float     fTransportSpeed;
    string    fFire;
    float     fDuration;
    float     fDistance;
    string    fWeaponCondition;

    float     fExplosionPercent;
    float     fBurnPercent;
    float     fMechPercent;
    float     fNonePercent;
    long      fDispersalMechanism;
}; // GroundTransportDataT

//-----
//      TYPE:          MissileSiloDataT
//*****
*****struct MissileSiloDataT
{
    // data specific to MissileSiloDataT
    string    fLocationSilo;
    string    fBurn;
    string    fMissileType;
    string    fWeaponCondition;
    float     fDistanceFromPropellant;
    float     fDurationPropellantBurn;

    float     fExplosionPercent;
    float     fBurnPercent;
    float     fMechPercent;
    float     fNonePercent;
    long      fDispersalMechanism;
}; // MissileSiloDataT

//-----
//      TYPE:          SubModelDataT
//*****
*****// SubModelDataT is one of the four struct defined above
union SubModelDataT  switch( long )
{

```

```

case AIRCRAFT_GROUND:
    AircraftGroundDataT      fAircraftGround;

case AIRCRAFT_AIR:
    AircraftAirDataT        fAircraftAir;

case GROUND_TRANSPORT:
    GroundTransportDataT    fGroundTransport;

case MISSILE_SILO:
    MissileSiloDataT        fMissileSilo;
} // SubModelDataT

//-----
//   TYPE:          NWIIncidentT
// ****
// NWIIncidentT is the server NWI part of an overall incident
// the rest of an overall incident is in IncidentT.
// NWIIncidentT consists of data common to all sources
// and a selected submodel.
struct NWIIncidentT
{
    // data common to all sources

    string          fModelVersion; // HPAC nwi module version

    // specify which 1 of 4 sources
    long            fSourceType;
    // one of:
    // EXPLOSIVE_DRIVEN_SOURCE
    // FOSSIL_FIRE_DRIVEN_SOURCE
    // PROPELLANT_DRIVEN_SOURCE
    // USER_DEFINED_SOURCE

    float           fCalcRadius;     // km
    long            fCloudShine;
    float           fExposureTime;   // hr

    long            fNumberWeapons;
    float           fPuMass;
    float           fHEMass;
    string          fTypePu;
    string          fTypeHE;
    string          fReleaseBasis;
    string          fSpecificWeapon;

    // a particular submodel if the source is USER_DEFINED_SOURCE
    // 1 of 4
    SubModelDataT   fSubModel;
    // one of:
    // AirCraftGroundDataT
    // AircraftAirDataT

```

```

// GroundTransportDataT
// MissileSiloDataT

}; // NWIIncidentT

//-----
// INTERFACE:      NWIServer
//***** *****
***** *****
interface NWIServer : mil::dtra::hpac::server::IncidentModelServer
{
}; // NWIServer

//-----
//      NWI Factory Service Name
//-----
const string          NWI_FACTORY_SERVICE_NAME = "NWIFactory";

//-----
// INTERFACE:      NWIServerFactory
//***** *****
***** *****
interface NWIServerFactory :
    mil::dtra::hpac::server::IncidentModelServerFactory
{
}; // NWIServerFactory
}; // server
}; // nwi
}; // models
}; // hpac
}; // dtra
}; // mil

#endif

```

3.13 NUCLEAR FACILITY

3.13.1 Nfac.idl

```

#ifndef __Nfac__
#define __Nfac__
//-----
//      NAME:      Nfac.idl
//-----
#include "server.idl"
#include "files.idl"
#include "project.idl"

```

```

#include "analyst.idl"

//-----
//      MODULE:          mil.dtra.hpac.models.nfac.server
//-----
module mil { module dtra { module hpac { module models { module nfac {
//****************************************************************************
* Definition of the Nfac incident model server
*****/
module server
{
    typedef mil::dtra::hpac::models::nfac::server::analyst::AnalystModelT
        AnalystModelT;

    typedef mil::dtra::hpac::server::files::FileReferenceT
        FileReferenceT;

    typedef mil::dtra::hpac::server::IncidentModelServer
        IncidentModelServer;

    typedef mil::dtra::hpac::server::IncidentModelServerFactory
        IncidentModelServerFactory;

    typedef mil::dtra::hpac::server::project::TimeT
        TimeT;

//-----
//      TYPE:          ReproInventoryT
//****************************************************************************
* Reprocessing inventory entry.
*****/
struct ReproInventoryT
{
    double           fAge; // (days)
    FileReferenceT   fFile;
    double           fFraction;
    double           fPower; // (Mw)
}; // ReproInventoryT

//-----
//      TYPE:          ReproInventoryListT
//****************************************************************************
* ****
*****/
typedef sequence<ReproInventoryT>
    ReproInventoryListT;

//-----
//      TYPE:          FacilityT
*****/

```

```

* Facility data as read from the legacy 3.2 <em>{lwr,rch,rpr}_xx.dat</em>
* files.
*****/
struct FacilityT
{
    long          fBuildingArea;
    long          fConstructionDate;
    double        fCoord[ 2 ];
    string        fCountryCode;
    double        fDesignLeakRate; // (Throughput for repro facilities)
    long          fDesignPressure;
    char          fIceBed;
    string        fInventoryFilename;
    string        fMet1HrFilename;
    string        fMet3HrFilename;
    string        fName;
    char          fOT;
    long          fPower; // (MWT)
    long          fStackHeight; // (ft)
    short         fSubType;
    string        fTypeName;
}; // FacilityT

//-----
//      TYPE:           FacilityDefT
//*****
* Complete facility definition, including any user customizations.
*****/
struct FacilityDefT
{
    FileReferenceT   fCustomInventory;
    boolean          fCustomInventoryFlag;
    double          fCustomLocation[ 3 ];
    boolean          fCustomLocationFlag;
    ReproInventoryListT fCustomReproInventory;

    FacilityT        fFacility;
}; // FacilityDefT

//-----
//      Model Type Constants
//-----
/***
 * Model type constant for a Moderate Accident
 */
const long        MODEL_MODERATE = 0;

/***
 * Model type constant for a Sever Accident
 */

```

```

const long      MODEL_SEVERE = 1;

           /**
           * Model type constant for a Technical Analysis
           */
const long      MODEL_ANALYST = 2;

const long      MODEL_LAST = 2;

//-----
//    TYPE:          ModelDefT
// ****
// ****
union ModelDefT
{
    switch( long )
    {
        case MODEL_ANALYST:
            AnalystModelT      fAnalyst;

        case MODEL_MODERATE:
            short              fModerate;

        case MODEL_SEVERE:
            short              fSevere;
    }; // ModelDefT

//-----
//    TYPE:          ModelTimesT
// ****
// ****
* Times.
// ****
struct ModelTimesT
{
    TimeT          fStartOfDecay;
    TimeT          fReleaseToContainment;
    TimeT          fReleaseToEnvironment;
    TimeT          fEndOfRelease;
    TimeT          fEndOfExposure;
}; // ModelTimesT

//-----
//    Cloud Dose Constants
//-----
const long      CD_AIR_SUBMERSION = 0;
const long      CD_CLOUD_SHINE = 1;

//-----
//    TYPE:          OptionsT
// ****

```

```

* Calculation options.
*****/
struct OptionsT
{
    /**
     * either <tt>CD_AIR_SUBMERSION</tt> or
     * <tt>CD_CLOUD_SHINE</tt>
     */
    long fCloudDose;

    /**
     * m^2
     */
    double fExhaustArea;

    /**
     * in m/s
     */
    double fExhaustVerticalVelocity;

    /**
     * 3.2 was miles, perhaps nautical miles
     */
    double fRadius;

    /**
     * in m
     */
    double fReleaseHeight;

    /**
     * above ambient (20C), in C
     */
    double fTemperatureExcess;
}; // OptionsT

//-----
//      TYPE:          NfacIncidentT
//*****
* Complete definition of the NFAC incident.
*****/
struct NfacIncidentT
{
    FacilityDefT   fFacilityDef;
    ModelDefT      fModelDef;
    ModelTimesT    fModelTimes;
    OptionsT       fOptions;
    string          fVersion;
}; // NfacIncidentT

//-----

```

```

//      INTERFACE:      NfacServer
//****************************************************************************
*****interface NfacServer : IncidentModelServer
{
};

//-----
//      Nfac Factory Service Name
//-----
const string          NFAC_FACTORY_SERVICE_NAME = "NfacFactory";

//-----
//      INTERFACE:      NfacServerFactory
//****************************************************************************
*****interface NfacServerFactory : IncidentModelServerFactory
{
};

};

};

};

};

};

#endif

```

3.13.2 analyst.idl

```

#ifndef __analyst__
#define __analyst__
//-
//      NAME:          analyst.idl
//-
#include "server.idl"
#include "files.idl"

//-
//      MODULE:        mil.dtra.hpac.models.nfac.server.analyst
//-
module mil { module dtra { module hpac { module models { module nfac {
    module server {
/******
* Analyst model stuff
******/
module analyst
{
    typedef mil::dtra::hpac::server::files::FileReferenceT
        FileReferenceT;

    typedef mil::dtra::hpac::server::project::TimeT

```

```

TimeT;

//-----
// Activity Units Constants
//-----
const string          ACTIVITY_UNITS_BQ = "Becquerels";
const string          ACTIVITY_UNITS_CI = "Curies";
const string          ACTIVITY_UNITS_GRAMS = "grams";
const string          ACTIVITY_UNITS_OUNCES = "ounces";
const string          ACTIVITY_UNITS_LBS = "pounds";

//-----
// VolumeMass Units Constants
//-----
const string          VM_UNITS_CC = "cc";
const string          VM_UNITS_FT3 = "ft^3";
const string          VM_UNITS_LITERS = "liters";
const string          VM_UNITS_GRAMS = "grams";

//-----
// Time Units Constants
//-----
const string          TIME_UNITS_SEC = "sec";
const string          TIME_UNITS_MIN = "min";
const string          TIME_UNITS_HR = "hr";

//-----
// TYPE:          ConcentrationUnitsT
// ****
struct ConcentrationUnitsT
{
    string          fActivityUnits;
    string          fVolumeMassUnits;
}; // ConcentrationUnitsT

//-----
// TYPE:          ReleaseRateT
// ****
struct ReleaseRateT
{
    double         fValue;
    string          fVolumeMassUnits;
    string          fTimeUnits;
}; // ReleaseRateT

```

```

//-----
//      TYPE:          ReleaseUnitsT
//*****
*****struct ReleaseUnitsT
{
    string           fActivityUnits;
    string           fTimeUnits;
}; // ReleaseUnitsT

//-----
//      TYPE:          IsotopeValueT
//*****
*** Simple name, value tuple.
*****struct IsotopeValueT
{
    string           fName;
    double           fValue;
}; // IsotopeValueT

//-----
//      TYPE:          IsotopeValueListT
//*****
*****typedef sequence<IsotopeValueT>
                    IsotopeValueListT;

//-----
//      TYPE:          IsotopicConcentrationsT
//*****
*****struct IsotopicConcentrationsT
{
    ReleaseRateT     fReleaseRate;
    ConcentrationUnitsT fUnits;
    IsotopeValueListT   fValues;
}; // IsotopicConcentrationsT

//-----
//      TYPE:          IsotopicReleaseRatesT
//*****
*****struct IsotopicReleaseRatesT
{
    ReleaseUnitsT     fUnits;
    IsotopeValueListT   fValues;
}; // IsotopicReleaseRatesT

```

```

//-----
//      Percent Inventory Release Factor Array Index Constants      -
//-----

const long          PIR_NOBEL_GASES = 0;
const long          PIR_ALKALI_METALS = 1;
const long          PIR_ALKALINE_EARTHS = 2;
const long          PIR_HALOGENS = 3;
const long          PIR_CHALCOGENS = 4;
const long          PIR_PLATINOIDS = 5;
const long          PIR_EARLY_TRANSITION_ELEMENTS = 6;
const long          PIR_TETRAVALENTS = 7;
const long          PIR_TRIVALENTS = 8;
const long          PIR_URANIUM = 9;
const long          PIR_MORE_VOLATILE_MAIN_GROUP = 10;
const long          PIR_LESS_VOLATILE_MAIN_GROUP = 11;

const long          PIR_GROUP_COUNT = 12;

//-----
//      TYPE:          PercentInventoryReleaseT      -
//***** ****
***** /
```

PercentInventoryReleaseT
{
 double fDuration; // hours
 double fPercentages[PIR_GROUP_COUNT];
}; // PercentInventoryReleaseT

```

//-----
//      TYPE:          PercentInventoryReleaseListT      -
//***** ****
***** /
```

typedef sequence<PercentInventoryReleaseT>
PercentInventoryReleaseListT;

```

//-----
//      TYPE:          PercentInventoryT      -
//***** ****
***** /
```

PercentInventoryT
{
 double fOperatingPower; // MwT
 PercentInventoryReleaseListT
 fReleaseList;
}; // PercentInventoryT

```

//-----
//      TYPE:          AnalystMixT      -
```

```

/***** */
***** /
struct AnalystMixT
{
    double          fGrossReleaseRate;
    double          fPercentages[ PIR_GROUP_COUNT ];
    ReleaseUnitsT   fUnits;
}; // AnalystMixT

//-----
// Coolant Concentration Constants
//-----
const long      COOLCONC_GAP = 0;
const long      COOLCONC_IN_VESSEL = 1;
const long      COOLCONC_TYPICAL = 2;
const long      COOLCONC_NON_NOBLES = 3;
const long      COOLCONC_FUEL = 4;
const long      COOLCONC_COUNT = 5;

//-----
// Core Condition Constants
//-----
const long      CORECOND_GAP = 0;
const long      CORECOND_IN_VESSEL_SEVERE = 1;
const long      CORECOND_VESSEL_MELT_THROUGH = 2;
const long      CORECOND_IN_CAVITY_TUBE_RUPTURE = 3;
const long      CORECOND_REACTOR_CAVITY_FAILURE = 4;
const long      CORECOND_FUEL_MELT_CONTAINED = 5;
const long      CORECOND_PRESSURE_TUBE_MELT_THROUGH = 6;
const long      CORECOND_COUNT = 7;

//-----
// Monitor Location Constants
//-----
const long      MONLOC_PWR = 0;
const long      MONLOC_BWR1_WET = 1;
const long      MONLOC_BWR2_WET = 2;
const long      MONLOC_BWR3_WET = 3;
const long      MONLOC_BWR1_DRY = 4;
const long      MONLOC_BWR2_DRY = 5;
const long      MONLOC_BWR3_DRY = 6;
const long      MONLOC_VVER1 = 7;
const long      MONLOC_VVER2 = 8;
const long      MONLOC_VVER3 = 9;
const long      MONLOC_COUNT = 10;

//-----
// TYPE:          ContainmentMonitorReadingT
// **** */

```

```
*****
struct ContainmentMonitorReadingT
{
    boolean          fFilteredReleasePath;
    double           fLeakRate; // percentage
    long             fMonitorLocation;
    double           fMonitorReading;
    double           fOperatingPower; // MWe
    boolean          fSpraysOn;
}; // ContainmentMonitorReadingT

//-----
//      Plant Condition Type Constants
//-----
const long          PC_LARGE_CONTAINMENT_LEAKAGE = 0;
const long          PC_CONTAINMENT_LEAKAGE = 1;
const long          PC_CONFINEMENT_LEAKAGE = 2;
const long          PC_SUBATMOSPHERIC_CONFINEMENT_LEAKAGE = 3;
const long          PC_STEAM_TUBE_RUPTURE = 4;
const long          PC_ICE_CONDENSER_CONTAINMENT_LEAKAGE = 5;
const long          PC_DRY_WELL_LEAKAGE = 6;
const long          PC_WET_WELL_LEAKAGE = 7;
const long          PC_CONTAINMENT_BYPASS = 8;
const long          PC_CONFINEMENT_BYPASS = 9;
const long          PC_CRITICAL_POWER_EXCURSION = 10;
const long          PC_SODIUM_WATER_REACTION = 11;
const long          PC_REPROCESSING_FACILITY = 12;
const long          PC_COUNT = 13;

//-----
//      TYPE:          PlantConditionsT
//*****
union PlantConditionsT
switch( long )
{
    case PC_LARGE_CONTAINMENT_LEAKAGE:
        long          fDummy1;

    case PC_CONTAINMENT_LEAKAGE:
        long          fDummy2;

    case PC_CONFINEMENT_LEAKAGE:
        long          fDummy3;

    case PC_SUBATMOSPHERIC_CONFINEMENT_LEAKAGE:
        long          fDummy4;

    case PC_STEAM_TUBE_RUPTURE:
        long          fDummy5;
}
```

```

case PC_ICE_CONDENSER_CONTAINMENT_LEAKAGE:
    long          fDummy6;

case PC_DRY_WELL_LEAKAGE:
    long          fDummy7;

case PC_WET_WELL_LEAKAGE:
    long          fDummy8;

case PC_CONTAINMENT_BYPASS:
    long          fDummy9;

case PC_CONFINEMENT_BYPASS:
    long          fDummy10;

case PC_CRITICAL_POWER_EXCURSION:
    long          fDummy11;

case PC_SODIUM_WATER_REACTION:
    long          fDummy12;

case PC_REPROCESSING_FACILITY:
    long          fDummy13;
};

// -----
// Fuel Condition Constants
// -----
const long      FUELCOND_ZIRCALLOY_FIRE = 0;
const long      FUELCOND_FUEL_CLADDING_FAILURE = 1;

// -----
// TYPE:           SpentFuelT
// ****
struct SpentFuelT
{
    long          fBatchCount;
    boolean       fFilteredReleasePath;
    long          fFuelCondition;
    double        fLeakRate; // percentage
    double        fOperatingPower; // MWe
    boolean       fSpraysOn;
    TimeT         fTimeLastBatchInPool;
};

// -----
// TYPE:           ExternalRadFileT
// ****

```

```

struct ExternalRadFileT
{
    FileReferenceT      fRadFile;
    long               fReleaseCount;
    double              fTotalDuration;
}; // ExternalRadFileT

//-----
// Analyst Model Constants
//-----
const long          ANALYST_ISOTOPIC_RELEASE_RATES = 0;
const long          ANALYST_ISOTOPIC_CONCENTRATIONS = 1;
const long          ANALYST_PERCENT_INVENTORY = 2;
const long          ANALYST_ANALYST_MIX = 3;
const long          ANALYST_PLANT_CONDITIONS = 4;
const long          ANALYST_CONTAINMENT_MONITOR_READING = 5;
const long          ANALYST_SPENT_FUEL = 6;
const long          ANALYST_EXTERNAL_RAD_FILE = 7;

//-----
// TYPE:           AnalystModelT
// ****
union AnalystModelT
switch( long )
{
    case ANALYST_ANALYST_MIX:
        AnalystMixT          fAnalystMix;

    case ANALYST_EXTERNAL_RAD_FILE:
        ExternalRadFileT     fExternalRadFile;

    case ANALYST_ISOTOPIC_RELEASE_RATES:
        IsotopicReleaseRatesT fIsotopicReleaseRates;

    case ANALYST_ISOTOPIC_CONCENTRATIONS:
        IsotopicConcentrationsT fIsotopicConcentrations;

    case ANALYST_PERCENT_INVENTORY:
        PercentInventoryT     fPercentInventory;

    case ANALYST_PLANT_CONDITIONS:
        PlantConditionsT      fPlantConditions;

    case ANALYST_SPENT_FUEL:
        SpentFuelT             fSpentFuel;
}; // AnalystModelT
}; // analyst

}; }; }; }; }; };

```

```
#endif
```

3.14 NUCLEAR WEAPON

3.14.1 Nwpn.idl

```
#ifndef __Nwpn__
#define __Nwpn__
//-
//      NAME:          Nwpn.idl
//-
#include "server.idl"

//-
//      MODULE:        mil.dtra.hpac.models.nwpn
//-
module mil { module dtra { module hpac { module models { module nwpn {

module server
{
//-
//      Constants
//-

//-
//      TYPE:          NwpnIncidentT
//


struct NwpnIncidentT
{
    string           modelVersion;
    string           hostURL;
    string           subdir;
    string           database;
    boolean          delfic;
}; // NwpnIncidentT

//-
//      INTERFACE:     NwpnServer
//


interface NwpnServer : mil::dtra::hpac::server::IncidentModelServer
{
//-
//      METHOD:        initIncident()
//-
```

```

void
nwpnInitIncident(
    inout mil::dtra::hpac::server::IncidentT           incident,
    inout any                                         model_incident
)
raises ( mil::dtra::hpac::server::ModelException );
}; // NwpnServer

//-----
//      Nwpn Factory Service Name
//-----
const string          NWPN_FACTORY_SERVICE_NAME = "NwpnFactory";

//-----
//      INTERFACE:      NwpnServerFactory
//-----

interface NwpnServerFactory : mil::dtra::hpac::server::IncidentModelServerFactory
{
}; // NwpnServerFactory
}; // server
}; // nwpn
}; // models
}; // hpac
}; // dtra
}; // mil

#endif

```

3.15 RADIOLOGICAL WEAPON

3.15.1 rwpn.idl

```

#ifndef __RWPN__
#define __RWPN__
//-----
//      NAME:          RWPN.idl
//-----
#include "server.idl"

#pragma prefix "mil.dtra.hpac"

//-----
//      MODULE:        rwpn
***** /*****
module mil { module dtra { module hpac { module models {module rwpn {
module server

```

```
{
//-----
//    Constants
//-----

// source choices
const long      LARGE_DEVICE_SOURCE = 0;
const long      MEDIUM_DEVICE_SOURCE = 1;
const long      SMALL_DEVICE_SOURCE = 2;
const long      USER_DEFINED_DEVICE_SOURCE = 3;
const long      ERAD_FILE_SOURCE = 4;
typedef float   oneDimFloatArray[90];
typedef long    oneDimLongArray[90];
typedef boolean oneDimBoolArray[90];

//-----
//    TYPE:          ERADPuffsT
//*****
//*****
struct ERADPuffsT
{
    // data specific to ERADPuffsT
    float        fERADMass;
    oneDimBoolArray fValidPuff;
    oneDimLongArray fMaterialSubGroup;
    oneDimFloatArray fMassFraction;
    oneDimFloatArray fX;
    oneDimFloatArray fY;
    oneDimFloatArray fZ;
    oneDimFloatArray fSigmaX;
    oneDimFloatArray fSigmaY;
    oneDimFloatArray fSigmaZ;
}; // ERADPuffsT

//-----
//    TYPE:          RWPNIIncidentT
//*****
//*****
// RWPNIIncidentT is the server RWPNI part of an overall incident
// the rest of an overall incident is in IncidentT.
// RWPNIIncidentT consists of data common to all sources
struct RWPNIIncidentT
{
    // data common to all sources

    string        fModelVersion; // HPAC rwpn module version

    // specify which 1 of 5 sources
}
```

```

long          fSourceType;
// one of:
//LARGE_DEVICE_SOURCE = 0;
//MEDIUM_DEVICE_SOURCE = 1;
//SMALL_DEVICE_SOURCE = 2;
//USER_DEFINED_DEVICE_SOURCE = 3;
//ERAD_FILE_SOURCE = 4;

float         fCalcRadius;      // km
long          fCloudShine;
float         fExposureTime;   // hr

float         fRadMatlMass;
float         fHEMass;
string        fTypeRadMatl;
string        fMatlForm;
ERADPuffsT    fEradPuffs;

};

} // RWPNIIncidentT

//-----
// INTERFACE:      RWPNServer
// ****
interface RWPNServer : mil::dtra::hpac::server::IncidentModelServer
{
};

} // RWPNServer

//-----
// RWPN Factory Service Name
//-----
const string      RWPN_FACTORY_SERVICE_NAME = "RWPNFactory";

//-----
// INTERFACE:      RWPNServerFactory
// ****
interface RWPNServerFactory : mil::dtra::hpac::server::IncidentModelServerFactory
{
};

} // RWPNServerFactory
}; // server
}; // rwpn
}; // models
}; // hpac
}; // dtra
}; // mil

#endif

```

3.16 SMOKE MUNITION

CHAPTER 4

IHPACServer

IDL modules and interfaces defining the servers comprising the IHPACServer are specified in a separate document [14]. The IDL files are listed below. In addition to the examples provided in the Interface Description Document [14], an example client program is provided in Appendix C.

Although access to all the detailed inputs for the T&D engine are not provided in IHPACServer, it presents a higher level, more easy to program abstraction of HPAC services. Extensions of the common `IIncident` interface are provided for the following subset of HPAC source models:

- Chemical-biological weapon,
- Missile intercept,
- Nuclear weapon, and
- Radiological weapon.

4.1 chembioweapon.idl

```
/*
** @FileName: chembioweapon.idl
**
** Copyright (c) 2001, Science Applications International Corporation,
** all rights reserved.
**
** @Purpose: This file contains the IDL specification the IHPACServer
** corba interface.
**
*/
#ifndef _chembioweapon_
#define _chembioweapon_

#include "incident.idl"

module mil { module dtra { module hpac { module ihpacserver {
module models {
module chembioweapon
{
    // Typdefs
    typedef mil::dtra::hpac::ihpacserver::incident::IIncident IIncident;
    typedef sequence<string> StringList;
```

```

// Constants
const string CHEMBIOWEAPON_MODEL = "Chemical/Biological Weapon";

// Forward declarations
interface IChemBioWeapon;

// Interfaces

// IChemBioWeapon interface
interface IChemBioWeapon : IIIncident
{
    long getAgent();
    double getAgentPurity();
    StringList getAvailAgents();
    StringList getAvailDeliverySystems();
    StringList getAvailMunitions();
    long getDeliverySystem();
    double getEfficiency();
    double getFallAngle();
    double getHeading();
    double getHOB();
    double getHorzUncertainty();
    double getInitialSize();
    double getLength();
    double getLiquidFraction();
    double getMass();
    double getMMD();
    string getModelVersion();
    long getMunition();
    double getNumber();
    double getReleaseMass();
    long getSeed();
    double getSigmaD();
    double getSpeed();
    double getSpread();
    double getVaporFraction();
    double getVertUncertainty();
    void setAgent(in long agent);
    void setAgentPurity(in double purity);
    void setAllData(in long munition, in long delivery, in long agent);
    void setDeliverySystem(in long delivery);
    void setEfficiency(in double efficiency);
    void setFallAngle(in double angle);
    void setHeading(in double heading);
    void setHOB(in double hob);
    void setHorzUncertainty(in double uncertainty);
    void setInitialSize(in double size);
    void setLength(in double length);
    void setLiquidFraction(in double fraction);
    void setMass(in double mass);
    void setMMD(in double mmd);
    void setMunition(in long munition);
}

```

```

    void setNumber(in double number);
    void setReleaseMass(in double mass);
    void setSeed(in long seed);
    void setSigmaD(in double sigmaD);
    void setSpeed(in double speed);
    void setSpread(in double spread);
    void setVaporFraction(in double fraction);
    void setVertUncertainty(in double uncertainty);
        string toAgentString(in long agent);
        string toDeliverySystemString(in long delivery);
        string toMunitionString(in long munition);
        long valueOfAgent(in string agent);
        long valueOfDeliverySystem(in string delivery);
        long valueOfMunition(in string munition);
    };
};

};

#endif

```

4.2 dispersion.idl

```

/*
** @FileName: dispersion.idl
**
** Copyright (c) 2001, Science Applications International Corporation,
** all rights reserved.
**
** @Purpose: This file contains the IDL specification the IHPACServer
** corba interface.
**
*/
#ifndef _dispersion_
#define _dispersion_

#include "parameters.idl"

module mil { module dtra { module hpac { module ihpacserver {
module dispersion
{
    // Typedefs
    typedef mil::dtra::hpac::ihpacserver::parameters::TimeS TimeS;
    typedef mil::dtra::hpac::ihpacserver::parameters::IIntervals IIntervals;
    typedef mil::dtra::hpac::ihpacserver::parameters::ITemporalDomain ITemporalDomain;

    // Constants
    const long POLL_SUCCESS = 1;
    const long POLL_ERROR   = 2;
    const long POLL_RUNNING = 3;
    const long POLL_IDLE    = 4;

    const long MESSAGE_INFO  = 1;
}
}
}
}

```

```

const long MESSAGE_ERROR = 2;
const long MESSAGE_REPLY = 3;

const long REPLY_AFFIRMATIVE = 1;
const long REPLY_NEGATIVE = 2;

// Structures
struct MessageS
{
    long fMessageType;
    string fMessage;
};

typedef sequence<MessageS> MessageList;

// Exceptions
exception DispersionException
{
    string fMessage;
};

// Forward declarations
interface IDispersionServer;

// Interfaces

// IDispersionServer interface
interface IDispersionServer
{
    void cancelCalculation() raises(DispersionException);
    TimeS getCurrentRunTime();
    IIntervals getIntervals();
    ITemporalDomain getTemporalDomain();
    long poll(out MessageList messages) raises(DispersionException);
    void reply(in long reply) raises(DispersionException);
    void resumeCalculation() raises(DispersionException);
    boolean resumeSynchCalculation(out MessageList messages)
        raises(DispersionException);
    void startCalculation() raises(DispersionException);
    boolean startSynchCalculation(out MessageList messages)
        raises(DispersionException);
};

};

};

};

};

};

#endif

/*
** @FileName: incident.idl
**
** Copyright (c) 2001, Science Applications International Corporation,

```

4.3 incident.idl

```

/*
** @FileName: incident.idl
**
** Copyright (c) 2001, Science Applications International Corporation,

```

```

** all rights reserved.
**
** @Purpose: This file contains the IDL specification the IHPACServer
** corba interface.
**
*/
#ifndef _incident_
#define _incident_

#include "parameters.idl"
#include "release.idl"

module mil { module dtra { module hpac { module ihpacserver {
module incident
{
    // Typedefs
    typedef mil::dtra::hpac::ihpacserver::release::IReleaseMgr IReleaseMgr;
    typedef mil::dtra::hpac::ihpacserver::parameters::TimeS TimeS;
    typedef mil::dtra::hpac::ihpacserver::parameters::LocationS LocationS;
    typedef sequence<string> StringList;

    // Exceptions
    exception IncidentException
    {
        string fMessage;
    };

    // Forward declarations
    interface IIIncidentMgr;
    interface IIIncident;

    // Interfaces

    // IIIncidentMgr interface
    interface IIIncidentMgr
    {
        IIIncident createIncident(in string model);
        IIIncident findIncident(in string name);
        StringList getAvailIncidentModels();
        StringList getAvailIncidents(in string file);
        IIIncident getIncident(in long index);
        long getIncidentCount();
        StringList getIncidentList();
        IIIncident importIncident(in string file, in string name);
        long indexOfIncident(in string name);
        void removeIncident(in long index);
    };

    // IIIncident interface
    interface IIIncident
    {
        Locations getLocation();
        any getModel();
    };
}
}
}
}

```

```

        string getModelType();
        string getName();
        IReleaseMgr getReleaseMgr();
        TimeS getStartTime();
        void setLocation(in LocationsS location);
        void setModel(in any model);
        void setName(in string name);
        void setStartTime(in TimeS time);
        void update() raises(IncidentException);
    };

};

};

};

#endif

```

4.4 missileintercept.idl

```

/*
** @FileName: missileintercept.idl
**
** Copyright (c) 2001, Science Applications International Corporation,
** all rights reserved.
**
** @Purpose: This file contains the IDL specification the IHPACServer
** corba interface.
**
*/
#ifndef _missileintercept_
#define _missileintercept_

#include "incident.idl"
#include "parameters.idl"

module mil { module dtra { module hpac { module ihpacserver {
module models {
module missileintercept
{
    // Typedefs
    typedef mil::dtra::hpac::ihpacserver::incident::IIncident IIncident;
    typedef mil::dtra::hpac::ihpacserver::parameters::Locations LocationS;
    typedef sequence<string> StringList;

    // Constants
    const string MISSILEINTERCEPT_MODEL = "Missile Intercept";
    const long MODE_PLAN = 1;
    const long MODE_LIVE = 2;

    // Forward declarations
    interface IMissileIntercept;

    // Interfaces

```

```

// IMissileIntercept interface
interface IMissileIntercept : IIIncident
{
    StringList getAvailDamageLevels();
    StringList getAvailMaterials(in long pair);
    StringList getAvailPairs();

    long getDamageLevel();
    double getLiveInterceptAzimuth();
    double getLiveInterceptFPA();
    double getLiveInterceptSpeed();
    double getLiveThreatAzimuth();
    double getLiveThreatFPA();
    double getLiveThreatSpeed();
    long getMaterial();
    long getMode();
        string getModelVersion();
    long getPair();
    LocationsS getPlanAimpointLocation();
    LocationsS getPlanInterceptLaunchLocation();
    LocationsS getPlanThreatLaunchLocation();
    double getPlanInterceptAltitude();
    void setDamageLevel(in long damage);
        void setLiveDefaultData(in LocationsS location);
    void setLiveInterceptAzimuth(in double azimuth);
    void setLiveInterceptFPA(in double fpa);
    void setLiveInterceptSpeed(in double speed);
    void setLiveThreatAzimuth(in double azimuth);
    void setLiveThreatFPA(in double fpa);
    void setLiveThreatSpeed(in double speed);
    void setMaterialPair(in long material, in long pair);
    void setMode(in long mode);
    void setPlanAimpointLocation(in LocationsS location);
        void setPlanDefaultData(in LocationsS location);
    void setPlanInterceptLaunchLocation(in LocationsS location);
    void setPlanThreatLaunchLocation(in LocationsS location);
    void setPlanInterceptAltitude(in double altitude);
        string toDamageLevelString(in long damage);
        string toMaterialString(in long material, in long pair);
        string toPairString(in long pair);
        long valueOfDamageLevel(in string damage);
        long valueOfMaterial(in string material, in string pair);
        long valueOfPair(in string pair);
};

};

};

};

};

};

#endif

```

4.5 nuclearweapon.idl

```

/*
** @fileName: nuclearweapon.idl

```

```

**
** Copyright (c) 2001, Science Applications International Corporation,
** all rights reserved.
**
** @Purpose: This file contains the IDL specification the IHPACServer
** corba interface.
**
*/
#ifndef _nuclearweapon_
#define _nuclearweapon_

#include "incident.idl"

module mil { module dtra { module hpac { module ihpacserver {
module models {
module nuclearweapon
{
    // Typedefs
    typedef mil::dtra::hpac::ihpacserver::parameters::TimeS TimeS;
    typedef mil::dtra::hpac::ihpacserver::parameters::LocationS LocationS;
    typedef mil::dtra::hpac::ihpacserver::incident::IIncident IIncident;

    // Constants
    const string NUCLEARWEAPON_MODEL = "Nuclear Weapon";

    // Forward declarations
    interface INuclearWeapon;

    // Interfaces

    // INuclearWeapon interface
    interface INuclearWeapon : IIncident
    {
        string getDatabase();
        boolean getDelficRise();
        string getHostURL();
        string getModelVersion();
        double getStrikeCEP(in long index);
        double getStrikeFF(in long index);
        double getStrikeHOB(in long index);
        Locations getStrikeLocation(in long index);
        long getStrikePA(in long index);
        TimeS getStrikeTime(in long index);
        long getStrikeType(in long index);
        double getStrikeYield(in long index);
        string getSubdirectory();
        void setDatabase(in string database);
        void setDelficRise(in boolean delfic);
        void setHostURL(in string url);
        void setStrikeCEP(in long index, in double cep);
        void setStrikeFF(in long index, in double ff);
        void setStrikeHOB(in long index, in double hob);
        void setStrikeLocation(in long index, in Locations location);
    }
}
}
}
}

```

```

    void setStrikePA(in long index, in long pa);
    void setStrikeTime(in long index, in TimeS time);
    void setStrikeType(in long index, in long type);
    void setStrikeYield(in long index, in double yield);
    void setSubdirectory(in string subdirectory);
}

};

};

};

};

#endif

```

4.6 parameters.idl

```

/*
** @FileName: parameters.idl
**
** Copyright (c) 2001, Science Applications International Corporation,
** all rights reserved.
**
** @Purpose: This file contains the IDL specification the IHPACServer
** corba interface.
**
*/
#ifndef _parameters_
#define _parameters_

module mil { module dtra { module hpac { module ihpacserver {
module parameters
{
    // Typedefs
    typedef sequence<string> StringList;

    // Constants
    const long SCIPUFF_METHOD_DYNAMIC = 1;
    const long SCIPUFF_METHOD_DENSE = 2;
    const long SCIPUFF_METHOD_STATIC = 4;
    const long SCIPUFF_METHOD_MULTICOMP = 8;

    const long SCIPUFF_MODE_FAST = 1;
    const long SCIPUFF_MODE_HAZARD = 2;
    const long SCIPUFF_MODE_DUAL = 4;

    const long HPAC_MODE_OPERATIONAL = 1;
    const long HPAC_MODE_EXTENDED = 2;
    const long HPAC_MODE_ULTIMATE = 3;

    // Structures
    struct TimeS
    {
        long fYear;
        long fMonth;
        long fDay;
    }
}
}
}
}

```

```

    long fHour;
    long fMinute;
    long fSecond;
    long fMilliSecond;
};

struct LocationsS
{
    double fAltitude;
    double fLatitude;
    double fLongitude;
};

// Forward declarations
interface ILimits;
interface IWeather;
interface IOptions;
interface IIntervals;
interface IFlags;
interface ITemporalDomain;
interface ISpatialDomain;

// Interfaces

// ILimits interface
interface ILimits
{
    long getMaxGridCellsPerSurface();
    long getMaxMetHorzSize();
    long getMaxPuffs();
    void setMaxGridCellsPerSurface(in long cells);
    void setMaxMetHorzSize(in long size);
    void setMaxPuffs(in long puffs);
    void setHPACMode(in long mode);
};

// IWeather interface
interface IWeather
{
    double getFixedWindsDirection();
    double getFixedWindsSpeed();
    string getMetFile1();
    string getMetFile2();
    void setFixedWindsDirection(in double direction);
    void setFixedWindsSpeed(in double speed);
    void setMetFile1(in string file);
    void setMetFile2(in string file);
};

// IOptions interface
interface IOptions
{
    double getAdaptiveGridMinSize();

```

```

StringList getAvailSubstrates();
long getGridResolution();
double getPuffMinMass();
string getSamplerLocations();
double getSamplerMinOutputInterval();
long getSubstrate();
double getSurfaceDoseHeight();
double getTropoAvgEnergyDissipationRate();
double getTropoVertLengthScale();
double getTropoVertVelocityVariance();
double getTurbDiffusiveAvgTime();
double getTurbLightWindScale();
double getTurbLightWindValue();
long getTurbVertGridPointCount();
void setAdaptiveGridMinSize(in double size);
void setGridResolution(in long resolution);
void setPuffMinMass(in double mass);
void setSamplerLocations(in string locations);
void setSamplerMinOutputInterval(in double interval);
void setSubstrate(in long substrate);
void setSurfaceDoseHeight(in double height);
void setTropoAvgEnergyDissipationRate(in double rate);
void setTropoVertLengthScale(in double scale);
void setTropoVertVelocityVariance(in double variance);
void setTurbDiffusiveAvgTime(in double time);
void setTurbLightWindScale(in double scale);
void setTurbLightWindValue(in double value);
void setTurbVertGridPointCount(in long count);
string toSubstrateString(in long substrate);
long valueOfSubstrate(in string substrate);
};

// IIntervals interface
interface IIntervals
{
    double getMaxTimeStep();
    double getOutputInterval();
    void setMaxTimeStep(in double step);
    void setOutputInterval(in double interval);
};

// IFlags interface
interface IFlags
{
    string getAnalyst();
    StringList getAvailClassifications();
    long getClassification();
    string getDate();
    string getHPACVersion();
    string getProjectTitle();
    long getScipuffMethod();
    long getScipuffMode();
    void setAnalyst(in string anaylst);
}

```

```

void setClassification(in long classification);
void setDate(in string date);
void setHPACVersion(in string version);
void setProjectTitle(in string title);
void setScipuffMethod(in long scipuffMethod);
void setScipuffMode(in long scipuffMode);
string toClassificationString(in long classification);
long valueOfClassification(in string classification);
};

// ITemporalDomain interface
interface ITemporalDomain
{
    boolean getComputeDefaultFlag();
    double getDuration();
    TimeS getEndTime();
    TimeS getStartTime();
    void setComputeDefaultFlag(in boolean flag);
    void setDuration(in double time);
    void setEndTime(in TimeS time);
    void setStartTime(in TimeS time);
};

// ISpatialDomain interface
interface ISpatialDomain
{
    boolean getComputeDefaultFlag();
    double getHorizontalResolution();
    double getMaxAltitude();
    LocationsS getNorthEast();
    LocationsS getSouthWest();
    double getVerticalResolution();
    void setComputeDefaultFlag(in boolean flag);
    void setHorizontalResolution(in double resolution);
    void setMaxAltitude(in double altitude);
    void setNorthEast(in LocationsS location);
    void setSouthWest(in LocationsS location);
    void setVerticalResolution(in double resolution);
};

};

};

};

};

};

#endif

```

4.7 plot.idl

```

/*
** @FileName: plot.idl
**
** Copyright (c) 2001, Science Applications International Corporation,
** all rights reserved.
*/

```

```

** @Purpose: This file contains the IDL specification the IHPACServer
** corba interface.
**
*/
#ifndef _plot_
#define _plot_

#include "parameters.idl"

module mil { module dtra { module hpac { module ihpacserver {
module plot
{
    // Typedefs
    typedef mil::dtra::hpac::ihpacserver::parameters::LocationS LocationS;
    typedef mil::dtra::hpac::ihpacserver::parameters::TimeS TimeS;
    typedef sequence<string> StringList;
    typedef sequence<TimeS> TimeList;
    typedef sequence<LocationS> PointList;

    // Constants
    const long EXPORT_TYPE_AVG = 0;
    const long EXPORT_TYPE_ARC = 1;
    const long EXPORT_TYPE_CTS = 2;
    const long EXPORT_TYPE_OVL = 3;
    const long EXPORT_TYPE_TAB = 4;
    const long EXPORT_TYPE_TXT = 5;

    const long CONTOUR_MODE_HPAC = 0;
    const long CONTOUR_MODE_UNIFORM = 1;
    const long CONTOUR_MODE_CUSTOM = 2;

    const long CONTOUR_SPACING_LOG = 0;
    const long CONTOUR_SPACING_LINEAR = 1;
    const long CONTOUR_SPACING_AUTO = 2;

    const long POPULATION_AREA_MODE_OFF = 0;
    const long POPULATION_AREA_MODE_POP_EXPECTED = 1;
    const long POPULATION_AREA_MODE_POP_CONTOUR = 2;
    const long POPULATION_AREA_MODE_AREA_EXPECTED = 3;
    const long POPULATION_AREA_MODE_AREA_CONTOUR = 4;

    // Structures
    struct Contours
    {
        string fUnits;
        string fLabel;
        double fValue;
        PointList fPoints;
    };
    typedef sequence<Contours> ContourList;

    struct ContourValues
    {

```

```

        long fColor;
        string fLabel;
        double fLevel;
    };
typedef sequence<ContourValueS> ContourValueList;

struct CategoryDataS
{
    LocationsS fSouthWest;
    LocationsS fNorthEast;
    double fRisk;
    long fZLevels;
};

// Exceptions
exception PlotException
{
    string fMessage;
};

// Forward declarations
interface IPotMgr;
interface IPot;

// Interfaces

// IPotMgr interface
interface IPotMgr
{
    IPot createPlot();
    /* Deprecated */ IPot exportPlot();
    IPot findPlot(in string name);
    IPot getPlot(in long index);
    long getPlotCount();
    StringList getPlotList();
    long indexOfPlot(in string name);
    void removePlot(in long index);
};

// IPot interface
interface IPot
{
    void compute() raises(PlotException);
    void export(in string file, in long type) raises(PlotException);
    StringList getAvailCategories();
    StringList getAvailChoices();
    StringList getAvailContourColors();
    StringList getAvailContourColorSchemes();
    StringList getAvailKinds(in long option, in long choice);
    StringList getAvailOptions();
    TimeList getAvailTimes(in long option, in long choice);
    StringList getAvailTypes();
    long getCategory();
};

```

```

CategoryDataS getCategoryData();
long getChoice();
long getContourColorScheme();
double getContourCustomScale();
string getContourCustomUnits();
double getContourIncrement();
double getContourMaxValue();
double getContourMinValue();
long getContourMode();
ContourList getContours();
long getContourSpacing();
double getContourUniformScale();
string getContourUniformUnits();
ContourValueList getContourValues();
long getKind();
string getName();
long getNumContours();
longgetOption();
long getPopulationAreaMode();
long getTime();
long getType();
double getUserTime();
/* Deprecated */ void setCategory(in long category);
void setCategoryData(in CategoryDataS data);
/* Deprecated */ void setChoice(in long choice);
void setContourColorScheme(in long scheme);
void setContourCustomScale(in double scale);
void setContourCustomUnits(in string units);
void setContourIncrement(in double increment);
void setContourMaxValue(in double value);
void setContourMinValue(in double value);
void setContourMode(in long mode);
void setContourSpacing(in long spacing);
void setContourUniformScale(in double scale);
void setContourUniformUnits(in string units);
void setContourValues(in ContourValueList contours);
void setDefaultPlot();
/* Deprecated */ void setKind(in long kind);
void setName(in string name);
void setNumContours(in long contours);
/* Deprecated */ void setOption(in long option);
void setPlotByTime(
    in long option,
    in long choice,
    in long kind,
    in long category,
    in long type,
    in long time
);
void setPlotByUserTime(
    in long option,
    in long choice,
    in long kind,

```

```

        in long category,
        in long type,
        in double time
    );
void setPopulationAreaMode(in long mode);
/* Deprecated */
void setTime(in long time, in long option, in long choice);
/* Deprecated */
void setType(in long type);
/* Deprecated */
void setUserTime(in double time, in long option, in long choice);
string toCategoryString(in long category);
string toChoiceString(in long choice);
string toContourColorSchemeString(in long scheme);
string toContourColorString(in long color);
string toKindString(in long kind);
string toOptionString(in long option);
TimeS toTime(in long time, in long option, in long choice);
string toTypeString(in long type);
long valueOfCategory(in string category);
long valueOfChoice(in string choice);
long valueOfContourColor(in string color);
long valueOfContourColorScheme(in string scheme);
long valueOfKind(in string kind, in string option, in string choice);
long valueOfOption(in string option);
long valueOfTime(in TimeS time, in string option, in string choice);
long valueType(in string type);
};

};

};

};

};

};

#endif

```

4.8 project.idl

```

/*
** @FileName: project.idl
**
** Copyright (c) 2001, Science Applications International Corporation,
** all rights reserved.
**
** @Purpose: This file contains the IDL specification the IHPACServer
** corba interface.
**
*/
#ifndef _project_
#define _project_

#include "parameters.idl"
#include "incident.idl"
#include "plot.idl"
#include "dispersion.idl"

```

```

module mil { module dtra { module hpac { module ihpacserver {
module project
{
// Typedefs
typedef mil::dtra::hpac::ihpacserver::parameters::ILimits ILimits;
typedef mil::dtra::hpac::ihpacserver::parameters::IWeather IWeather;
typedef mil::dtra::hpac::ihpacserver::parameters::IOptions IOptions;
typedef mil::dtra::hpac::ihpacserver::parameters::IIntervals IIntervals;
typedef mil::dtra::hpac::ihpacserver::parameters::IFlags IFlags;
typedef mil::dtra::hpac::ihpacserver::parameters::ITemporalDomain
    ITemporalDomain;
typedef mil::dtra::hpac::ihpacserver::parameters::ISpatialDomain
    ISpatialDomain;
typedef mil::dtra::hpac::ihpacserver::parameters::TimeS TimeS;
typedef mil::dtra::hpac::ihpacserver::incident::IIIncidentMgr IIIncidentMgr;
typedef mil::dtra::hpac::ihpacserver::plot::IPlotMgr IPPlotMgr;
typedef mil::dtra::hpac::ihpacserver::dispersion::IDispersionServer
    IDispersionServer;

// Exceptions
exception ProjectException
{
    string fMessage;
};

// Forward declarations
interface IProject;

// Interfaces

// IProject interface
interface IProject
{
    void close();
    void delete() raises(ProjectException);
    TimeS getCurrentRunTime();
    IDispersionServer getDispersionServer();
    IFlags getFlags();
    IIIncidentMgr getIncidentMgr();
    IIntervals getIntervals();
    ILimits getLimits();
    string getName();
    IOptions getOptions();
    IPPlotMgr getPlotMgr();
    ISpatialDomain getSpatialDomain();
    ITemporalDomain getTemporalDomain();
    IWeather getWeather();
    void importParameters(in string file) raises(ProjectException);
    void importWeather(in string file) raises(ProjectException);
    void open(in string file) raises(ProjectException);
    void save() raises(ProjectException);
    void saveAs(in string file) raises(ProjectException);
}

```

```

        void setName(in string name);
    };

};

};

};

#endif

```

4.9 radweapon.idl

```

/*
** @FileName: radweapon.idl
**
** Copyright (c) 2001, Science Applications International Corporation,
** all rights reserved.
**
** @Purpose: This file contains the IDL specification the IHPACServer
** corba interface.
**
*/
#ifndef _radweapon_
#define _radweapon_

#include "incident.idl"

module mil { module dtra { module hpac { module ihpacserver {
module models {
module radweapon
{
    // Typedefs
    typedef mil::dtra::hpac::ihpacserver::incident::IIncident IIncident;
    typedef sequence<string> StringList;

    // Constants
    const string RADWEAPON_MODEL = "Radiological Weapon Incident";
    const long SOURCE_LARGE = 0;
    const long SOURCE_MEDIUM = 1;
    const long SOURCE_SMALL = 2;
    const long SOURCE_USER = 3;
    const long SOURCE_ERAD = 4;

    // Exceptions
    exception RadWeaponException
    {
        string fMessage;
    };

    // Forward declarations
    interface IRadWeapon;

    // Interfaces

    // IRadWeapon interface
}
}
}
}
}

```

```

interface IRadWeapon : IIIncident
{
    StringList getAvailMaterialForms(in long type);
    StringList getAvailMaterialTypes();
    double getCalcRadius();
    boolean getCloudShine();
    double getExposureTime();
    double getHEMass();
    long getMaterialForm();
    double getMaterialMass();
    long getMaterialType();
    string getModelVersion();
    long getSourceType();
    void importERADSource(in string file)
        raises(RadWeaponException);
    void setCalcRadius(in double radius);
    void setCloudShine(in boolean shine);
    void setExposureTime(in double time);
    void setHEMass(in double mass);
    /* Deprecated */
    void setMaterialForm(in long form, in long type);
    void setMaterialFormType(in long form, in long type);
    void setMaterialMass(in double mass);
    /* Deprecated */ void setMaterialType(in long type);
    void setSourceType(in long type);
    string toMaterialFormString(in long form, in long type);
    string toMaterialTypeString(in long type);
    long valueOfMaterialForm(in string form, in string type);
    long valueOfMaterialType(in string type);
};

};

};

};

};

};

#endif


```

4.10 release.idl

```

/*
** @FileName: release.idl
**
** Copyright (c) 2001, Science Applications International Corporation,
** all rights reserved.
**
** @Purpose: This file contains the IDL specification the IHPACServer
** corba interface.
**
*/
#ifndef _release_
#define _release_

#include "parameters.idl"

```

```

module mil { module dtra { module hpac { module ihpacserver {
module release
{
    // Typedefs
    typedef mil::dtra::hpac::ihpacserver::parameters::TimeS TimeS;
    typedef mil::dtra::hpac::ihpacserver::parameters::LocationS LocationS;
    typedef sequence<string> StringList;

    // Constants
    const long RELEASE_COMMON = 1;
    const long RELEASE_CONTINUOUS = 2;
    const long RELEASE_FILE = 3;
    const long RELEASE_INSTANTANEOUS = 4;
    const long RELEASE_LIQUIDPOOL = 5;
    const long RELEASE_MOVING = 6;
    const long RELEASE_STACK = 7;

    // Exceptions
    exception ReleaseException
    {
        string fMessage;
    };

    // Forward declarations
    interface IReleaseMgr;
    interface IRelease;
    interface IContinuousRelease;
    interface IFileRelease;
    interface IInstantaneousRelease;
    interface ILiquidPoolRelease;
    interface IMovingRelease;
    interface IStackRelease;

    // Interfaces

    // IReleaseMgr interface
    interface IReleaseMgr
    {
        IRelease findRelease(in string name);
        IRelease getRelease(in long index);
        long getReleaseCount();
        StringList getReleaseList();
        long indexOfRelease(in string name);
    };

    // IRelease interface
    interface IRelease
    {
        double getHorzSize();
        double getHorzUncertainty();
        LocationsS getLocation();
        long getLocationGroup();
        string getMaterialName();
    };
}
}
}
}

```

```

double getPuffDuration();
long getReleaseType();
TimeS getStartTime();
double getVertSize();
double getVertUncertainty();
void setHorzSize(in double size);
void setHorzUncertainty(in double uncertainty);
void setLocation(in LocationsS location);
void setLocationGroup(in long group);
void setPuffDuration(in double duration);
void setStartTime(in TimeS time);
void setVertSize(in double size);
void setVertUncertainty(in double uncertainty);
};

// IContinuousRelease interface
interface IContinuousRelease : IRelease
{
    StringList getAvailDistributions();
    double getBuoyancy();
    long getDistribution();
    double getDryMassFraction();
    double getDuration();
    double getMassMeanDiameter();
    double getMassRate();
    double getMassSigma();
    double getMomentum();
    double getSigmaY();
    double getSigmaZ();
    void setBuoyancy(in double buoyancy);
    void setDistribution(in long distribution);
    void setDryMassFraction(in double fraction);
    void setDuration(in double duration);
    void setMassMeanDiameter(in double diameter);
    void setMassRate(in double rate);
    void setMassSigma(in double sigma);
    void setMomentum(in double momentum);
    void setSigmaY(in double sigma);
    void setSigmaZ(in double sigma);
    string toDistributionString(in long distribution);
    long valueOfDistribution(in string distribution);
};

// IFileRelease interface
interface IFileRelease : IRelease
{
    long getRandomCount();
    double getRandomSpread();
    void setFile(in string file) raises(ReleaseException);
    void setRandomCount(in long count);
    void setRandomSpread(in double spread);
};

```

```

// IInstantaneousRelease interface
interface IInstantaneousRelease : IRelease
{
    StringList getAvailDistributions();
    double getBuoyancy();
    long getDistribution();
    double getDryMassFraction();
    double getMass();
    double getMassMeanDiameter();
    double getMassSigma();
    double getMomentum();
    long getRandomCount();
    double getRandomSpread();
    double getSigmaX();
    double getSigmaY();
    double getSigmaZ();
    void setBuoyancy(in double buoyancy);
    void setDistribution(in long distribution);
    void setDryMassFraction(in double fraction);
    void setMass(in double mass);
    void setMassMeanDiameter(in double diameter);
    void setMassSigma(in double sigma);
    void setMomentum(in double momentum);
    void setRandomCount(in long count);
    void setRandomSpread(in double spread);
    void setSigmaX(in double sigma);
    void setSigmaY(in double sigma);
    void setSigmaZ(in double sigma);
    string toDistributionString(in long distribution);
    long valueOfDistribution(in string distribution);
};

// ILiquidPoolRelease interface
interface ILiquidPoolRelease : IRelease
{
    double getMass();
    double getSizeX();
    double getSizeY();
    void setMass(in double mass);
    void setSizeX(in double size);
    void setSizeY(in double size);
};

// IMovingRelease interface
interface IMovingRelease : IRelease
{
    StringList getAvailDistributions();
    double getBuoyancy();
    long getDistribution();
    double getDryMassFraction();
    double getDuration();
    double getMassMeanDiameter();
    double getMassRate();
}

```

```

double getMassSigma();
double getMomentum();
double getSigmaY();
double getSigmaZ();
double getVelocityX();
double getVelocityY();
double getVelocityZ();
void setBuoyancy(in double buoyancy);
void setDistribution(in long distribution);
void setDryMassFraction(in double fraction);
void setDuration(in double duration);
void setMassMeanDiameter(in double diameter);
void setMassRate(in double rate);
void setMassSigma(in double sigma);
void setMomentum(in double momentum);
void setSigmaY(in double sigma);
void setSigmaZ(in double sigma);
void setVelocityX(in double velocity);
void setVelocityY(in double velocity);
void setVelocityZ(in double velocity);
string toDistributionString(in long distribution);
long valueOfDistribution(in string distribution);
};

// IStackRelease interface
interface IStackRelease : IRelease
{
    StringList getAvailDistributions();
    double getDiameter();
    long getDistribution();
    double getDuration();
    double getExitTemperature();
    double getExitVelocity();
    double getMassMeanDiameter();
    double getMassRate();
    double getMassSigma();
    void setDiameter(in double diameter);
    void setDistribution(in long distribution);
    void setDuration(in double duration);
    void setExitTemperature(in double temperature);
    void setExitVelocity(in double velocity);
    void setMassMeanDiameter(in double diameter);
    void setMassRate(in double rate);
    void setMassSigma(in double sigma);
    string toDistributionString(in long distribution);
    long valueOfDistribution(in string distribution);
};

};

};

};

};

};

#endif

```

4.11 server.idl

```

/*
** @FileName: server.idl
**
** Copyright (c) 2001, Science Applications International Corporation,
** all rights reserved.
**
** @Purpose: This file contains the IDL specification the IHPACServer
** corba interface.
**
*/
#ifndef _server_
#define _server_

#include "project.idl"

module mil { module dtra { module hpac { module ihpacserver {
module server
{
    // Typedefs
    typedef mil::dtra::hpac::ihpacserver::project::IProject IProject;
    typedef sequence<string> StringList;

    // Constants
    const string IHPACSERVER_FACTORY_SERVICE_NAME = "IHPACServerFactory";

    // Exceptions
    exception ServerException
    {
        string fMessage;
    };

    // Forward declarations
    interface IHPACServerFactory;
    interface IHPACServer;

    // Interfaces

    // IHPACServerFactory interface
    interface IHPACServerFactory
    {
        IHPACServer getInstance();
    };

    // IHPACServer interface
    interface IHPACServer
    {
        void closeProject(in long index);
        IProject createProject(in string file);
        void deleteProject(in long index) raises(ServerException);
        IProject findProject(in string name);
        StringList getAvailProjects(in string directory);
    };
}
}
}
}

```

```
IProject getProject(in long index);
long getProjectCount();
StringList getProjectList();
long indexOfProject(in string name);
IProject loadProject(in string file);
void shutdown();
};

};

};

};

};

#endif
```

CHAPTER 5

Java Beans and Components

A plethora of reusable Java components comprise the HPAC client application, the Project Editor. Many components are shared by client and server components. The Java interfaces and classes are documented in detail and organized hierarchically by their Java package, the Java archive (JAR) file in which they're distributed, and the location in the HPAC installation where they can be found. Shared jar files are installed in the `shared/ext` subdirectory under the HPAC installation directory. Server jar files are found in `server/ext` and client jar files are in `client/ext`.

Jar files available in the HPAC distribution are summarized in the following sections. The API and detailed descriptions of these components appears in the companion document *HPAC Java Beans and Components*.

Available online at the URL

<http://wigner.cped.ornl.gov/hpac/4.0.1/api/>

is the API documentation for the contents of the following jars: `clientutils`, `dtraswing`, `dtrautil`, `fileutils`, `hpacclient`, `hpacitpts`, `scipuffdefs`, `scipuffserver`, and `scipuffshared`.

5.1 SHARED JAR FILES

These jars are located in the `shared/ext` subdirectory under the HPAC installation.

cbfacshared.jar IDL-defined data structures and interface stubs for the Chemical-Biological Facility source model. Packages in the jar are:

mil.dtra.hpac.models.CBFac.data
mil.dtra.hpac.models.CBFac.server
mil.dtra.hpac.models.CBFac.Helpers.DWFI
mil.dtra.hpac.models.CBFac.Helpers.DamCat
mil.dtra.hpac.models.CBFac.Helpers.SWFI
mil.dtra.hpac.models.CBFac.Helpers.server
mil.dtra.models.CBFac.AgentsList
mil.dtra.models.CBFac.CloudTrans
mil.dtra.models.CBFac.CloudTrans.Helpers
mil.dtra.models.CBFac.DWFI
mil.dtra.models.CBFac.DWFI.Helpers

mil.dtra.models.CBFac.DamCat
 mil.dtra.models.CBFac.DamCat.Helpers
 mil.dtra.models.CBFac.SWFI
 mil.dtra.models.CBFac.SWFI.Helpers
 mil.dtra.models.CBFac.WeaponServer

cbwpnshared.jar IDL-defined data structures and interface stubs for the Chemical-Biological Weapon source model. Packages in the jar:

- mil.dtra.hpac.models.cbwpn.data
- mil.dtra.hpac.models.cbwpn.server

datums.jar Components for converting to and from UTM coordinates using specific datums.

Packages in the jar:

- mil.dtra.datums
- mil.dtra.datums.constants
- mil.dtra.datums.convert
- mil.dtra.datums.data
- mil.dtra.datums.mgrs
- mil.dtra.datums.molod
- mil.dtra.datums.utils
- mil.dtra.datums.utm

dtrautil.jar Common utility components and utilities including properties serialization support and unit conversions; also defines the map display framework. Packages in the jar:

- mil.dtra.map
- mil.dtra.map.imagemap
- mil.dtra.units
- mil.dtra.util
- mil.dtra.util.parsers

fileutils.jar Facilities for processing file references for transfer from client to server. Packages in the jar:

- mil.dtra.hpac.server.files
- mil.dtra.hpac.server.fileutils
- mil.dtra.hpac.server.impl

hpacitpts.jar Agent for plugging HPAC into the Integrated Target Planning Toolset (ITPTS). Packages in the jar:

- mil.dtra.hpac.itpts.client
- mil.dtra.hpac.itpts.tool.impl
- mil.dtra.hpac.itpts.tool.impl

hpacshared.jar IDL-defined data structures for incidents and releases as well as properties-serializable Java class wrappers; contains the interfaces and abstract base classes for the incident source model server framework. Packages in the jar:

- mil.dtra.hpac.data
- mil.dtra.hpac.server
- mil.dtra.hpac.server.effects
- mil.dtra.hpac.server.radfile

mil.dtra.hpac.server.release
 mil.dtra.hpac.units

IHPACServer.jar Shared and server components for IHPACServer. Packages in the jar:

mil.dtra.hpac.ihpacserver.dispersion
 mil.dtra.hpac.ihpacserver.incident
 mil.dtra.hpac.ihpacserver.models.chembioweapon
 mil.dtra.hpac.ihpacserver.models.missileintercept
 mil.dtra.hpac.ihpacserver.models.nuclearweapon
 mil.dtra.hpac.ihpacserver.models.radweapon
 mil.dtra.hpac.ihpacserver.parameters
 mil.dtra.hpac.ihpacserver.plot
 mil.dtra.hpac.ihpacserver.project
 mil.dtra.hpac.ihpacserver.release
 mil.dtra.hpac.ihpacserver.server

materialshared.jar IDL-defined data structures and interface stubs for the material server. Packages in the jar:

mil.dtra.hpac.material.data
 mil.dtra.hpac.material.server

mintshared.jar IDL-defined data structures and interface stubs for the Missile Intercept source model. Packages in the jar:

mil.dtra.hpac.models.mint.data
 mil.dtra.hpac.models.mint.server

nfacshared.jar IDL-defined data structures and interface stubs for the Nuclear Facility source model. Packages in the jar:

mil.dtra.hpac.models.nfac.data
 mil.dtra.hpac.models.nfac.data.analyst
 mil.dtra.hpac.models.nfac.server
 mil.dtra.hpac.models.nfac.server.analyst

nwishedited.jar IDL-defined data structures and interface stubs for the Nuclear Weapon Incident source model. Packages in the jar:

mil.dtra.hpac.models.nfac.data
 mil.dtra.hpac.models.nfac.server

nwpnglobal.jar Additional support for the Nuclear Weapon source model. Packages in the jar:

mil.dtra.hpac.models.nwpn.data
 mil.dtra.hpac.models.nwpn.dataserver
 mil.dtra.hpac.models.nwpn.utils

nwpnshared.jar IDL-defined data structures and interface stubs for the Nuclear Weapon source model. Packages in the jar:

mil.dtra.hpac.models.nwpn.server

rwpnshared.jar IDL-defined data structures and interface stubs for the Radiological Weapon source model. Packages in the jar:

mil.dtra.hpac.models.nwpn.data
 mil.dtra.hpac.models.nwpn.server

scipuffdefs.jar IDL-defined data structures defining additional input to the T&D calculation engine via ScipuffServer. Packages in the jar:
 mil.dtra.hpac.server.plot
 mil.dtra.hpac.server.project

scipuffshared.jar IDL-defined interface stubs for ScipuffServer. Packages in the jar:
 mil.dtra.hpac.server.scipuff

smokeshared.jar IDL-defined data structures and interface stubs for the Smoke Munition source model. Packages in the jar:
 mil.dtra.hpac.models.swpn.data
 mil.dtra.hpac.models.swpn.server

stcalcshared.jar IDL-defined data structures and interface stubs for StcalcServer. Packages in the jar:
 mil.dtra.hpac.data.stcalc
 mil.dtra.hpac.server.stcalc

strikestable.jar StrikeServer implementation supporting the Nuclear Weapon source model; includes client stubs and server skeletons. Packages in the jar:
 mil.dtra.hpac.models.nwpn.strikeserver
 mil.dtra.hpac.models.nwpn.strikeserver.data
 mil.dtra.hpac.models.nwpn.strikeserver.exceptions
 mil.dtra.hpac.models.nwpn.strikeserver.readers
 mil.dtra.hpac.models.nwpn.strikeserver.weapons

weathershared.jar IDL-defined data structures defining the weather specification. Packages in the jar:
 mil.dtra.weather.shared.data
 mil.dtra.weather.shared.data.weatherT

5.2 SERVER JAR FILES

These jars are located in the `server/ext` subdirectory under the HPAC installation.

cbfacserver.jar Implementation of services defined for the Chemical-Biological Facility source model. Packages in the jar:
 mil.dtra.hpac.models.CBFac.server.impl
 mil.dtra.models.CBFac.AgentsList.impl
 mil.dtra.models.CBFac.CloudTrans.impl
 mil.dtra.models.CBFac.DWFI.impl
 mil.dtra.models.CBFac.DamCat.impl
 mil.dtra.models.CBFac.SWFI.impl
 mil.dtra.models.CBFac.WeaponServer.impl

cbwpnserver.jar Implementation of services defined for the Chemical-Biological Weapon source model. Packages in the jar:
 mil.dtra.hpac.models.cbwpn.server.impl

hpacserver.jar Implementation of common support for source models as well as the HPAC Server-Launcher. Packages in the jar:
 mil.dtra.hpac.server.impl
 mil.dtra.hpac.server.radfile.impl

nfacserver.jar Implementation of services defined for the Nuclear Facility source model. Packages in the jar:
 mil.dtra.hpac.models.nfac.server.impl

nwiserver.jar Implementation of services defined for the Nuclear Weapon Incident source model. Packages in the jar:
 mil.dtra.hpac.models.nwi.server.impl

nwpnserver.jar Implementation of services defined for the Nuclear Weapon source model. Packages in the jar:
 mil.dtra.hpac.models.nwpn.server.data
 mil.dtra.hpac.models.nwpn.server.ftndlls
 mil.dtra.hpac.models.nwpn.server.impl

rwpnserver.jar Implementation of services defined for the Radiological Weapon source model. Packages in the jar:
 mil.dtra.hpac.models.rwpn.server.impl

scipuffserver.jar Implementation of ScipuffServer and ScipuffServerFactory services. Packages in the jar:
 mil.dtra.hpac.server.EntityRelationship
 mil.dtra.hpac.server.scipuff.impl
 mil.dtra.hpac.server.scipuff.jni
 mil.dtra.hpac.server.utils

smokeserver.jar Implementation of services defined for the Smoke Munition source model. Packages in the jar:
 mil.dtra.hpac.models.swpn.server.impl

stcalcserver.jar Implementation of StcalcServer. Packages in the jar:
 mil.dtra.hpac.server.stcalc.impl

5.3 CLIENT JAR FILES

These jars are located in the `client/ext` subdirectory under the HPAC installation.

cbfacclient.jar Bean and support components for the Chemical-Biological Facility source model. Packages in the jar:
 mil.dtra.hpac.models.CBFac.client
 mil.dtra.models.CBFac.WhatUI

cbwpnclient.jar Bean and support components for the Chemical-Biological Weapon source model. Packages in the jar:
 mil.dtra.hpac.models.cbwpn.client

clientutil.jar Components for determining the user's directory for HPAC files based on property settings; also provides graphical user interface (GUI) beans for user specification of their HPAC directory. Packages in the jar:
 mil.dtra.hpac.client.util

coda.jar Beans for user definition of human effects parameters. Packages in the jar:
 mil.dtra.hpac.client.plot.effects

dtraswing.jar Collection of Java Foundation Classes (JFC) and Swing extensions as well as additional components. Packages in the jar:
 mil.dtra.awt
 mil.dtra.swing

help.jar Collection of Java Foundation Classes (JFC) and Swing extensions as well as additional components. Packages in the jar:
 mil.dtra.hpac.help

hpacclient.jar Implementation of the HPAC Project Editor; also contains interfaces and abstract base classes for points-of-interest, map display, and incident source model bean frameworks. Packages in the jar:
 mil.dtra.hpac.client
 mil.dtra.hpac.client.dialog
 mil.dtra.hpac.client.display
 mil.dtra.hpac.client.event
 mil.dtra.hpac.client.folder
 mil.dtra.hpac.client.io
 mil.dtra.hpac.client.models
 mil.dtra.hpac.client.poi
 mil.dtra.hpac.client.swing
 mil.dtra.hpac.client.swing.location
 mil.dtra.hpac.client.swing.location.lls
 mil.dtra.hpac.client.swing.project
 mil.dtra.hpac.client.swing.release
 mil.dtra.hpac.client.swing.time

hpacmap.jar Map display framework implementation based on OpenMap™. Packages in the jar:
 mil.dtra.map.openmap

jchart.jar General charting components used for displaying graphics in the Chemical-Biological Facility model bean; commercial off-the-shelf (COTS) software.

landsea.jar Implementation of an OpenMap™ layer for displaying the "Land Sea" map data. Packages in the jar:
 mil.dtra.map.openmap.layer
 mil.dtra.map.openmap.layer.landsea

materialclient.jar GUI bean for editing material properties and creating new materials; communicates with the MaterialServer. Packages in the jar:
 mil.dtra.hpac.client.swing.material

mckoidb.jar Commercial components providing SQL database access.

mintclient.jar Bean and support components for the Missile Intercept source model. Packages in the jar:

- mil.dtra.hpac.models.mint.client
- mil.dtra.hpac.models.mint.client.swing

nfacclient.jar Bean and support components for the Nuclear Facility source model. Packages in the jar:

- mil.dtra.hpac.models.nfac.client
- mil.dtra.hpac.models.nfac.client.swing
- mil.dtra.hpac.models.nfac.client.swing.facility
- mil.dtra.hpac.models.nfac.client.swing.model
- mil.dtra.hpac.models.nfac.client.swing.model.analyst
- mil.dtra.hpac.models.nfac.client.swing.times

nwiclient.jar Bean and support components for the Nuclear Weapon Incident source model. Packages in the jar:

- mil.dtra.hpac.models.nwi.client
- mil.dtra.hpac.models.nwi.client.swing

nwpnclient.jar Bean and support components for the Nuclear Weapon source model. Packages in the jar:

- mil.dtra.hpac.models.nwpn.client
- mil.dtra.hpac.models.nwpn.client.data
- mil.dtra.hpac.models.nwpn.client.dialogs

obscure.jar Beans for user definition of smoke effects parameters. Packages in the jar:

- mil.dtra.hpac.models.smoke.effects

openmap.jar Open source components providing a map display capability. (Note this is version 3.6.2 of the OpenMap™ jar. The current version maintained by BBN as of this writing is 4.5.3).

plot.jar Beans for user selection of plot options and parameters, communication with Scipuff-Server to retrieve plot data; implementation of plot overlay components for plot rendering; implementation of data export capabilities. Packages in the jar:

- mil.dtra.hpac.client.plot.data
- mil.dtra.hpac.client.plot.display
- mil.dtra.hpac.client.plot.swing
- mil.dtra.hpac.client.plot.utils

poi.jar Implementation of the weather station and nuclear facility POI groups. Packages in the jar:

- mil.dtra.hpac.client.io
- mil.dtra.hpac.client.pio

reqgen.jar Weather data retrieval support. Packages in the jar:

- mil.dtra.weather.client.images
- mil.dtra.weather.client.io
- mil.dtra.weather.client.reqgen
- mil.dtra.weather.client.swing
- mil.dtra.weather.data

rwpnclient.jar Bean and support components for the Radiological Weapon source model. Packages in the jar:

- mil.dtra.hpac.models.rwpn.client
- mil.dtra.hpac.models.rwpn.client.swing

smokeclient.jar Bean and support components for the Smoke Munition source model.

- mil.dtra.hpac.models.swpn.client
- mil.dtra.hpac.models.swpn.client.swing

stneditor.jar Weather definition support. Packages in the jar:

- mil.dtra.weather.client
- mil.dtra.weather.client.io
- mil.dtra.weather.client.stneditor
- mil.dtra.weather.data

terrain.jar Weather definition support. Packages in the jar:

- mil.dtra.weather.client.display
- mil.dtra.weather.client.terrldr
- mil.dtra.weather.client.terrldr.data
- mil.dtra.weather.client.terrldr.landuse
- mil.dtra.weather.client.terrldr.swing
- mil.dtra.weather.client.terrldr.util

unvwxrdr.jar Universal weather reader. Packages in the jar:

- mil.dtra.weather.client.unvwxrdr
- mil.dtra.weather.exceptions

weatherclient.jar Implementation of the weather button. Packages in the jar:

- mil.dtra.weather.client
- mil.dtra.weather.client.display
- mil.dtra.weather.client.io
- mil.dtra.weather.data
- mil.dtra.weather.exceptions

wxeditor.jar Weather definition support. Packages in the jar:

- mil.dtra.weather.client.images
- mil.dtra.weather.client.io
- mil.dtra.weather.client.swing
- mil.dtra.weather.client.wxeditor
- mil.dtra.weather.data
- mil.dtra.weather.exceptions

wxplot.jar Weather plotting support. Packages in the jar:

- mil.dtra.weather.client.data
- mil.dtra.weather.client.display
- mil.dtra.weather.client.io

wxutils.jar Beans supporting user edits of the weather defintion. Packages in the jar:

- mil.dtra.weather
- mil.dtra.weather.client.io

mil.dtra.weather.client.swing
mil.dtra.weather.data
mil.dtra.weather.exceptions

PART III

Interfaces for Extending HPAC

CHAPTER 6

Incident Source Models

HPAC was explicitly designed to allow addition of incident source models to a deployed system. Source models come in two parts:

1. a server side containing a factory and a per-client server implementation (described in Section 1.1.2),
2. a client bean for user interaction in the HPAC Project Editor application.

If the desire is for HPAC server side processing only, the client bean is not necessary. Both the server and client sides of an incident source model implementation must adhere to a framework.

6.1 MODEL SERVER FRAMEWORK

As stated in Section 3.3.2, the purpose of a source model is to take a parameterized description of an operational incident and produce detailed releases suitable for input to the T&D calculation engine. The collaboration and flow sequence for a client accessing an incident source model server is described in Section 3.3.2. The *server.idl* file listed in Section 3.4.5 specifies the framework for a model server. There are two interfaces of significance: `IncidentModelServer` and `IncidentModelServerFactory`.

Although a model can implement the defined interfaces, it is suggested that a model provide its own IDL defining interfaces extending `IncidentModelServer` and `IncidentModelServerFactory` and defining a structure for the model-specific incident data described below in Section 6.1.3.

6.1.1 IncidentModelServerFactory

A model implementation must provide a factory server object implementing the `IncidentModelServerFactory` interface, which specifies a single method, `getInstance()`. This method returns a per-client server object specific to the model but implementing the `IncidentModelServer` interface. Any additional methods may be specified in model-specific factory interface that extends `IncidentModelServerFactory`.

Further, a model factory will be a single instance (i.e., singleton) registered in the naming service. In order to be launched by the HPAC ServerLauncher, the model factory must also implement

the Java interface `FactoryServer` defined in the `mil.dtra.hpac.server.impl` package and bundled in `hpacserver.jar`.

6.1.2 FactoryServer

`FactoryServer` specifies two methods. The first, `getDefaulName()`, is called by `ServerLauncher` and must return the unique service name for the factory. This is the name by which the factory server is bound in the naming service. Generally, it's the model short name followed by "Factory". For example, "NfacFactory" is returned by the Nuclear Facility model factory.

The second method, `setContext()`, is called by `ServerLauncher` and passes Java Naming and Directory Interface (JNDI) context handle. This is the model's opportunity to store this reference for future lookups of other server objects, such as the material server. `ServerLauncher` will automatically instantiate and bind in the naming service any `FactoryServer` implementation specified in the server properties file, as described in Section 3.1. A model factory or per-client server is free to bind or register additional server objects in the registry after the `setContext()` method has been called.

6.1.3 IncidentModelServer

Eight methods are defined in the `IncidentModelServer` interface. An abstract base class implementing `IncidentModelServer` is provided in the `mil.dtra.hpac.server.impl` package as the `IncidentModelServerImpl` class. It provides noop implementations of `closeIncident()` and `terminate()` and default implementations of `computeEffect()`, `restoreCustomizedProperties()`, and `updateRelease()`. The default `updateRelease()` makes a copy of the release parameter, calls `updateReleaseData()`, and then calls `restoreCustomizedProperties()`.

Thus, a model server implementation need only provide implementations of three methods: `initIncident()`, `updateIncident()`, and `updateReleaseData()`. It is free to override the others.

The sequence for method calls as per the steps outlined in Section 3.3.2 provides a better way to describe them. Refer to Section 3.4.5 for a listing of the `server.idl` file.

Model Incident Data

Incident data are divided into two parts. One is the standard incident definition shared by all models and used throughout HPAC. This is the `IncidentT` IDL structure specified in `server.idl`. Another is the description of the operational incident specific to the model. Since this can be anything the model defines, it is passed to and from model server methods as a CORBA any type.

Method initIncident()

First of the methods that may be called is `initIncident()`. In the original design, the parameters were *out* instead of *inout*. The change to *inout* was made to satisfy the needs of a particular model. In future versions they are likely to again be specified as *out* parameters. Regardless, a model's `initIncident()` method is responsible for returning a default, valid incident definition. This includes both the model-specific incident structure and the shared `IncidentT` structure. Note that in order to be a valid incident, the `fReleaseList` field of the `IncidentT` must

include at least one valid instance of `ReleaseT`. `ReleaseT` is defined in the *release.idl* file listed in Section 3.4.4.

There is one caveat on the validity of the returned incident objects. Prior to calculation `ScipuffServer` will call the `updateIncident()` method of each incident's model server in *last chance* mode, described below. Consequently, the results of `initIncident()` and `updateIncident()` must be such that `updateIncident()` in *last chance* mode can produce an incident (and releases) suitable for T&D calculation.

For this method a model may return canned or pre-fabricated objects, or it may dynamically build the objects. Either way, the result must be a valid incident description ready to be passed to a calculation method of `ScipuffServer`.

Method updateIncident()

The `updateIncident()` may be called many times and will almost definitely be called twice. It has two modes of operation determined by the `request_mask` parameter. If the `HU_LASTCHANCE` bit of `request_mask` is set, it is a *last chance* call from `ScipuffServer` prior to a calculation. Otherwise, the call is very similar to `initIncident()` in that the model is responsible for producing a "valid" incident (see discussion above). It differs from `initIncident()` in that the fields and properties of the incident and `model_incident` parameters are now input as well, and the model must respond to the settings accordingly. If the inputs are invalid or unacceptable, the model must throw a `ModelException`.

A *last chance* call occurs as the last call to the model **before** starting a T&D calculation in HPACtool. This allows the model to avoid invoking expensive computations or processing multiple times.

Methods updateRelease() and updateReleaseData()

The `fStatus` field of the `ReleaseT` structure holds an array of status values indexed by constants provided in *release.idl*. The first entry (`RSI_RELEASE_STATUS`) is the status for the entire release. If any portion of the release depends on weather or environmental conditions, the release status must be set to `RS_INVALID`. The status value of the deferred fields should also be marked as invalid as well. During calculation and as weather data are read, HPACtool will check for releases marked invalid and call back through `ScipuffServer` to the `updateRelease()` method with `HU_ENVIRONMENT` bit set in the mode parameter. Parameters to the method include the environment and the next update time.

If the model server implementation extends `IncidentModelServerImpl` and does not override `updateRelease()`, the `updateReleaseData()` method will be called. Note for both methods the release parameter is *inout*, which means the implementer receives the input release and must provide the updated result via the same parameter.

If the status of the release is set to `RS_VALID`, no `updateRelease()` call will occur.

Method computeEffect()

Under version 4.0.1, the model effects mechanism supports blast and prompt effects related to nuclear weapons. Model implementations should not override the `computeEffect()` method inherited from `IncidentModelServerImpl` that reports no support for effects computations.

Method terminate()

The `terminate()` method is a hook for model server implementers to clean up any resources acquired or created during the life time of the server object instance. It is invoked prior to a CORBA `_release()` call. Inherited from `IncidentModelServerImpl` is a no-op version.

Method closeIncident()

Whereas `terminate()` is to free resources associated with a single server instance, `closeIncident()` is a hook for models to clean up resources that were created by some server instance and persisted between instances. There are no calls to this method from `ScipuffServer` or from the default model implementation. Rather, requirements for clients to call this method are part of the “contract” of the specific model.

6.2 MODEL BEAN FRAMEWORK

In the HPAC client application, Project Editor, incident source models work within a framework based on the JavaBeans™ paradigm. A model must include its bean class, a `BeanInfo` class describing various properties, and a *customizer* or editor GUI bean for modifying model properties. In addition, a simple class specifying properties for an incident definition icon must be provided. All of the framework interfaces and base classes are bundled in `hpacclient.jar`.

6.2.1 Model Bean

In order to standardize behavior and appearance of incident source model beans in HPAC, all model beans must extend the `IncidentModel` class defined in the `mil.dtra.hpac.client.models`. package. Basically, the model bean is a button. Each incident defined for the project in Project Editor is represented by a button in the object palette, and the buttons are the model beans.

Method createIncidentObjects()

Most of what a model needs is provided in `IncidentModel`, so a model is required to implement only one method, `createIncidentObjects()`, in addition to a no-arg constructor. The return value of `createIncidentObjects()` is an `Object[]` and must contain two values in order: an `Incident` and a `ModelIncident`. These are defined in the `mil.dtra.hpac.data`. package in `hpacshared.jar`.

`Incident` is the properties-serializable Java class corresponding to the `IncidentT` IDL structure defined in `server.idl`. `ModelIncident` is a Java interface specifying methods to convert an instance to and from a CORBA any value. A model implementer must include a Java class which implements the `ModelIncident` and `PropsSerializer` interfaces and corresponds to the model incident IDL structure defined for the model as described in Section 6.1.3. The class should include methods to convert to and from the IDL structure. Note these conversions are provided in the `Incident` class.

The objects returned by `createIncidentObjects()` must represent a valid incident definition for the model, as described in Section 6.1.3. Most likely, the code for this method will contact the model’s server and call the `initIncident()` method. From `IncidentModel` the model inherits a reference to an object implementing the `ProjectEditorIfc` interface, accessible via the

`getProjectEditor()` method. The `ProjectEditorIfc` object in turn holds a reference to another object, an instance of `ProjectEditorProfile` retrieved via the `getProfile()` method.

The profile has references to the naming service contexts by which the model can look up its server. Call the `getNamingContext()` and `getRMINamingContext()` methods of the profile for lookups of CORBA and RMI servers, respectively. `ProjectEditorIfc` and `ProjectEditorProfile` are defined in the `mil.dtra.hpac.client`. package.

Other Methods to Override

There are other `IncidentModel` methods a model may wish to override. One of these is `createMapIcons()`. The default, inherited behavior is to create a `ReleaseIcon` for each location group for releases in the incident's release list. The icon is the 32x32 pixel color icon defined in the model's `BeanInfo` class. A model can override this method for different behavior. Note a call to `super.createMapIcons()` can be used to merely modify icons created in the default manner. For example, a model may desire that icons not be draggable on the map display.

Another method to override is `createPopupMenu()`. Any override should call `super.createPopupMenu()` first to get the default Edit, Delete, and Show Releases items.

Properties serialization implemented in `IncidentModel` follows an explicit model and processes the `modelIncident`, `incident`, `showMapIconsFlag`, and `notes` properties. If a model has additional properties to (de) serialize, it should override the `readProps()` and `writeProps()` methods, taking care to call the `super` methods within them. Refer to Chapter 9 for a description of HPAC's properties (de)serialization mechanisms.

6.2.2 Model BeanInfo

Part of the HPAC model bean framework is an extension of the `java.beans.SimpleBeanInfo` class called `ModelBeanInfo`. Models must provide an extension of `ModelBeanInfo`, but this extension need only provide a no-arg constructor invoking the `ModelBeanInfo` constructor. It has four parameters, two classes and two strings. The classes are the model bean and customizer. The strings are the display name and a short description. The display name is used in buttons in the object palette and should be a short abbreviation for the model name in six characters or less. The short description appears in tool tips and should be a short phrase of five words or less giving the full name of the model. Examples are "NFAC" for a display name and "Nuclear Facility Incident" for a short description.

6.2.3 Model Bean Customizer

A bean customizer is a GUI component (extension of `java.awt.Component`) providing the user a means of viewing and editing the properties of the bean. Since the model bean itself is a button, the customizer is the editor for the model's operational incident description.

Once again, HPAC defines an abstract base class that must be extended by models, `ModelPanel`, to provide the customizer. `ModelPanel` provides a good deal of default behavior. The key methods a model implementer must provide are `init()`, `load()`, and `store()`. `ModelPanel` uses a tabbed pane component for user interaction.

Method init()

The `init()` method is responsible for creating the controls and beans comprising the editor component. `ModelPanel`'s version creates a panel for editing the incident name, and the tabbed pane. Further, `ModelPanel` provides `addWhenTab()`, `addWhereTab()`, and `addNotesTab()` methods creating standard "When", "Where", and "Notes" components for tabs by those names. A model should call `super.init()` in its `init()`. The tabbed pane created in `ModelPanel.init()` is accessible via the `getTabbedPane()` method and may be used for `addTab()` calls to add tabs desired by the model.

Method load()

The `load()` method is called to populate the customizer bean with a specific model bean, or more precisely its incident and model incident objects. These objects must be retrieved from the model bean via `getIncident()` and `getModelIncident()` methods defined for `IncidentModel`. The model bean itself is accessed in the customizer via the `getModelBean()` method provided by `ModelPanel`.

Properties of the incident and model incident should be used to populate the beans in the customizer. Default processing for default beans, if they were created from `init()`, is provided in `ModelPanel.load()`. This includes a call to `updateReleaseProperties()` on the incident, setting the name text field, initializing the standard location ("When" tab) and start time ("What" tab) beans, and loading the notes text area. A model's `load()` method must call `super.load()` before any other processing.

Method store()

When the user "applies" or "okays" the dialog containing the customizer, the `store()` method is called. It is responsible for sending the incident and model incident to the server's `updateIncident()` method and storing the results back into the model bean. Whereas `load()` must call the superclass method, this is not necessary with `store()` since it is called from another `ModelPanel` method, `storeAndUpdate()`. Properties associated with standard beans such as the location and start time beans are set in `storeAndUpdate()`.

6.2.4 Icon Definition

For each model defined in the HPAC configuration, an icon is placed in the incident definition palette in the Project Editor. Information necessary to build the icon and make it work is provided in a class extending the HPAC `IncidentDefinition` class. The model's extension is very simple and need only provide a no-arg constructor which calls either the three- or four-arg `IncidentDefinition` constructor. The required parameters are the bean name, display name, and short description, all strings. The bean name is the full class path to the model bean. The display name and short description are identical to those for the `ModelBeanInfo` extension, except these are applied to the icon in the incident definition palette.

6.2.5 Configuring HPAC Models

Configuration of the HPAC client application, Project Editor, is specified in the `hpac.properties` file in the `client` / subdirectory under the HPAC installation. Two property keys in the file pertain

to models: *hpac.modelBeans* and *hpac.palettes*.

A comma-separated list of class paths of available model beans is the value of the *hpac.modelBeans* property. The HPAC 4.0.1 distribution sets it as follows:

```
hpac.modelBeans=\
mil.dtra.hpac.client.models.Advanced,\
mil.dtra.hpac.models.cbwpn.client.ChemBioWeapon,\
mil.dtra.hpac.models.CBFac.client.CBFacWeapon,\
mil.dtra.hpac.models.swpn.client.SmokeWeapon,\
mil.dtra.hpac.models.mint.client.MissileIntercept,\
mil.dtra.hpac.models.nwi.client.NWI,\
mil.dtra.hpac.models.nfac.client.Nfac,\
mil.dtra.hpac.models.nwpn.client.Nwpn,\
mil.dtra.hpac.models.rwpn.client.RWPN
```

Note the backslash (\) merely escapes the newline to make successive lines appear as one. The value of this property determines the contents of the *Edit -> Add Incident* pullright menu.

The contents of the incident definition palette are determined by the *hpac.palettes* property. Each palette entry specifies a group, and within each group are a list of icons specifying the class path to the model's *IncidentDefinition* extension. The values for HPAC 4.0.1 follow.

```
hpac.palettes.0.name=WMD Usage
hpac.palettes.0.icons.0=mil.dtra.hpac.models.cbwpn.client.ChemBioWeaponDefinition
hpac.palettes.0.icons.1=mil.dtra.hpac.models.nwpn.client.NwpnDefinition
hpac.palettes.0.icons.2=mil.dtra.hpac.models.rwpn.client.RWPNDefinition

hpac.palettes.1.name=NBC Releases
hpac.palettes.1.icons.0=mil.dtra.hpac.models.CBFac.client.CBFacWeaponDefinition

hpac.palettes.1.icons.1=mil.dtra.hpac.models.nwi.client.NWIDefinition
hpac.palettes.1.icons.2=mil.dtra.hpac.models.mint.client.MissileInterceptDefinition
hpac.palettes.1.icons.3=mil.dtra.hpac.models.nfac.client.NfacDefinition

hpac.palettes.2.name=Other
hpac.palettes.2.icons.0=mil.dtra.hpac.client.models.AdvancedDefinition
hpac.palettes.2.icons.1=mil.dtra.hpac.models.swpn.client.SmokeWeaponDefinition
```

CHAPTER 7

Map Display

As for other Project Editor customizations, available map displays are configured in the *hpac.properties* file in the `client` / subdirectory under the HPAC installation. The property key is `hpac.maps`, which has subkeys starting with `0` to specify individual maps. The first (0th) entry is the default map displayed when Project Editor is initialized. Each entry is a comma-delimited triple specifying the name of the map in the Project Editor menu, the path of the class implementing the map display, and a full or relative URL to a configuration file specific to the map display implementation. Example entries are:

```
hpac.maps.0=Global Land/Sea,mil.dtra.map.openmap.OmMapDisplay,\n${hpac.rootURL}data/maps/landsea/landsea.map\nhpac.maps.1=Global Political,mil.dtra.map.openmap.OmMapDisplay,\n${hpac.rootURL}data/maps/admin/admin.map\nhpac.maps.2=North Georgia Topo Image,mil.dtra.map.imagemap.ImageMapDisplay,\n  data/north-georgia.imagemap
```

Note in the first two URLs the reference to the `hpac.rootURL` system property, which specifies the full URL to the `client` / subdirectory in the HPAC installation. The third example above shows a relative URL. All relative URLs assume the value of `hpac.rootURL` as the base.

Also note there are two different map display implementation classes specified in the above example. There are two ways to customize an individual map display definition. One is to provide a custom map display implementation, and the other is to use the existing implementations but provide custom configurations.

7.1 MAP DISPLAY FRAMEWORK

7.1.1 MapDisplay and MapProjection

Interfaces and classes in the `mil.dtra.map`. package (distributed in `dtrutil.jar`) define the map display framework. The primary interface is `MapDisplay`. It defines zooms and pan methods as well as methods for dealing with `MapProjections` and user edits of the map display configuration. Classes implementing this interface must extend `javax.swing.JComponent` and render via the `paintComponent()` method. Optionally, they can extend `AbstractMapDisplay` and inherit implementations of the pan and zoom methods. The `define()` and `edit()` methods must be provided in the extension.

The `MapProjection` interface defines the forward and reverse coordinate conversions a projection must supply. `AbstractProjection` is an abstract base class isolating the specifics of a projection in two methods, `projectExtToWorld()` and `projectWorldToExt()`, representing conversion to and from *extension* coordinates. both of which must be provided in an extension, and all the other projection machinery is implemented in `AbstractProjection`. A single concrete extension of `AbstractProjection` is provided in `RectangularProjection`.

7.1.2 ImageMapDisplay

A reference implementation of the map display framework is included in the `mil.dtra.map.imagemap` package. It simply takes an image and and a specification of the world coordinate extents (or georeference) for the image and displays them, projected as necessary. The configuration file identified in the third item of an `hpac.maps` entry is a properties file with three keys, as shown below in the listing of `north-georgia.imagemap` from the example above.

```
description=Topographical Depiction of North Georgia
image=north-georgia.gif
mapWindow=-86.525,32.330,4.797,2.736
```

The `mapWindow` property specifies the west, south, width, and height of the image's extent. The image file itself is specified with the `image` property and must reside in the same URL directory as the configuration file. An optional property, `description`, is not processed.

Note map images may also be displayed as a layer in an OpenMap configuration using the `OmMapDisplay` map display implementation as described below.

7.2 OPENMAP MAP DISPLAY IMPLEMENTATION

For most HPAC map displays, `OmMapDisplay` from the `mil.dtra.map.openmap` package in `hpacmap.jar` is the implementation. Built on version 3.6.2 of OpenMap™, `OmMapDisplay` plugs OpenMap's capabilities into the map display framework by implementing the `MapDisplay` interface.¹

OpenMap configurations are specified as a properties file, again identified by the URL in the `hpac.maps` entry in the application properties file. Documentation on configuring or extending OpenMap is provided at the OpenMap Web site:

<http://openmap.bbn.com>

An abbreviated example configuration file is listed below.

```
openmap.layers=graticule contrast scaled
openmap.startupLayers=scaled

graticule.class=mil.dtra.map.openmap.layer.GraticularLayer2
graticule.prettyName=Graticule
graticule.showRuler=true
graticule.show1And5Lines=true
```

¹This is an old version of OpenMap. As of the date of this document, the current version is 4.5.3

```

graticule.threshold=2
graticule.10DegreeColor=FF000000
graticule.5DegreeColor=FF009900
graticule.1DegreeColor=FF003300
graticule.TenthsDegreeColor=FF003300
graticule.equatorColor=FFFF0000
graticule.dateLineColor=FF000099
graticule.specialLineColor=FF000000
graticule.textColor=FF000000

contrast.class=mil.dtra.map.openmap.ContrastLayer
contrast.prettyName=Contrast Mask
contrast.color=80ffffff

scaled.class=com.bbn.openmap.layer.ScaleFilterLayer
scaled.prettyName=Administrative Boundaries
scaled.layers=po adm
scaled.transitionScales=40000000

po.class=com.bbn.openmap.layer.shape.areas.AreaShapeLayer
po.prettyName=Political Outlines
po.shapeFile=${dot}/country.shp
po.spatialIndex=${dot}/country.ssx
po.csvFile=file:///${dot}/country.csv
po.lineColor=ff000000
po.fillColor=ffbdbdde83

adm.class=com.bbn.openmap.layer.shape.areas.AreaShapeLayer
adm.prettyName=Administrative Boundaries
adm.shapeFile=${dot}/admin98.shp
adm.spatialIndex=${dot}/admin98.ssx
adm.csvFile=file:///${dot}/admin98.csv
adm.lineColor=ff000000
adm.fillColor=ffbdbdde83

```

Each layer identified in the *openmap.layers* property must be specified as its own property. A layer's *class* property gives the path to the class of the OpenMap Layer object, and the *prettyName* property is used to identify the layer in user interactions. The remaining properties are specific to the layer implementation. Note the use of the \${dot} notation. This is specific to OmMapDisplay and represents the URL directory in which the configuration file itself exists.

Use of OmMapDisplay offers two means of tailoring or customizing available map displays. First, one can modify an existing configuration file or add new ones to provide additional layers. Second, one can extend the OpenMap Layer class or one of the extensions that comes with OpenMap to read a custom data format or perform any special rendering or processing.

CHAPTER 8

Points of Interest

Points of Interest are any collection geographic locations to be displayed on the map independent of any incidents or releases in the current project. The framework for POI display is in the `mil.dtra.hpac.client.poi` package, beginning with the `POIGroup` interface.

8.1 POI GROUP FRAMEWORK

8.1.1 POIGroup

`POIGroup`'s methods begin with `init()`, which is passed a root or base URL for any URLs specifying data for the group. The `name` property with associated accessor methods is assigned during configuration, described below. The `createMapOverlay()` method dictates that a POI group must create a single `MapOverlay` instance for representing or displaying the individual POIs in the group. The overlay will be sized to fit the map display viewport, or the size of the `MapPane` in the Project Editor.

If there are few POIs in the group, on the order of a couple of dozen, the overlay can act as a container with a `JComponent` for each POI. However, if there are more than a couple of dozen, it is best to render the representation of the POIs (e.g., an icon or label) from the `MapOverlay` itself.

8.1.2 SimplePOIGroup and SimplePOIOverlay

Other classes provided in the `mil.dtra.hpac.client.poi` framework package aid in implementing `POIGroup`. `SimplePOIGroup` is a class implementing `POIGroup`. Generally, it should be overridden, although it has useful functionality to be inherited, and it also can be instantiated. Its `createMapOverlay()` method returns an instance of `SimplePOIOverlay`, described below. An extension can override this method to return a different `MapOverlay` extension. Further, it adds the `dataURL` property for specifying a single file containing the group's data or individual POI records. The class method `readGroups()` is called during initialization of the Project Editor to read POI group configurations, described in Section 8.2.

`AbstractPOIOverlay` extends `MapOverlay` to provide basic behavior necessary for an overlay returned from a group's `createMapOverlay()` method. Specifically, it forces its size to be that of its parent. `SimplePOIOverlay()` extends `AbstractPOIOverlay` to create a `POIIcon` instance for each POI read from the group data file. An extension of `SimplePOIOverlay()` may override

the `createPOIIcon()` method to return instances of a `POIIcon` extension. The `PointOfInterest` parameter passed to `createPOIIcon()` is a minimal representation of a POI as a name and a coordinate. It can also be extended to include additional information.

`SimplePOIOverlay`'s `init()` method calls the class method `PointOfInterest.read()` passing an `InputStream` for a URL, the `dataURL` from the `SimplePOIGroup`. Each `PointOfInterest` object read is passed to the `createPOIIcon()` method to create an icon. Thus, a `SimplePOIOverlay` extension can override `init()` to read instances of a `PointOfInterest` extension and pass them to an override of `createMapIcon()`. Of course, one could extend `AbstractPOIOverlay` directly and customize the entire process of reading and representing POI data.

8.1.3 IncidentPOIOverlay and IncidentPOIIcon

An extension of `SimplePOIOverlay` is provided in `IncidentPOIOverlay`. Its purpose is to represent POIs with icons such that are drop sites for incident definition drags. Those drop site icons are instances of `IncidentPOIIcon`. The `ReactorPOIGroup` in the `POI` framework package is a `SimplePOIGroup` extension that represents nuclear reactors in an `IncidentPOIOverlay`. It allowed any incident definition to be dragged over an icon representing a reactor to create an incident at the reactor's location.

Note `ReactorPOIGroup` is not used in the Project Editor configuration to represent nuclear reactors. An alternative POI group implementation, `mil.dtra.hpac.client.poi.ReactorsPOIGroup`, was selected by the HPAC Integrator to display more reactor information via the popup menu for reactor icons. It does not extend `ReactorPOIGroup` and does not provide drop site functionality for icons on its own. Nonetheless, `ReactorPOIGroup` is available for extension as part of the `POI` framework.

8.2 POI GROUP CONFIGURATION

The `hpac.properties` file in the `client` / subdirectory under the HPAC installation includes the `hpac.poiGroups` key. It is list property with subkeys starting with `0` to specify individual POI groups. The entries from the installed `hpac.properties` are listed below.

```
hpac.poiGroups.0.class=mil.dtra.hpac.client.poi.WeatherPOIGroup
hpac.poiGroups.0.dataURL=data/wx/wmo.stn
hpac.poiGroups.0.name=Weather Stations
```

```
hpac.poiGroups.1.class=mil.dtra.hpac.client.poi.ReactorsPOIGroup
hpac.poiGroups.1.dataURL=data/nfac/reactors.stn
hpac.poiGroups.1.name=Nuclear Reactors
```

```
hpac.poiGroups.2.class=mil.dtra.hpac.client.poi.SfcClimoPOIGroup
hpac.poiGroups.2.dataURL=data/wx/sfcclimo.idx
hpac.poiGroups.2.name=Surface Climatology Stations
```

Three groups are specified as instances of `WeatherPOIGroup`, `ReactorsPOIGroup`, and `SfcClimoPOIGroup`, respectively. Each also includes `dataURL` and `name` properties, just as defined

for `SimplePOIGroup`. The relative URLs point to files containing the data for the groups, and the names are used in Project Editor overlay menus.

Any implementation of `POIGroup` may be specified via the `class` property. The other properties which must be specified are defined by the `POIGroup` class itself and must be read in the properties deserialization method `readProps()` (refer to Chapter 9).

CHAPTER 9

Properties Serialization and Deserialization

Like most applications, HPAC needs to save and load projects. An HPAC project is merely an object containing other objects comprising the state of the project as defined and edited by the user. In object-oriented terminology, saving an external representation of an object is referred to as *object persistence*, and the process of writing and reading an object's representation is called *serialization* and *deserialization*. (Reading and writing are often referred to collectively as *serialization*.) Rather than have the top level object, in this case the HPAC project, responsible for saving and loading a representation of all its contents, each component object shares the responsibility by (de)serializing its own state. Java provides binary object serialization for classes tagged with the `java.io.Serializable` interface, but a binary representation requires Java code to view or edit the serialized state of the project.

In order for project files need to be human readable and suited for manipulation by other tools, HPAC employs an extensible serialization framework based on Java properties files and the `java.util.Properties` class. An abbreviated class diagram is shown in Figure 9.1.

9.1 PROPERTIES SERIALIZATION FRAMEWORK

The serialization framework consists of the `PropsSerializer` and `Parser` interfaces, the `AbstractPropsSerializer` abstract base class, and the `ValueProperties` class. Almost all the implementation details are encapsulated in the `ValueProperties` class. `Parser` and implementations for commonly used core Java classes are in the `mil.dtra.util.parsers` package, and the other three are in `mil.dtra.util`. All of them are in distributed in `dtrautil.jar`.

`ValueProperties` relies upon `java.beans.Introspector` to determine an object's properties by referencing a `BeanInfo` provided for the object or using Java's reflection mechanism. An object's properties are converted to a string representation in one of two ways.

9.1.1 String Property Representation

First, an object may be represented by a single string. In this case a `Parser` implementation is necessary and must be registered for the class in a `ValueProperties` object. `ValueProperties` includes parsers for common Java core classes such as `java.awt.Dimension` and `java.awt.Point`. A snippet from an HPAC project file follows.

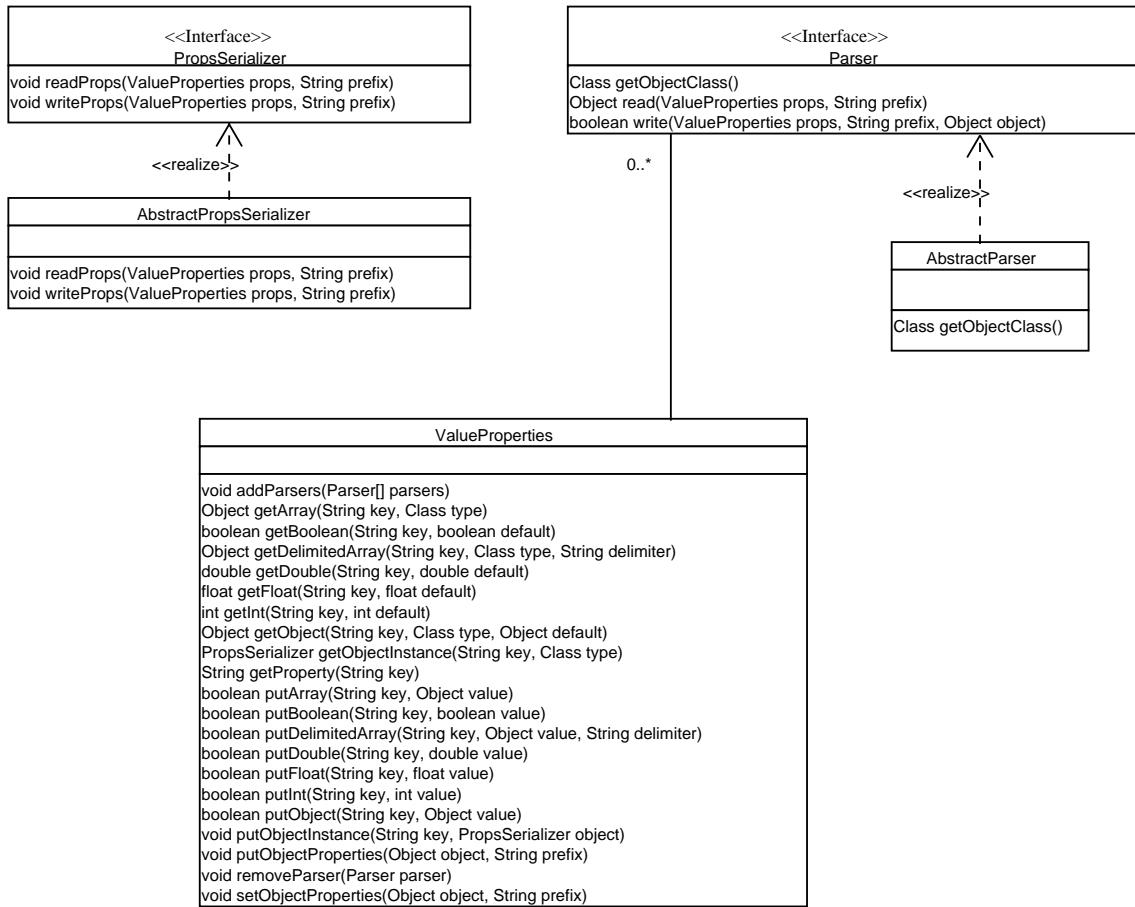


Figure 9.1: Properties serialization framework.

```

URL=file\:/home/re7/.hpactest/nfac1.hpac
class=mil.dtra.hpac.data.Project
limits.class=mil.dtra.hpac.data.project.Limits
limits.maxGridCellsPerSurface=25000
limits.maxMetHorzSize=1000
limits.maxPuffs=20000
maxTimeStep=900.0
objectSet.0.class=mil.dtra.hpac.models.nfac.client.Nfac
objectSet.0.incident.ID=incident-1013192452738
objectSet.0.incident.class=mil.dtra.hpac.data.Incident
objectSet.0.incident.coord=-80.43389129638672,40.621944427490234
objectSet.0.incident.hasCustomMaterials=false
objectSet.0.incident.hasCustomReleases=false
objectSet.0.incident.location.class=mil.dtra.hpac.data.LLALocation
objectSet.0.incident.location.value=-80.43,40.62,0.0

```

Single string representation works well for classes with a small number of properties that are of primitive types like int, boolean, or double. Refer to the “objectSet.0.incident.coord” and “objectSet.0.incident.location.value” lines above. Properties of these objects are delimited with

commas in a single value.

9.1.2 Hierarchical Object Property Representation

Second, an object may be represented as a hierarchy of properties and sub-objects with each leaf property stored as a single string. This is illustrated by the limits and “objectSet. 0. incident” objects in the listing above. The limits object has three properties, *maxGridCellsPerSurface*, *maxMethOrSize*, and *maxPuffs*, each of type *int* and represented as a separate key or line in the file. A period separates the property names. For example, “limits. *maxGridCellsPerSurface*” represents the *maxGridCellsPerSurface* property of the *limits* property of the project object represented by the listing.

The *incident* property is an example of a hierarchical object. Of the five properties illustrated above, three are of primitive types: *ID*, *hasCustomMaterials*, and *hasCustomReleases*. The other two properties, *coord* and *location*, are themselves objects. Finally, the project object itself is a hierarchical object. In the listing we see its *URL*, *limits*, *maxTimeStep*, and *objectSet* properties.

An object capable of serializing itself must implement the `PropsSerializer` interface, analogous to implementing `Serializable` for Java binary serialization. `PropsSerializer` specifies two methods, `readProps()` for deserialization and `writeProps()` for serialization. Hierarchical objects must implement `PropsSerializer`. Further, there are two serialization modes in this framework, *implicit* and *explicit*.

9.1.3 Implicit Serialization

Implicit serialization relies on introspection to recognize the properties of an object, either via public getter and setter accessor methods as per the JavaBean pattern or a `BeanInfo` associated with the object. The return or parameter type class specified in the accessor is checked first to see if it implements `PropsSerializer`, in which case the parameter’s `readProps()` or return value’s `writeProps()` method is called for deserialization or serialization, respectively. Otherwise, the class is checked against registered parsers. If a match is found the parser is used to convert the object to or from a string representation. In the absence of a parser, the class is compared against the primitive types which are handled internally. Finally, the object is ignored if no information about its type can be determined.

This is analogous to the implicit way fields not tagged as `transient` are serialized with Java’s binary mechanism.

9.1.4 Explicit Serialization

Explicit serialization relies upon the class implementer to make explicit calls in `readProps()` and `writeProps()` methods to deserialize or serialize the object’s properties. This is analogous to providing `readObject()` and `writeObject()` methods for binary serialization. A class with no other superclasses can extend `AbstractPropsSerializer` and inherit implicit serialization. Any class can invoke the implicit mechanisms by calling the `setObjectProperties()` and `putObjectProperties()` methods of the `ValueProperties` instance for implicit deserialization and serialization, respectively.

9.1.5 Arrays

There are two forms of array processing, delimited arrays and indexed arrays. For the former, element values are delimited by a specified string, which can be defaulted to a comma. Note that some object representations use delimiters between values as well, so delimited arrays must be used carefully. Also bear in mind that a `java.util.StringTokenizer` instance is used to parse delimited arrays. Thus, blank values cannot be used since successive delimiter characters are treated as a single delimiter during the parse. Note `ValueProperties` has two `getDelimitedArray()` and `putDelimitedArray()` methods each, one specifying the delimiter, and the other defaulting to comma.

The indexed array methods, `getArray()` and `putArray()` use a *length* property to determine the number of objects in the array and zero-based index values for the individual objects. For example, a `Color[]` property named `colors` with 4 elements would appear as follows if `colors` were a property of the top level object:

```
colors.length=4
colors.0=255,0,0
colors.1=0,255,0
colors.2=0,0,255
colors.3=255,255,255
```

`ValueProperties`' `getObject()` and `putObject()` methods call `getDelimitedArray()` and `putDelimitedArray()`, respectively, when implicitly processing an array property value. There are string size limitations when storing `Properties` objects to files, so classes with large arrays must explicitly (de)serialize array properties using the indexed methods `putArray()` and `getArray()`.

9.2 PLUGGING INTO THE SERIALIZATION FRAMEWORK

This framework allows an organization to extend HPAC via one the frameworks it provides. An organization desiring to provide their own incident source model, map display, or POI group must have a means of serializing properties of an object. For incident source models, the incident description becomes part of the HPAC project.

Extension classes may implement `PropsSerializer` and methods `readProps()` and `writeProps()`. Classes such as `IncidentModel` force an extender to inherit the responsibility for explicit serialization. Others allow an extender to supplant implicit serialization with explicit serialization. A new class that does not extend an existing class with an established serialization approach is free to implement `PropsSerializer` implicitly or explicitly, or a `Parser` implementation may be provided and registered with `ValueProperties` if an object can be represented in a single string.

Bibliography

- [1] Transmission Control Protocol, DARPA Internet Program Protocol Specification, RFC 793, Defense Advanced Research Projects Agency, Arlington, Virginia, September, 1981
- [2] Java™ 2 Platform, Standard Edition (J2SE™) Version 1.3.1,
<http://java.sun.com/j2se/1.3/docs/index.html>
- [3] Java™ 2 Platform, Standard Edition (J2SE™) Version 1.4.0,
<http://java.sun.com/j2se/1.4/docs/index.html>
- [4] Erich Gamma et al., Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1995, ISBN: 0201633612.
- [5] SCIPUFF Dispersion Model,
<http://www.titan.com/appliedtech/Pages/TRT/pages/scipuff/scipuff.htm>.
- [6] R.I. Sykes, C.P. Cerasoli and D.S. Henn, "The representation of dynamic flow effects in a Lagrangian puff dispersion model", J. Haz. Mat., 64, 223-247, 1999.
- [7] R.I. Sykes and R.S. Gabruk, "A second-order closure model for the effect of averaging time on turbulent plume dispersion", J. Appl. Met., 36, 165-184, 1997.
- [8] R.I. Sykes, D.S. Henn, S.F. Parker and R.S. Gabruk, "SCIPUFF - A generalized hazard dispersion model", Ninth Joint Conference on the Applications of Air Pollution Meteorology with A&WMA, American Met. Soc., 1996.
- [9] The Common Object Request Broker: Architecture and Specification, Revision 2.0, Object Management Group, Inc., July, 1995.
- [10] The Common Object Request Broker: Architecture and Specification, Revision 2.3, Object Management Group, Inc., June, 1999.
- [11] IDL to Java Language Mapping Specification, formal, 99-07-03, Object Management Group, Inc., June, 1999.
- [12] OpenMap™ Open Systems Mapping Technology,
<http://openmap.bbn.com>
- [13] Stephen F. Parker, The HPAC Application Programming Interface HPAC Version 4.0 (Draft), Titan Research and Technology, Titan Corp., Princeton, NJ, December 2000.
- [14] IHPACServer CORBA Interface Design Description for HPAC 4.0, Version 1.2, WMDOAM-IDD-01-U-R3C0, Science Applications International Corporation, San Diego, CA, April 2002.

Appendix A

Sample HPACtool Program

This is an example of calls to the HPACtool Windows DLL. It contains a pair of C++ classes, a C++ main program, a Makefile, an `hpac.ini` file (required for a run), and the output from a run. The `hpactool.h` file referenced in the source is the same as the one listed in Section 2.2. The `server bin` subdirectory under the HPAC installation (typically `c:\hpac4`) must be in the path to pick up `HPACtool.dll` and other DLLs.

The program was compiled and linked using Microsoft Visual C++ version 6.0 and tested under Windows XP/Professional and Windows 2000.

A.1 HPACFunctions.h

Definitions of HPACtool functions and some C++ operators invoked in the example.

```
#ifndef __HPACFunctions__
#define __HPACFunctions__
//*********************************************************************
*      NAME:          HPACFunctions.h
*****
#include <windows.h>
#include <iostream>

#include "hpactool.h"

using namespace std;

//*********************************************************************
* )      CLASS:          HPACFunctions
* )      PURPOSE:
* )          Dynamically loads HPACtool.dll and gets references to
* )          the functions we want to call
*****
class
HPACFunctions
{
    // Types
```

```

//



typedef int          (*Function)(...);




// Object Attributes
//


protected:

HINSTANCE           fDllHandle;

Function            fContourField,
                    fCreateField,
                    fCreateNewProject,
                    fExitTool,
                    fGetContourCount,
                    fGetDefaultInput,
                    fGetLastError,
                    fGetNumPlotClasses,
                    fGetNumPlotTimes,
                    fGetPlotClasses,
                    fGetPlotTimes,
                    fGetVersion,
                    fInitTool,
                    fRunProject;




// Object Methods
//


public:

/*********************************************
*      METHOD:          HPACFunctions()          *
*      PURPOSE:          *
*                      Default constructor making the LoadLibrary() and   *
*                      GetProcAddress() calls for HPACtool.dll functions       *
********************************************/


HPACFunctions();



/*********************************************
*      METHOD:          ~HPACFunctions()          *
*      PURPOSE:          *
*                      Destructor. Calls FreeLibrary() to release           *
*                      HPACtool.dll                                     *
********************************************/


virtual             ~HPACFunctions();



/*********************************************
*      METHOD:          ContourField()           *
*      PURPOSE:          *
********************************************/

```

```

*           Wrapper for HPACtool's ContourField() function      *
***** */
inline int          ContourField(
    int             caller_id,
    int             grid_id,
    HPACPlotFieldT & field,
    HPACPlotTypeT & plot_type,
    HPACContourHeaderT & contour_head,
    HPACContourElementT * contour_list,
    int             mode,
    HPACLineT *     lines,
    HPACPointT *    points
) const
{
return
fContourField(
    &caller_id, &grid_id,
    &field, &plot_type,
    &contour_head, contour_list,
    &mode, lines, points
);
}

/*****
*   METHOD:          CreateField()                                *
*   PURPOSE:         *                                         *
*   Wrapper for HPACtool's CreateField() function                *
***** */
inline int          CreateField(
    int             caller_id,
    HPACPlotFieldT & field,
    float *         class_data
) const
{
return
fCreateField( &caller_id, &field, class_data );
};

/*****
*   METHOD:          CreateNewProject()                           *
*   PURPOSE:         *                                         *
*   Wrapper for HPACtool's NewProject() function                 *
*   (Method names should be verbs:)                            *
***** */
inline int          CreateNewProject(
    int             caller_id,
    CreateNewT &   project,
    MaterialT *    materials,
    ReleaseT *     releases
) const
{
}

```

```

        return
fCreateNewProject(
    &caller_id, &project, materials, releases
);
}

/*****************/
*      METHOD:          ExitTool()                      *
*      PURPOSE:          *                                *
*                  Wrapper for HPACtool's ExitTool() function   *
*                                *                                *
inline int          ExitTool() const
{
    return fExitTool();
}

/*****************/
*      METHOD:          GetContourCount()                *
*      PURPOSE:          *                                *
*                  Wrapper for HPACtool's ContourCount() function   *
*      (Method names should be verbs:)                   *
*                                *                                *
inline int          GetContourCount(
    int             caller_id,
    int             grid_id,
    HPACPlotFieldT & field,
    HPACPlotTypeT & plot_type,
    HPACContourHeaderT & contour_head,
    HPACContourElementT * contour_list,
    int             mode,
    int &           line_count,
    int &           point_count
) const
{
    return
fGetContourCount(
    &caller_id, &grid_id,
    &field, &plot_type,
    &contour_head, contour_list,
    &mode, &line_count, &point_count
);
}

/*****************/
*      METHOD:          GetDefaultInput()               *
*      PURPOSE:          *                                *
*                  Wrapper for HPACtool's DefaultInput() function   *
*      (Method names should be verbs:)                   *
*                                *                                *
inline int          GetDefaultInput(

```

```

        int          caller_id,
        int          request_type,
        void *      out0,
        void *      out1
    ) const
{
    return
    fGetDefaultInput(
        &caller_id, &request_type,
        out0, out1
    );
}

/*****************/
*   METHOD:      GetLastError()           *
*   PURPOSE:     *
*           Wrapper for HPACtool's GetLastError() function   *
/*****************/
inline int      GetLastError( MessageT & error ) const
{
    return  fGetLastError( &error );
}

/*****************/
*   METHOD:      GetNumPlotClasses()      *
*   PURPOSE:     *
*           Wrapper for HPACtool's NumPlotClasses() function   *
*           (Method names should be verbs:)                      *
/*****************/
inline int      GetNumPlotClasses(
    int          caller_id,
    const ProjectIDT & project_id,
    int &         class_count,
    int &         choice_count,
    int &         kind_count
) const
{
    return
    fGetNumPlotClasses(
        &caller_id, &project_id,
        &class_count, &choice_count, &kind_count
    );
}

/*****************/
*   METHOD:      GetNumPlotTimes()       *
*   PURPOSE:     *
*           Wrapper for HPACtool's NumPlotTimes() function   *
*           (Method names should be verbs:)                     *
/*****************/

```

```

inline int      GetNumPlotTimes(
                int                  caller_id,
                const ProjectIDT & project_id,
                int &               puff_time_count,
                int &               surface_time_count,
                int &               met_time_count,
                int &               rad_surf_grid_sub_times
                ) const
{
    return
    fGetNumPlotTimes(
        &caller_id, &project_id,
        &puff_time_count, &surface_time_count,
        &met_time_count, &rad_surf_grid_sub_times
        );
}

/*****************
*   METHOD:      GetPlotClasses()          *
*   PURPOSE:     *
*           Wrapper for HPACtool's GetPlotClasses() function  *
*****************/
inline int      GetPlotClasses(
                int                  caller_id,
                const ProjectIDT & project_id,
                Char64T *            class_strings,
                Char64T *            choice_strings,
                Char64T *            kind_strings,
                HPACCCategoryClassT * cat_class_array,
                HPACClassChoiceT *   class_choice_array,
                HPACFieldCoordinateT & project_coord
                ) const
{
    return
    fGetPlotClasses(
        &caller_id, &project_id,
        class_strings, choice_strings, kind_strings,
        cat_class_array, class_choice_array,
        &project_coord
        );
}

/*****************
*   METHOD:      GetPlotTimes()          *
*   PURPOSE:     *
*           Wrapper for HPACtool's GetPlotTimes() function  *
*****************/
inline int      GetPlotTimes(
                int                  caller_id,
                const ProjectIDT & project_id,
                HPACTimeT *          puff_times,

```

```

        HPACTimeT *          surface_times,
        HPACTimeT *          met_times
    ) const
{
return
fGetPlotTimes(
    &caller_id, &project_id,
    puff_times, surface_times, met_times
);
};

/***** METHOD:           GetVersion() *****
* PURPOSE:              *
*           Wrapper for HPACtool's GetVersion() function *
***** inline int          GetVersion() const
{
return fGetVersion();
};

/***** METHOD:           InitTool() *****
* PURPOSE:              *
*           Wrapper for HPACtool's InitTool() function *
***** inline int          InitTool(
    int                  caller_id,
    HPACC callbackProc   callback,
    int                  request,
    const LimitT &       limits,
    const Char128T &     ini_file
) const
{
return
fInitTool(
    &caller_id, &callback, &request,
    &limits, &ini_file
);
};

/***** METHOD:           RunProject() *****
* PURPOSE:              *
*           Wrapper for HPACtool's RunProject() function *
***** inline int          RunProject( int caller_id, const PEndT & params ) const
{
return fRunProject( &caller_id, &params );
};

```

```

}; // HPACFunctions

/*
*      FUNCTION:          _AssignCString()
*      PURPOSE:
*              Free function for copying a Fortran string to a C
*              string
*/
void          _AssignCString(
                  char * to, int size, const char * from
                );

/*
*      MACRO:            AssignCString()
*      PURPOSE:
*              Calls _AssignCString() using sizeof() for the 2nd param
*/
#ifndef AssignCString
#define AssignCString( C, F ) \
{ \
    _AssignCString( (C), sizeof(C), (F) ); \
}
#endif

/*
*      FUNCTION:          _AssignFString()
*      PURPOSE:
*              Free function for copying a C string to a Fortran
*              string
*/
void          _AssignFString(
                  char * to, int size, const char * from
                );

/*
*      MACRO:            AssignFString()
*      PURPOSE:
*              Calls _AssignFString() using sizeof() for the 2nd param
*/
#ifndef AssignFString
#define AssignFString( F, C ) \
{ \
    _AssignFString( (F), sizeof(F), (C) ); \
}
#endif

/*
*      FUNTION:          operator <<
*/

```

```

*      PURPOSE: *
*          Formatted output of an AuditT *
***** */
ostream &           operator <<( ostream &, const AuditT & );

/* **** */
*      FUNCTION:     operator <<
*      PURPOSE: *
*          Formatted output of a Char128T *
***** */
ostream &           operator <<( ostream &, const Char128T & );

/* **** */
*      FUNCTION:     operator <<
*      PURPOSE: *
*          Formatted output of a DomainT *
***** */
ostream &           operator <<( ostream &, const DomainT & );

/* **** */
*      FUNCTION:     operator <<
*      PURPOSE: *
*          Formatted output of a FlagsT *
***** */
ostream &           operator <<( ostream &, const FlagsT & );

/* **** */
*      FUNCTION:     operator <<
*      PURPOSE: *
*          Formatted output of a MessageT *
***** */
ostream &           operator <<( ostream &, const MessageT & );

/* **** */
*      FUNCTION:     operator <<
*      PURPOSE: *
*          Formatted output of an OptionsT *
***** */
ostream &           operator <<( ostream &, const OptionsT & );

/* **** */
*      FUNCTION:     operator <<
*      PURPOSE: *
*          Formatted output of a ReferenceT *
***** */
ostream &           operator <<( ostream &, const ReferenceT & );

```

```

/****************************************************************************
*      FUNCTION:          operator <<
*      PURPOSE:
*              Formatted output of a SpatialT
****************************************************************************/
ostream &           operator <<( ostream &, const SpatialT & );



/****************************************************************************
*      FUNCTION:          operator <<
*      PURPOSE:
*              Formatted output of a TemporalT
****************************************************************************/
ostream &           operator <<( ostream &, const TemporalT & );


/****************************************************************************
*      FUNCTION:          operator <<
*      PURPOSE:
*              Formatted output of a TimeT
****************************************************************************/
ostream &           operator <<( ostream &, const TimeT & );


#endif

```

A.2 HPACFunctions.cpp

Implementation of the HPACFunctions class.

```

/****************************************************************************
*      NAME:          HPACFunctions.cpp
****************************************************************************/
#include <windows.h>
#include <memory.h>

#include "HPACFunctions.h"


/****************************************************************************
*      METHOD:        HPACFunctions::HPACFunctions()
****************************************************************************/
HPACFunctions::HPACFunctions() :
    fDllHandle( NULL ),
    fContourField( NULL ),
    fCreateField( NULL ),
    fCreateNewProject( NULL ),
    fExitTool( NULL ),
    fGetContourCount( NULL ),
    fGetDefaultInput( NULL ),
    fGetLastError( NULL ),

```

```

fGetNumPlotClasses( NULL ),
fGetNumPlotTimes( NULL ),
fGetPlotClasses( NULL ),
fGetPlotTimes( NULL ),
fGetVersion( NULL ),
fInitTool( NULL ),
fRunProject( NULL )
{
    // Load DLL
    //
    if ( (fDllHandle = LoadLibrary( "HPACtool" )) == NULL )
        throw "[HPACFunctions] error loading HPACtool.dll";

    // Load Functions
    //
else if (
    (fContourField = (Function)
     GetProcAddress( fDllHandle, "HPACCONTOURFIELD" )) == NULL
    )
    throw "[HPACFunctions] error mapping HPACContourField()";

else if (
    (fCreateField = (Function)
     GetProcAddress( fDllHandle, "HPACCREATEFIELD" )) == NULL
    )
    throw "[HPACFunctions] error mapping HPACCCreateField()";

else if (
    (fCreateNewProject = (Function)
     GetProcAddress( fDllHandle, "HPACNEWPROJECT" )) == NULL
    )
    throw "[HPACFunctions] error mapping HPACNewProject()";

else if (
    (fExitTool = (Function)
     GetProcAddress( fDllHandle, "HPACEXITTOOL" )) == NULL
    )
    throw "[HPACFunctions] error mapping HPACExitTool()";

else if (
    (fGetContourCount = (Function)
     GetProcAddress( fDllHandle, "HPACCONTOURCOUNT" )) == NULL
    )
    throw "[HPACFunctions] error mapping HPACContourCount()";

else if (
    (fGetLastError = (Function)
     GetProcAddress( fDllHandle, "HPACGETLASTERROR" )) == NULL
    )
    throw "[HPACFunctions] error mapping HPACGetLastError()";

else if (
    (fGetNumPlotClasses = (Function)

```

```

        GetProcAddress( fDllHandle, "HPACNUMPLOTCLASSES" ) ) == NULL
    )
throw "[HPACFunctions] error mapping HPACNumPlotClasses()";

else if (
    (fGetNumPlotTimes = (Function)
     GetProcAddress( fDllHandle, "HPACNUMPLOTTIMES" ) ) == NULL
    )
throw "[HPACFunctions] error mapping HPACNumPlotTimes()";

else if (
    (fGetPlotClasses = (Function)
     GetProcAddress( fDllHandle, "HPACGETPLOTCLASSES" ) ) == NULL
    )
throw "[HPACFunctions] error mapping HPACGetPlotClasses()";

else if (
    (fGetPlotTimes = (Function)
     GetProcAddress( fDllHandle, "HPACGETPLOTTIMES" ) ) == NULL
    )
throw "[HPACFunctions] error mapping HPACGetPlotTimes()";

else if (
    (fGetVersion = (Function)
     GetProcAddress( fDllHandle, "HPACGETVERSION" ) ) == NULL
    )
throw "[HPACFunctions] error mapping HPACGetVersion()";

else if (
    (fGetDefaultInput = (Function)
     GetProcAddress( fDllHandle, "HPACDEFAULTINPUT" ) ) == NULL
    )
throw "[HPACFunctions] error mapping HPACDefaultInput()";

else if (
    (fInitTool = (Function)
     GetProcAddress( fDllHandle, "HPACINITTOOL" ) ) == NULL
    )
throw "[HPACFunctions] error mapping HPACInitTool()";

else if (
    (fRunProject = (Function)
     GetProcAddress( fDllHandle, "HPACRUNPROJECT" ) ) == NULL
    )
throw "[HPACFunctions] error mapping HPACRunProject()";
} // HPACFunctions::HPACFunctions

/**************************************************************************
*      METHOD:          HPACFunctions::~HPACFunctions()                  *
* **************************************************************************/
HPACFunctions::~HPACFunctions()
{

```

```

if ( fDllHandle != NULL )
    FreeLibrary( fDllHandle );
} // HPACFunctions::~HPACFunctions()

/**************************************************************************
*      NAME:          _AssignCString()                               *
*************************************************************************/
void
.AssignCString( char * to, int size, const char * from )
{
    register char * ptr;

memset( to, ' ', size );
strncpy( to, from, size - 1 );
for ( ptr = to + size - 2; *ptr == ' ' && ptr > to; ptr-- ) ;
*(ptr + 1) = 0;
} // _AssignCString

/**************************************************************************
*      FUNCTION:        _AssignFString()                            *
*************************************************************************/
void
.AssignFString( char * to, int size, const char * from )
{
    int len = strlen( from );

memcpy( to, from, len );
memset( to + len, ' ', size - len );
} // _AssignFString

/**************************************************************************
*      FUNCTION:        operator <<                           *
*************************************************************************/
ostream &
operator <<( ostream & output, const AuditT & audit )
{
    char title[ 80 ], analyst[ 32 ], class_name[ 32 ], version[ 32 ], date[ 32 ];

AssignCString( title, audit.title );
AssignCString( analyst, audit.analyst );
AssignCString( class_name, audit.className );
AssignCString( version, audit.version );
AssignCString( date, audit.date );

return
output <<
    "title=" << title << ",analyst=" << analyst <<
    ",className=" << class_name << ",version=" << version <<
    ",date=" << date;
} // operator <<

```

```

*****
*      FUNCTION:          operator <<
*****
ostream &
operator <<( ostream & output, const Char128T & value )
{
    char the_str[ 128 ];

_AssignCString( the_str, sizeof(the_str), value.string );
return output << the_str;
} // operator <<

*****
*      FUNCTION:          operator <<
*****
ostream &
operator <<( ostream & output, const DomainT & domain )
{
return
output <<
    "map=" << domain.map << ",zoneUtm=" << domain.zoneUtm <<
    ",bounds=" << domain.xMin << "," << domain.yMin << "," <<
    domain.xMax << "," << domain.yMax << ",res=" <<
    domain.hRes << "," << domain.vRes;
} // operator <<

*****
*      FUNCTION:          operator <<
*****
ostream &
operator <<( ostream & output, const FlagsT & flags )
{
return
output <<
    "start=" << flags.start << ",method=" << flags.method <<
    ",mode=" << flags.mode << ",prjEffects=" << flags.prjEffects <<
    ",audit=(" << flags.audit << ")");
} // operator <<

*****
*      FUNCTION:          operator <<
*****
ostream &
operator <<( ostream & output, const MessageT & message )
{
    char astr[ 128 ], bstr[ 128 ], cstr[ 128 ], routine[ 80 ];

_AssignCString( astr, sizeof(astr), message.aString );

```

```

_AssignCString( bstr, sizeof(bstr), message.bString );
_AssignCString( cstr, sizeof(cstr), message.cString );
_AssignCString( routine, sizeof(routine), message.routine );

return
output <<
    " i=" << message.iParm << ", j=" << message.jParm << endl <<
    " a: '" << astr << "'" << endl <<
    " b: '" << bstr << "'" << endl <<
    " c: '" << cstr << "'" << endl <<
    " routine: '" << routine << "'" << endl;
} // operator <<

//*****************************************************************************
*      FUNCTION:          operator <<
//*****************************************************************************
ostream &
operator <<( ostream & output, const OptionsT & options )
{
    char sampler_file[ 64 ], sampler_path[ 128 ];

AssignCString( sampler_file, options.samplerFile );
AssignCString( sampler_path, options.samplerPath );

return
output <<
    "nzBL=" << options.nzBL << ",mGrd=" << options.mGrd <<
    ",substrate=" << options.substrate <<
    ",timeAvg=" << options.timeAvg <<
    ",massMin=" << options.massMin <<
    ",delMin=" << options.delMin <<
    ",wwTrop=" << options.wwTrop <<
    ",epsTrop=" << options.epsTrop <<
    ",slTrop=" << options.slTrop <<
    ",uuCalm=" << options.uuCalm <<
    ",slCalm=" << options.slCalm <<
    ",zDosage=" << options.zDosage <<
    ",dtSampler=" << options.dtSampler <<
    ",samplerFile=" << sampler_file <<
    ",samplerPath=" << sampler_path;
} // operator <<

//*****************************************************************************
*      FUNCTION:          operator <<
//*****************************************************************************
ostream &
operator <<( ostream & output, const ReferenceT & ref )
{
return
output <<
    "x=" << ref.x << ",y=" << ref.y <<

```

```

    ",lat=" << ref.lat << ",lon=" << ref.lon;
} // operator <<

/************************************************************************
*      FUNCTION:          operator <<
*****
ostream &
operator <<( ostream & output, const SpatialT & spatial )
{
return
output << "[" << spatial.domain << "] [ " << spatial.reference << "]";
} // operator <<

/************************************************************************
*      FUNCTION:          operator <<
*****
ostream &
operator <<( ostream & output, const TemporalT & value )
{
return
output <<
    "start.zone=" << value.start.zone << endl <<
    "start.time=" << value.start.time << endl <<
    "end.time=" << value.end.time << endl <<
    "end.step=(" << value.end.step.max << "," << value.end.step.output << ")" <<
    endl;
} // operator <<

/************************************************************************
*      FUNCTION:          operator <<
*****
ostream &
operator <<( ostream & output, const TimeT & value )
{
return
output <<
    value.reference << "," << value.year << "," << value.month << "," <<
    value.day << "," << value.hour << "," << value.runTime;
} // operator <<

```

A.3 HPACDefaults.h

Definition of an `HPACDefaults` class for establishing defaults for inputs to `HPACtool`. Some defaults are retrieved from `HPACtool` and customized as necessary. A release and material are created from scratch.

```
#ifndef __HPACDefaults__
#define __HPACDefaults__
*****
```

```

*      NAME:          HPACDefaults.h
***** */
#include "HPACFunctions.h"

#ifndef MAX
#define MAX( A, B )      ((A) > (B) ? (A) : (B))
#endif

#ifndef MIN
#define MIN( A, B )      ((A) < (B) ? (A) : (B))
#endif

/*
*)      CLASS:          HPACDefaults
*)      PURPOSE:        Encapsulates the retrieval and/or creation of defaults
*)      suitable for input to HPACtool
*/
class
HPACDefaults
{
    // Constants
    //
public:
    const static char * PLOT_CATEGORY_NAMES[ ];

protected:
    enum
    {
        MAX INCIDENTS = 3,
        MAX MATERIALS = 1,
        MAX RELEASES = 5,
    };

    // Object Attributes
    //
protected:
    CreateNewT           fCreateNewDef;
    PEndT                fEnd;
    LimitT               fLimits;
    MaterialT            fMaterials[ MAX_MATERIALS ];

```

```

ReleaseT           fReleases[ MAX_RELEASES ];

           // Object Methods
           //

public:

/***** METHOD:      HPACDefaults() *****
* PURPOSE:
*   Default constructor. Makes GetDefaultInput() calls,
*   creates, and initializes HPACtool inputs. The ProjectIDT is
*   composed from the parameters.;
***** */

    HPACDefaults(
        const HPACFunctions &,
        int                  caller_id,
        int                  id,
        const char *         project_name,
        const char *         project_path,
        const char *         hpac_dir = "c:\\hpac4"
    );

/***** METHOD:      ~HPACDefaults() *****
* PURPOSE:
*   Destructor
***** */

//virtual          ~HPACDefaults();

           // Protected Methods
           //

protected:

/***** METHOD:      InitMaterial() *****
* PURPOSE:
*   Fills the material object with values for ANTH, but
*   uses ANTH2 as the material name
***** */

void          InitMaterial(
    MaterialT &,
    const char * hpac_dir = "c:\\hpac4"
);

/***** METHOD:      InitRelease() *****
* PURPOSE:
***** */

```

```

*           Fills the release object with values for a continuous      *
*           release of the ANTH2 material                           *
***** */
void           InitRelease(
                ReleaseT &           release,
                const char *          material_name,
                float                 lon,
                float                 lat,
                float                 alt_meters
                );

/*****
*   METHOD:           InitWeather()                                *
*   PURPOSE:          initializes the weather object to specify 10 mph fixed   *
*                     winds from 180 degrees                         *
***** */
void           InitWeather( WeatherT & );

// Public Methods
//
public:

/*****
*   METHOD:           CreateProjectEnd()                            *
*   PURPOSE:          creates a PEndT object from the default project id and   *
*                     temporal domain suitable for input to RunProject().        *
***** */
inline
const PEndT &           CreateProjectEnd()
{
    fEnd.project = fCreateNewDef.project;
    fEnd.end = fCreateNewDef.input.time.end;
    return fEnd;
}

/*****
*   METHOD:           GetCreateNewDef()                            *
*   PURPOSE:          Returns a const reference to the default CreateNewT   *
*                     value.                                         *
***** */
inline
const CreateNewT &       GetCreateNewDef() const
{
    return fCreateNewDef;
}

```

```

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

*      METHOD:          GetCreateNewDef()           *  

*      PURPOSE:          *  

*                      Returns a reference to the default CreateNewT value.  *  

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

inline  

CreateNewT &          GetCreateNewDef()  

{  

    return fCreateNewDef;  

};  

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

*      METHOD:          GetFlags()                *  

*      PURPOSE:          *  

*                      Returns a const reference to the default FlagsT value.  *  

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

inline  

const FlagsT &        GetFlags() const  

{  

    return fCreateNewDef.input.flags;  

};  

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

*      METHOD:          GetFlags()                *  

*      PURPOSE:          *  

*                      Returns a reference to the default FlagsT value.  *  

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

inline  

FlagsT &              GetFlags()  

{  

    return fCreateNewDef.input.flags;  

};  

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

*      METHOD:          GetLimits()               *  

*      PURPOSE:          *  

*                      Returns a const reference to the default LimitT value.  *  

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

inline  

const LimitT &        GetLimits() const  

{  

    return fLimits;  

};  

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

*      METHOD:          GetLimits()               *  

*      PURPOSE:          *  

*                      Returns a reference to the default LimitT value.  *  


```

```

***** /
inline LimitT & GetLimits()
{
    return fLimits;
};

***** *
*   METHOD:          GetMaterials() *
*   PURPOSE:         *
*           Returns a const reference to the material array *
***** /
inline const MaterialT * GetMaterials() const
{
    return fMaterials;
};

***** *
*   METHOD:          GetMaterials() *
*   PURPOSE:         *
*           Returns a reference to the material array *
***** /
inline MaterialT * GetMaterials()
{
    return fMaterials;
};

***** *
*   METHOD:          GetOptions() *
*   PURPOSE:         *
*           Returns a const reference to the default OptionsT. *
***** /
inline const OptionsT & GetOptions() const
{
    return fCreateNewDef.input.options;
};

***** *
*   METHOD:          GetOptions() *
*   PURPOSE:         *
*           Returns a reference to the default OptionsT. *
***** /
inline OptionsT & GetOptions()
{
    return fCreateNewDef.input.options;
};

```

```

};

/*****
*      METHOD:          GetProjectID()                      *
*      PURPOSE:         Returns a const reference to the ProjectIDT value.   *
*****/
inline
const ProjectIDT &    GetProjectID() const
{
    return fCreateNewDef.project;
}

/*****
*      METHOD:          GetReleases()                     *
*      PURPOSE:        Returns a const reference to the Release array.       *
*****/
inline
const ReleaseT *     GetReleases() const
{
    return fReleases;
}

/*****
*      METHOD:          GetReleases()                     *
*      PURPOSE:        Returns a reference to the Release array.           *
*****/
inline
ReleaseT *           GetReleases()
{
    return fReleases;
}

/*****
*      METHOD:          GetSpatialDomain()                 *
*      PURPOSE:        Returns a const reference to the default SpatialT value *
*****/
inline
const SpatialT &    GetSpatialDomain() const
{
    return fCreateNewDef.input.domain;
}

/*****
*      METHOD:          GetSpatialDomain()                 *
*****/

```

```

*      PURPOSE: *
*          Returns a reference to the default SpatialT value *
***** */
inline
SpatialT &           GetSpatialDomain()
{
    return fCreateNewDef.input.domain;
}

***** */
*      METHOD:      GetTemporalDomain() *
*      PURPOSE:      *
*          Returns a const reference to the default TemporalT value*
***** */
inline
const TemporalT &   GetTemporalDomain() const
{
    return fCreateNewDef.input.time;
}

***** */
*      METHOD:      GetTemporalDomain() *
*      PURPOSE:      *
*          Returns a reference to the default TemporalT value      *
***** */
inline
TemporalT &          GetTemporalDomain()
{
    return fCreateNewDef.input.time;
}

***** */
*      METHOD:      GetWeather() *
*      PURPOSE:      *
*          Returns a const reference to the default WeatherT value *
***** */
inline
const WeatherT &     GetWeather() const
{
    return fCreateNewDef.weather;
}

***** */
*      METHOD:      GetWeather() *
*      PURPOSE:      *
*          Returns a reference to the default WeatherT value      *
***** */
inline
WeatherT &           GetWeather()

```

```

    {
        return fCreateNewDef.weather;
    };

/*
 *      METHOD:          SetReleaseCount()
 *
 *      PURPOSE:
 *          Sets the number of releases for the project.
 */
inline void SetReleaseCount( int count )
{
    fCreateNewDef.scnHead.number =
        MIN( count, MAX_RELEASES );
}

/*
 *      METHOD:          SetSpatialDomain()
 *
 *      PURPOSE:
 *          Sets the spatial domain.
 */
void SetSpatialDomain(
    float      west,
    float      south,
    float      east,
    float      north
);

/*
 *      METHOD:          SetTemporalDomain()
 *
 *      PURPOSE:
 *          Sets the temporal domain.
 */
void SetTemporalDomain(
    float      timezone_gmt_offset,
    int       year,
    int       month,
    int       day,
    float     hour,
    float     start_time_offset_hrs,
    float     run_time_hrs
);

} ; // HPACDefaults

#endif

```

A.4 HPACDefaults.cpp

Implementation of the HPACDefaults class.

```
***** NAME: HPACDefaults.cpp ****
#include <windows.h>
#include <memory.h>
#include <string.h>
#include <iostream>

#include "HPACDefaults.h"

using namespace std;

***** CONSTANT: HPACDefaults::PLOT_CATEGORY_NAMES ****
const char * HPACDefaults::PLOT_CATEGORY_NAMES[] =
{
    "Surface", "Surface Slice", "Horizontal Slice",
    "Vertically Integrated Slice", "Vertical Slice",
    "Table",
};

***** METHOD: HPACDefaults::HPACDefaults() ****
HPACDefaults::HPACDefaults()
{
    const HPACFunctions & funcs,
    int caller_id,
    int id,
    const char * project_name,
    const char * project_path,
    const char * hpac_dir
}

{
    // Limits must be created by hand
    //
    fLimits.puffs = 20000;
    fLimits.surfaceGrid = 25000;
    fLimits.met1D = 1000;

    // Initialize Project ID
    //
    fCreateNewDef.project.id = id;
    fCreateNewDef.project.version = funcs.GetVersion();
    AssignFString( fCreateNewDef.project.name, project_name );
    AssignFString( fCreateNewDef.project.path, project_path );
}
```

```

        // Retrieve Defaults
        //
PFlagsT pflags;
PMaterialT pmaterial;
POptionsT poptions;
PSpatialT pspatial;
PTemporalT pttime;

pflags.project = fCreateNewDef.project;
poptions.project = fCreateNewDef.project;
pspatial.project = fCreateNewDef.project;
ptime.project = fCreateNewDef.project;

pmaterial.project = fCreateNewDef.project;
pmaterial.mtlHead.max = MAX_MATERIALS;
pmaterial.mtlHead.number = 1;
pmaterial.control.mode = HC_SEARCH;
pmaterial.control.fileExtension[ 0 ] = 0;
AssignFString( pmaterial.control.searchName, "gb" );

        // Flags
        //
if (
    funcs.GetDefaultInput( caller_id, HPAC_FLAGS, &pflags, NULL ) !=
    HPACsuccess
)
throw "[HPACDefaults] error retrieving default flags";

fCreateNewDef.input.flags = pflags.flags;

        // Spatial Domain
        //
if (
    funcs.GetDefaultInput( caller_id, HPAC_DOMAIN, &pspatial, NULL ) !=
    HPACsuccess
)
throw "[HPACDefaults] error retrieving default spatial domain";

fCreateNewDef.input.domain = pspatial.spatial;
fCreateNewDef.input.domain.map = HD_LATLON;

        // Materials
        //
if (
    funcs.GetDefaultInput( caller_id, HPAC_MATERIAL, &pmaterial, fMaterials ) !=
    HPACsuccess
)
throw "[HPACDefaults] error retrieving default materials";

fCreateNewDef.input.mtlHead = pmaterial.mtlHead;
InitMaterial( fMaterials[ 0 ], hpac_dir );

```

```

        // Options
        //
if (
    funcs.GetDefaultInput( caller_id, HPAC_OPTIONS, &poptions, NULL ) !=
    HPACsuccess
)
throw "[HPACDefaults] error retrieving default options";

fCreateNewDef.input.options = poptions.options;

        // Temporal Domain
        //
if (
    funcs.GetDefaultInput( caller_id, HPAC_TIME, &ptime, NULL ) !=
    HPACsuccess
)
throw "[HPACDefaults] error retrieving default temporal domain";

fCreateNewDef.input.time = ptime.time;
fCreateNewDef.input.time.start.time.reference = HT_UTC;
fCreateNewDef.input.time.end.step.max = 300.0; // max step is 5 minutes
fCreateNewDef.input.time.end.step.output = 0.25; // output every 15 minutes

        // Max Release Count
        //
fCreateNewDef.scnHead.max = MAX_RELEASES;
fCreateNewDef.scnHead.number = 1;

        // Weather Definition
        //
InitWeather( fCreateNewDef.weather );

        // Release
        //
InitRelease( fReleases[ 0 ], "ANTH2", -80., 30., 10. );
} // HPACDefaults::HPACDefaults

/**************************************************************************
*      METHOD:          HPACDefaults::InitMaterial()                      *
***** */
void
HPACDefaults::InitMaterial( MaterialT & mat, const char * hpac_dir )
{
    char buffer[ 256 ];
    MatParticleT anthrax;

    anthrax.minConcentration = 0.0;
    anthrax.decayAmp = 1.67e-4;
    anthrax.decayMin = 1.67e-5;
    anthrax.NWPNDecay = 0.0;
    anthrax.save = HS_GROUPDEP | HS_GROUPDOS;
    anthrax.density = 500.0;
}

```

```

anthrax.nSizeBins = 1;
anthrax.binBounds[ 0 ] = 5.0e-6;
anthrax.binBounds[ 1 ] = 7.5e-6;
//anthrax.binSize[ 0 ] = anthrax.binBounds[ 1 ] - anthrax.binBounds[ 0 ];
anthrax.binSize[ 0 ] = anthrax.binBounds[ 0 ];

mat.type = HM_PARTICLE;
memcpy( &mat.matData, &anthrax, sizeof(anthrax) );
AssignFString( mat.name, "ANTH2" );
AssignFString( mat.units, "kg" );
AssignFString( mat.file, "anth_wet.mtl" );

sprintf( buffer, "%s\\server\\data\\materials", hpac_dir );
AssignFString( mat.path, buffer );
} // HPACDefaults::InitMaterial

***** NAME: HPACDefaults::InitRelease() *****
*****
void
HPACDefaults::InitRelease(
    ReleaseT & release,
    const char * material_name,
    float lon,
    float lat,
    float alt_meters
)
{
    RelContT cont_rel;

    cont_rel.distribution = HD_LOGNORM;
    cont_rel.rate = 100.0;
    cont_rel.duration = 4.0;
    cont_rel.sigY = 2.2;
    cont_rel.sigZ = 3.3;
    cont_rel.MMD = 5.0e-6;
    cont_rel.sigma = 1.01;
    cont_rel.momentum = 0.0;
    cont_rel.buoyancy = 0.0;
    cont_rel.dryFrac = 1.0;

    release.type = HR_CONT;
    release.status = HS_VALID;
    release.tRel = 0.0;
    release.xRel = lon;
    release.yRel = lat;
    release.zRel = alt_meters;
    release.horzUnc = 5.0;
    release.vertUnc = 1.0;
    //release.pa = 1.0;
    memcpy( &release.relData, &cont_rel, sizeof(cont_rel) );
    AssignFString( release.material, material_name );
}

```

```

AssignFString( release.relName, "continuous anthrax" );
AssignFString( release.relDisplay, "Continuous Anthrax" );
} // HPACDefaults::InitRelease

```

```

*****
*      METHOD:          HPACDefaults::InitWeather()           -
*****
void
HPACDefaults::InitWeather( WeatherT & wx )
{
wx.flags.reference = HD_LATLON;
wx.flags.doMC = HPACOff;
wx.flags.doOutput = HO_OUTPUT | HO_2D;
wx.flags.tOutput = -REAL_DEFAULT;
wx.flags.slHazard = 10;

wx.met.type = HW_METFIX;
wx.met.nnPrf =
wx.met.nnsfc = 0;
wx.met.timeBin = 3600.0;
wx.met.nStations =
wx.met.monthDay = 0;
wx.met.unitSpd = HU MPH;
wx.met.unitDir = HU_DEG;
wx.met.speed = 10.0;
wx.met.direction = 180.0;
wx.met.input[ 0 ][ 0 ] = 0;

wx.bl.type = HB_BLCALC;
wx.bl.wetness = HM_MSTNORM;
wx.bl.ziDay = 1000.0;
wx.bl.ziNight = 50.0;
wx.bl.hfluxDay = 50.0;
wx.bl.hfluxNight = 0.0;
wx.bl.canopy = 1.0;
wx.bl.roughness = 0.1;
wx.bl.albedo = 0.16;
wx.bl.bowen = 0.6;
wx.bl.cloud = 0.0;
wx.bl.canopyParam = 0.0;

wx.lsv.type = HL_LSVOPER;
wx.lsv.uu = 0.0;
wx.lsv.sl = 100.0;

wx.precip.type = HP_PRCPINP;
wx.precip.classType = 0;

wx.terrain.type = HT_SCIPUFF;
wx.terrain.mc.dtSWIFT = 0.0;
wx.terrain.mc.maxIter[ 0 ] =
wx.terrain.mc.maxIter[ 1 ] = 5;

```

```

wx.terrain.mc.eps[ 0 ] =
wx.terrain.mc.eps[ 1 ] = 1.0e-2;
wx.terrain.mc.alpha[ 0 ] = 1.0e-2;
wx.terrain.mc.alpha[ 1 ] = 1.0;
wx.terrain.mc.nz = 1;
wx.terrain.name[ 0 ] = 0;
wx.terrain.path[ 0 ] = 0;
} // HPACDefaults::InitWeather

/**************************************************************************
*      NAME:          HPACDefaults::SetSpatialDomain()                  *
***** /*****
void
HPACDefaults::SetSpatialDomain(
    float           west,
    float           south,
    float           east,
    float           north
)
{
fCreateNewDef.input.domain.domain.map = HD_LATLON;
fCreateNewDef.input.domain.domain.xMin = west;
fCreateNewDef.input.domain.domain.yMin = south;
fCreateNewDef.input.domain.domain.xMax = east;
fCreateNewDef.input.domain.domain.yMax = north;
} // HPACDefaults::SetSpatialDomain

/**************************************************************************
*      NAME:          HPACDefaults::SetTemporalDomain()                  *
***** /*****
void
HPACDefaults::SetTemporalDomain(
    float           timezone_gmt_offset,
    int             year,
    int             month,
    int             day,
    float           hour,
    float           start_time_offset_hrs,
    float           run_time_hrs
)
{
fCreateNewDef.input.time.start.zone = timezone_gmt_offset;
fCreateNewDef.input.time.start.time.reference = HT_UTC;
fCreateNewDef.input.time.start.time.year = year;
fCreateNewDef.input.time.start.time.month = month;
fCreateNewDef.input.time.start.time.day = day;
fCreateNewDef.input.time.start.time.hour = hour;
fCreateNewDef.input.time.start.time.runTime = start_time_offset_hrs;

fCreateNewDef.input.time.end.time = fCreateNewDef.input.time.start.time;
fCreateNewDef.input.time.end.time.hour += run_time_hrs;

```

```
fCreateNewDef.input.time.end.time.runTime = run_time_hrs;
} // HPACDefaults::SetTemporalDomain
```

A.5 example.cpp

Main program demonstrating call sequence to HPACtool.

```
***** NAME: example.cpp ****
* PURPOSE: Example code to exercise the HPACtool DLL *
***** */

#include <windows.h>
#include <memory.h>
#include <string.h>
#include <iostream>

#include "HPACFunctions.h"
#include "HPACDefaults.h"

using namespace std;

***** NAME: HandleHPACCalls() ****
* PURPOSE: This is the callback function for HPACtool to give
* feedback messages and ask for input during calculation and
* plotting.
***** */

int
__stdcall
HandleHPACCalls( int caller_id, int message_id, int param )
{
    static char clock_chars[] = { '|', '/', '-', '\\' };
    static char clock_count = 0;

    int status = HPACsuccess;

    switch ( message_id )
    {
        case HM_ERROR:
            cerr << endl << "[example] Error:" << endl << *((MessageT*)param);
            break;

        case HM_INFO:
            cerr << endl << "[example] Info:" << endl << *((MessageT*)param);
            break;

        case HM_PROGRESSMSG:
            cerr << endl << "[example] Progress" << endl << *((MessageT*)param);
    }
}
```

```

        break;

    case HM_REPLY:
        cerr << endl << "[example] Reply" << endl << *((MessageT*)param);
        status = HPACAffirmative;
        break;

    case HM_SETCLOCK:
        cerr << endl << clock_chars[ 0 ] << flush;
        clock_count = 0;
        break;

    case HM_STEPCLOCK:
        clock_count = (clock_count + 1) % sizeof(clock_chars);
        cerr << "\r" << clock_chars[ clock_count ] << flush;
        break;

    case HM_STOPCLOCK:
        cerr << endl;
        break;

    case HM_BUTTONSTATE:
    case HM_BUTTONTAG:
    case HM_CHECK:
    case HM_RELEASEWAIT:
    case HM_SETWAIT:
    case HM_START:
    case HM_STOP:
    default:
        break;
    } // switch

    return status;
} // HandleHPACCalls

static HPACCallbackProc _callback = HandleHPACCalls;

/**************************************************************************
*           FUNCTION:          main()                               *
*************************************************************************/
int
main( int argc, char * argv[] )
{
try
{
    char * root_dir = argc > 1 ? argv[ 1 ] : "c:\\hpac4";
    int caller_id = 0x1234;
    HPACFunctions hpac;
    HPACDefaults defaults( hpac, caller_id, 1, "example", "", root_dir );
    char current_dir[ 256 ], ini_path_str[ 512 ];
    Char128T ini_path;
    int status;
}

```

```

    // Get current directory for use as the project directory
    //
GetCurrentDirectory( sizeof(current_dir) - 1, current_dir );
sprintf( ini_path_str, "%s\\hpac.ini", current_dir );
AssignFString( ini_path.string, ini_path_str );
cerr << "[example] ini_path=" << ini_path << endl;

    // Initialize HPACtool
    //
cerr << "[example] DLL loaded, initializing tool..." << endl;
if (
    hpac.InitTool(
        caller_id, HandleHPACCalls, 1, defaults.GetLimits(), ini_path
    ) == HPACfailure
)
cerr << "[example] InitTool() failed" << endl;

else
{
    cerr << "[example] InitTool() succeeded" << endl;

        // Set input locations
        //
cerr << "[example] setting spatial domain" << endl;
defaults.SetSpatialDomain( -90., 30., -80., 35. );

    cerr << "[example] setting temporal domain" << endl;
    defaults.SetTemporalDomain( -4., 2002, 9, 27, 13.3, 0., 4. );

    cerr << "[example] setting release location" << endl;
    ReleaseT * releases = defaults.GetReleases();
releases[ 0 ].tRel = 0.0;
releases[ 0 ].xRel = -85.0;
releases[ 0 ].yRel = 32.5;
releases[ 0 ].zRel = 10.0;

        // Create new project
        //
cerr << "[example] creating new project" << endl;
if (
    hpac.CreateNewProject(
        caller_id, defaults.GetCreateNewDef(),
        defaults.GetMaterials(),
        defaults.GetReleases()
    ) != HPACsuccess
)
{
    MessageT message;
hpac.GetLastError( message );
cerr << "[example] CreateNewProject() failed" << endl << message;
}

```

```

        // Run project
        //
else if (
    hpac.RunProject( caller_id, defaults.CreateProjectEnd() ) != HPACsuccess
)
{
    MessageT message;
    hpac.GetLastError( message );
    cerr << "[example] error running project" << endl << message;
}

        // On success, get plot data
        //
else
{
    int
        class_count, choice_count, kind_count,
        puff_time_count, surface_time_count,
        met_time_count, rad_time_count;
    Char64T * class_names = NULL;
    Char64T * choice_names = NULL;
    Char64T * kind_names = NULL;
        // 2D arrays are column major from Fortran
    HPACCATEGORYCLASST * cat_classes = NULL; // nclass x HP_NUMCAT
    HPACCLASSCHOICET * class_choices = NULL; // nchoice x nclass
    HPACFIELDCOORDINATET project_coord;
    HPACTIMET * puff_times = NULL;
    HPACTIMET * surface_times = NULL;
    HPACTIMET * met_times = NULL;

    cerr <<
        endl << "[example] Calculation Complete!" <<
        endl << "[example] retrieving plot information" <<
        endl;

        // Get plot class/choice/kind counts so we'll
        // know how big to allocate arrays
        //
if (
    hpac.GetNumPlotClasses(
        caller_id, defaults.GetProjectID(),
        class_count, choice_count, kind_count
    ) != HPACsuccess
)
    cerr << "[example] error getting plot class/choice/kind counts" << endl;

        // Get plot time counts for array sizing
        //
else if (
    hpac.GetNumPlotTimes(
        caller_id, defaults.GetProjectID(),
        puff_time_count, surface_time_count,
        met_time_count, rad_time_count

```

```

        ) != HPACsuccess
    )
cerr << "[example] error getting plot time counts" << endl;

        // Allocate arrays
        //
else if (
    (class_names = new Char64T[ class_count ]) == NULL ||
    (choice_names = new Char64T[ choice_count ]) == NULL ||
    (kind_names = new Char64T[ kind_count ]) == NULL ||
    (cat_classes =
        new HPACCATEGORYCLASST[ class_count * HP_NUMCAT ]) == NULL ||
    (class_choices =
        new HPACCLASSCHOICET[ choice_count * class_count ]) == NULL ||
    (puff_times = new HPACTIMET[ puff_time_count ]) == NULL ||
    (surface_times = new HPACTIMET[ surface_time_count ]) == NULL ||
    (met_times = new HPACTIMET[ met_time_count ]) == NULL
)
cerr << "[example] error allocating plot data buffers" << endl;

        // Get plot class/choice/kind data
        //
else if (
    hpac.GetPlotClasses(
        caller_id, defaults.GetProjectID(),
        class_names, choice_names, kind_names,
        cat_classes, class_choices, project_coord
    ) != HPACsuccess
)
cerr << "[example] error getting plot class/choice/kind data" << endl;

        // Get plot times
        //
else if (
    hpac.GetPlotTimes(
        caller_id, defaults.GetProjectID(),
        puff_times, surface_times, met_times
    ) != HPACsuccess
)
cerr << "[example] error getting plot times" << endl;

        // Plot info retrieved
        //
else
{
    // Dump it for funzies
    //
    int i, j;

    cout <<
        endl << "[PLOT INFO]" << endl <<
        " class_count=" << class_count <<

```

```

    ", choice_count=" << choice_count <<
    ", kind_count=" << kind_count <<
    endl;

    cout << " class names:" << endl;
    for ( i = 0; i < class_count; i++ )
    {
        char name[ 64 ];
        AssignCString( name, class_names[ i ].string );
        cout << " " << name << endl;
    }

    cout << " choice names:" << endl;
    for ( i = 0; i < choice_count; i++ )
    {
        char name[ 64 ];
        AssignCString( name, choice_names[ i ].string );
        cout << " " << name << endl;
    }

    cout << " kind names:" << endl;
    for ( i = 0; i < kind_count; i++ )
    {
        char name[ 64 ];
        AssignCString( name, kind_names[ i ].string );
        cout << " " << name << endl;
    }

    cout << " puff times:" << endl;
    for ( i = 0; i < puff_time_count; i++ )
    {
        char name[ 24 ];
        AssignCString( name, puff_times[ i ].string );

        cout <<
            " " << puff_times[ i ].time <<
            ", " << puff_times[ i ].nItems <<
            ", " << name <<
            endl;
    }

    cout << " surface times:" << endl;
    for ( i = 0; i < surface_time_count; i++ )
    {
        char name[ 24 ];
        AssignCString( name, surface_times[ i ].string );

        cout <<
            " " << surface_times[ i ].time <<
            ", " << surface_times[ i ].nItems <<
            ", " << name <<
            endl;
    }
}

```

```

        // Remember, column major
        //
cout << " category-class availability:" << endl;
for ( i = 0; i < HP_NUMCAT; i++ )
{
    cout <<
        "      category " <<
        HPACDefaults::PLOT_CATEGORY_NAMES[ i ] <<
    endl;

for ( j = 0; j < class_count; j++ )
{
    HPACCCategoryClassT * ptr = cat_classes + (j * HP_NUMCAT) + i;
    char name[ 64 ];

AssignCString( name, class_names[ j ].string );
cout <<
        "      " << name <<
        ": avail=" << ptr->available << ", type=" <<
        ptr->type <<
        endl;
}
}

// Remember, column major
//
cout << " class-choice availability:" << endl;
for ( i = 0; i < class_count; i++ )
{
    char class_name[ 64 ];
AssignCString( class_name, class_names[ i ].string );

for ( j = 0; j < choice_count; j++ )
{
    char name[ 64 ], units[ 16 ];
    HPACClassChoiceT * ptr = class_choices + (j * class_count) + i;
    char * units_addr = ptr->units;

AssignCString( name, choice_names[ j ].string );
AssignCString( units, ptr->units );
cout <<
        "      " << class_name << "-" << name <<
        ": available=" << ptr->available <<
        ", kind=" << ptr->kind <<
        ", ikind=" << ptr->ikind <<
        ", nkind=" << ptr->nkind <<
        ", itime=" << ptr->itime <<
        ", usertime=" << ptr->usertime <<
        ", units=" << units <<
    endl;
}
}

// Set inputs for a call to CreateField()
//

```

```

const ProjectIDT & project_id = defaults.GetProjectID();
int sag_id;
HPACPlotFieldT plot_field;
HPACPlotTypeT plot_type;
HPACContourHeaderT contour_header;
HPACContourElementT contour_element[ 3 ];
HPACLineT * lines;
HPACPointT * points;
int
    line_count, point_count,
    mode = CLOSE_CONTOUR + LATLON_OUTPUT;
float class_data[ 1 ];

plot_field.category = HP_SURF;
plot_field.theclass = 1;
plot_field.choice = 1;
plot_field.kind = 0;
plot_field.timeID = 1;
plot_field.userTime =
    surface_times[ surface_time_count - 1 ].time.runTime;
plot_field.hazard = HPACfalse;
plot_field.maxCells = 0;
plot_field.maxLevel = -1;
memset( &plot_field.project, ' ', sizeof(plot_field.project) );
memcpy(
    &plot_field.project, &project_id.name, sizeof(project_id.name)
);
memcpy( &plot_field.path, &project_id.path, sizeof(project_id.path) );

plot_type.type = HP_MEAN;
plot_type.data = 0;
plot_type.areaMode = HP_ON;
                                // Contour values are from the
                                // ANTH material file
contour_header.number = 3;
contour_header.scale = 16667.0;
contour_header.labelXMode = PLOT_ON;
contour_header.drawMode = PLOT_LIN;
AssignFString( contour_header.unit, "mg-min/m3" );

contour_element[ 0 ].contour = 7.5e-5;
AssignFString( contour_element[ 0 ].label, "ICt10" );
contour_element[ 1 ].contour = 5.9e-4;
AssignFString( contour_element[ 1 ].label, "ICt50" );
contour_element[ 2 ].contour = 4.7e-3;
AssignFString( contour_element[ 2 ].label, "ICt90" );

                                // Create a field
                                //
if (
    (sag_id =
        hpac.CreateField( caller_id, plot_field, class_data ) ) < 0
)

```

```

{
    MessageT message;
    hpac.GetLastError( message );
    cerr << "[example] error creating plot field" << endl << message;
}

        // Succeeded, so we can contour it
        // Get counts for array sizing
        //
else if (
    hpac.GetContourCount(
        caller_id, sag_id,
        plot_field, plot_type,
        contour_header, contour_element, mode,
        line_count, point_count
    ) != HPACsuccess
)
cerr << "[example] error counting plot field contours" << endl;

        // Allocate contour polygon arrays
        //
else if (
    (lines = new HPACLineT[ line_count ]) == NULL ||
    (points = new HPACPPointT[ point_count ]) == NULL
)
cerr << "[example] error allocating contour data" << endl;

        // Contour the field
        //
else if (
    hpac.ContourField(
        caller_id, sag_id,
        plot_field, plot_type,
        contour_header, contour_element, mode,
        lines, points
    ) != HPACsuccess
)
cerr << "[example] error contouring plot field" << endl;

        // Succeeded, dump the polygon data
        //
else
{
    int pt_index;

cout <<
endl << "[countour lines] line_count=" <<
line_count << ", point_count=" << point_count <<
endl;

for ( i = 0; i < line_count; i++ )
{
    cout <<

```

```

        i << ":" index=" << lines[ i ].index <<
        ", start=" << lines[ i ].start <<
        ", number=" << lines[ i ].number <<
        ", mode=" << lines[ i ].mode <<
        endl;

    for (
        j = 0, pt_index = lines[ i ].start - 1;
        j < lines[ i ].number;
        j++, pt_index++
    )
    {
        cout <<
            " " << points[ pt_index ].x <<
            ", " << points[ pt_index ].y <<
            endl;
    }
} // for each line
} // field created
} // else plot info data retrieved
} // else run successful, retrieve plot

        // Exit the tool and reclaim that memory!
        //
if ( hpac.ExitTool() != HPACsuccess )
{
    MessageT message;
    hpac.GetLastError( message );
    cerr << "[example] error exiting HPACtool" << endl << message;
}
} // else HPACtool initialized
} // try

catch ( const char * ex )
{
    cerr << "[example] exception:" << endl << ex << endl;
}

return 0;
} // main

```

A.6 Makefile

```

#-----
#-      NAME:          example::Makefile
#-----
MAKE=nmake -nologo

#-----
#-      Project Macros
#-----

```

```

PROCS= \
    example.exe

EX_SRC= \
    HPACFunctions.cpp \
    HPACDefaults.cpp \
    example.cpp

#-----
#-      Command Macros
#-----
CXX= cl

LD= link

#-----
#-      Compiler Macros
#-----
CFLAGS= \
    -GA -GX -O2
#    -GA -Yd
DEFINES=
IDIRS= \
    -I.

CPPFLAGS= \
    $(CFLAGS) $(DEFINES) $(IDIRS)

#-----
#-      Linker Macros
#-----
LDOPTS= \
    /machine:ix86 /subsystem:console
LIBS= \
    kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib \
    advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib \
    odbccp32.lib

LDFLAGS= \
    $(LDOPTS) $(LIBS)

#-----
#-      Derived Macros
#-----
EX_OBJS= \
    $(EX_SRC:.cpp=.obj)

```

```

#-----
#-      Targets
#-----
all:    $(PROCS)

clean:
    del *.obj *.res

objs:   $(EX_OBJS)

example.exe:    $(EX_OBJS)
               $(LD) /out:$@ $(EX_OBJS) $(LDFLAGS)

#-----
#-      Rules
#-----
.cpp.obj:
    $(CXX) -c $(CPPFLAGS) $<

```

A.7 hpac.ini

Sample configuration file for HPACtool. It must be customized to match a particular HPAC installation.

```

[Paths]
HPACDbgDir=D:\users\re7\src\hpactooltest
HPACBinDir=C:\hpac4\server\bin
HPACDataDir=c:\hpac4\server\data
SciDataDir=C:\hpac4\server\data\scipuff
HPACTempDir=C:\users\re7\src\hpactooltest\temp
SfcClimoDir=C:\hpac4\server\data\sfcclimo,F:\sfccclimo
UaClimoDir=C:\hpac4\server\data\uaclimo,F:\uaclimo
3DMetDir=C:\hpac4\server\data\uaclimo,F:\uaclimo
hastempdir=D:\users\re7\src\hpactooltest\temp
PopDir=C:\hpac4\server\data\populate,F:\populate
etacmetdir=C:\hpac4\server\data\sfcclimo,F:\sfccclimo
3DCompressedmetdir=D:\users\re7\src\hpactooltest\temp

[Land_Use]
LandUseDataFile=C:\hpac4\server\data\scipuff\landuse.dat

[FxTool]
Library1=FxCoda.dll
Library2=Obscure.dll
[Currency]
RipdlipiVer=2.00
HPACtoolVersion=4.0.1
SCIPUFF_Version=1.605

```

```
LandUse_Version=0.1
SYStool_Version=0.104
```

A.8 run.log

Standard error log messages produced from execution of `example` with no command-line arguments. Note if HPAC is not installed in the directory `c:\hpac4`, the installation directory must be specified as the first command line argument to `example`.

```
[example] ini_path=D:\users\re7\src\hpactooltest\hpac.ini
[example] DLL loaded, initializing tool...

[example] Info:
  i=0, j=0
  a: 'Unable to locate path'
  b: 'entry=3Dmetdir : path=C:\hpac4\server\data\uaclimo,F:\uaclimo'
  c: 'Upper Air Climatology will not be available'
  routine: 'HPACInitTool'
[example] InitTool() succeeded
[example] setting spatial domain
[example] setting temporal domain
[example] setting release location
[example] creating new project

[example] Progress
  i=0, j=0
  a: 'Preparing to create project'
  b: 'Project=example'
  c: 'Validating input data'
  routine: 'Progress'

[example] Progress
  i=0, j=0
  a: 'Preparing to create project'
  b: 'Project=example'
  c: 'Deleting existing project files'
  routine: 'Progress'

[example] Progress
  i=0, j=0
  a: 'Creating project input files'
  b: 'Project=example'
  c: 'Writing INP file'
  routine: 'Progress'

[example] Progress
  i=0, j=0
  a: ''
  b: ''
  c: 'Writing SCN file'
  routine: 'Progress'
```

```
[example] Progress
i=0, j=0
a: ''
b: ''
c: 'Writing MSC file'
routine: 'Progress'

[example] Progress
i=0, j=0
a: 'SCIPUFF starting creation of project files'
b: 'Project=example'
c: ''
routine: 'Progress'

[example] Progress
i=0, j=0
a: 'Initializing SCIPUFF'
b: ''
c: ''
routine: 'Progress'

[example] Progress
i=0, j=0
a: 'example : Initializing'
b: ''
c: ''
routine: 'Progress'

[example] Progress
i=0, j=0
a: 'example : Initializing'
b: 'Reading project input data'
c: ''
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: 'Initializing data'
c: ''
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: 'Validating meteorology input'
c: ''
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: ''
```

```

c: 'Initializing meteorology parameters'
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: ''
c: 'Reading meteorology input file'
routine: 'Progress'

[example] Progress
i=0, j=0
a: 'example : Initializing'
b: 'Validating meteorology input'
c: 'Preparing meteorology'
routine: 'Progress'

[example] Progress
i=0, j=0
a: 'example : Initializing'
b: ''
c: ''
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: 'Creating project output files'
c: ''
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: ''
c: 'Creating surface output files'
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: ''
c: 'Creating surface deposition file'
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: ''
c: 'Creating surface dosage file'
routine: 'Progress'

[example] Progress

```

```

i=0, j=0
a: ''
b: ''
c: 'Creating project puff file'
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: ''
c: 'Writing output file(s)'
routine: 'Progress'

[example] Progress
i=0, j=0
a: 'example : Initialized'
b: ''
c: ''
routine: 'Progress'

[example] Progress
i=0, j=0
a: 'SCIPUFF finished creating project files'
b: 'Project=example'
c: ''
routine: 'Progress'

[example] Progress
i=0, j=0
a: 'SCIPUFF starting dispersion calculation'
b: 'Project=example'
c: ''
routine: 'Progress'

[example] Progress
i=0, j=0
a: 'Initializing SCIPUFF'
b: ''
c: ''
routine: 'Progress'

[example] Progress
i=0, j=0
a: 'example : Initializing'
b: ''
c: ''
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: 'Reading project data files'
c: ''

```

```

routine: 'Progress'

[example] Progress
  i=0, j=0
  a: ''
  b: 'Initializing data'
  c: ''
  routine: 'Progress'

[example] Progress
  i=0, j=0
  a: ''
  b: 'Initializing meteorology'
  c: ''
  routine: 'Progress'

[example] Progress
  i=0, j=0
  a: ''
  b: ''
  c: 'Initializing meteorology parameters'
  routine: 'Progress'

[example] Progress
  i=0, j=0
  a: ''
  b: ''
  c: 'Reading meteorology input file'
  routine: 'Progress'

[example] Progress
  i=0, j=0
  a: 'example : Initializing'
  b: 'Initializing meteorology'
  c: 'Preparing meteorology'
  routine: 'Progress'

[example] Progress
  i=0, j=0
  a: 'example : Initializing'
  b: 'Initializing meteorology'
  c: 'Continuing meteorology preparation'
  routine: 'Progress'

[example] Progress
  i=0, j=0
  a: 'example : Initializing'
  b: ''
  c: ''
  routine: 'Progress'

[example] Progress
  i=0, j=0

```

```

a: 'example : Initializing'
b: ''
c: ''
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: 'Initializing sources'
c: ''
routine: 'Progress'

|
[example] Progress
i=0, j=0
a: ''
b: ''
c: 'Preparing source      1 (Continuous Anthrax )'
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: ''
c: 'Initializing 1 new releases'
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: 'Beginning run with      0 puffs'
c: ''
routine: 'Progress'

[example] Progress
i=0, j=0
a: 'example : Initialized'
b: ''
c: ''
routine: 'Progress'

|
[example] Progress
i=0, j=0
a: 'example : Calculating'
b: '27-SEP-02 13:18:00Z (0.0 hr)'
c: '      0 Puffs  0.586s timestep'
routine: 'Progress'

```

```
|  
|  
[example] Progress  
i=0, j=0  
a: 'example : Calculating'  
b: '27-SEP-02 13:23:00Z (5.00 min)'  
c: '    30 Puffs  9.38s timestep'  
routine: 'Progress'
```

```
|  
|  
[example] Progress  
i=0, j=0  
a: 'example : Calculating'  
b: '27-SEP-02 13:28:00Z (10.0 min)'  
c: '    39 Puffs  9.38s timestep'  
routine: 'Progress'
```

```
[example] Progress  
i=0, j=0  
a: 'example : Writing output'  
b: ''  
c: ''  
routine: 'Progress'
```

```
[example] Progress  
i=0, j=0  
a: ''  
b: ''  
c: 'Writing puff file'  
routine: 'Progress'
```

```
[example] Progress  
i=0, j=0  
a: ''  
b: ''  
c: 'Writing project file'  
routine: 'Progress'
```

```
[example] Progress  
i=0, j=0  
a: ''  
b: ''  
c: 'Writing surface file(s)'  
routine: 'Progress'
```

```
[example] Progress  
i=0, j=0  
a: ''
```

```

b: ''
c: ''
routine: 'Progress'

|
| [example] Progress
i=0, j=0
a: 'example : Calculating'
b: '27-SEP-02 13:33:00Z (15.0 min)'
c: '    43 Puffs  9.38s timestep'
routine: 'Progress'

|
| [example] Progress
i=0, j=0
a: 'example : Calculating'
b: '27-SEP-02 13:38:00Z (20.0 min)'
c: '    50 Puffs  18.8s timestep'
routine: 'Progress'

|
| [example] Progress
i=0, j=0
a: 'example : Calculating'
b: '27-SEP-02 13:43:00Z (25.0 min)'
c: '    54 Puffs  18.8s timestep'
routine: 'Progress'

[example] Progress
i=0, j=0
a: 'example : Writing output'
b: ''
c: ''
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: ''
c: 'Writing puff file'
routine: 'Progress'

[example] Progress
i=0, j=0

```

```

a: ''
b: ''
c: 'Writing project file'
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: ''
c: 'Writing surface file(s)'
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: ''
c: ''
routine: 'Progress'

|
| [example] Progress
| i=0, j=0
| a: 'example : Calculating'
| b: '27-SEP-02 13:48:00Z (30.0 min)'
| c: '    54 Puffs 18.8s timestep'
| routine: 'Progress'

|
| [example] Progress
| i=0, j=0
| a: 'example : Calculating'
| b: '27-SEP-02 13:53:00Z (0.583 hr)'
| c: '    51 Puffs 18.8s timestep'
| routine: 'Progress'

|
| [example] Progress
| i=0, j=0
| a: 'example : Calculating'
| b: '27-SEP-02 13:58:00Z (0.667 hr)'
| c: '    57 Puffs 18.8s timestep'
| routine: 'Progress'

[example] Progress

```

```

i=0, j=0
a: 'example : Writing output'
b: ''
c: ''
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: ''
c: 'Writing puff file'
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: ''
c: 'Writing project file'
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: ''
c: 'Writing surface file(s)'
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: ''
c: ''
routine: 'Progress'

|
| [example] Progress
| i=0, j=0
| a: 'example : Calculating'
| b: '27-SEP-02 14:03:00Z (0.750 hr)'
| c: '    58 Puffs 18.8s timestep'
| routine: 'Progress'

|
| [example] Progress
| i=0, j=0
| a: 'example : Calculating'
| b: '27-SEP-02 14:08:00Z (0.833 hr)'
| c: '    60 Puffs 18.8s timestep'

```

```

routine: 'Progress'

|
| [example] Progress
|   i=0, j=0
|   a: 'example : Calculating'
|   b: '27-SEP-02 14:13:00Z (0.917 hr)'
|   c: '    66 Puffs 18.8s timestep'
|   routine: 'Progress'

[example] Progress
  i=0, j=0
  a: 'example : Writing output'
  b: ''
  c: ''
  routine: 'Progress'

[example] Progress
  i=0, j=0
  a: ''
  b: ''
  c: 'Writing puff file'
  routine: 'Progress'

[example] Progress
  i=0, j=0
  a: ''
  b: ''
  c: 'Writing project file'
  routine: 'Progress'

[example] Progress
  i=0, j=0
  a: ''
  b: ''
  c: 'Writing surface file(s)'
  routine: 'Progress'

[example] Progress
  i=0, j=0
  a: ''
  b: ''
  c: ''
  routine: 'Progress'

|
| [example] Progress

```

```

i=0, j=0
a: 'example : Calculating'
b: '27-SEP-02 14:18:00Z (1.00 hr)'
c: '    60 Puffs 18.8s timestep'
routine: 'Progress'

|
| [example] Progress
| i=0, j=0
| a: 'example : Calculating'
| b: '27-SEP-02 14:23:00Z (1.08 hr)'
| c: '    66 Puffs 18.8s timestep'
| routine: 'Progress'

|
| [example] Progress
| i=0, j=0
| a: 'example : Calculating'
| b: '27-SEP-02 14:28:00Z (1.17 hr)'
| c: '    68 Puffs 18.8s timestep'
| routine: 'Progress'

[example] Progress
i=0, j=0
a: 'example : Writing output'
b: ''
c: ''
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: ''
c: 'Writing puff file'
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: ''
c: 'Writing project file'
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''

```

```

b: ''
c: 'Writing surface file(s)'
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: ''
c: ''
routine: 'Progress'

|
| [example] Progress
| i=0, j=0
| a: 'example : Calculating'
| b: '27-SEP-02 14:33:00Z (1.25 hr)'
| c: '    66 Puffs 18.8s timestep'
| routine: 'Progress'

|
| [example] Progress
| i=0, j=0
| a: 'example : Calculating'
| b: '27-SEP-02 14:38:00Z (1.33 hr)'
| c: '    68 Puffs 18.8s timestep'
| routine: 'Progress'

|
| [example] Progress
| i=0, j=0
| a: 'example : Calculating'
| b: '27-SEP-02 14:43:00Z (1.42 hr)'
| c: '    73 Puffs 18.8s timestep'
| routine: 'Progress'

[example] Progress
i=0, j=0
a: 'example : Writing output'
b: ''
c: ''
routine: 'Progress'

[example] Progress
i=0, j=0

```

```

a: ''
b: ''
c: 'Writing puff file'
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: ''
c: 'Writing project file'
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: ''
c: 'Writing surface file(s)'
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: ''
c: ''
routine: 'Progress'

|
| [example] Progress
| i=0, j=0
| a: 'example : Calculating'
| b: '27-SEP-02 14:48:00Z (1.50 hr)'
| c: '    74 Puffs 18.8s timestep'
| routine: 'Progress'

|
| [example] Progress
| i=0, j=0
| a: 'example : Calculating'
| b: '27-SEP-02 14:53:00Z (1.58 hr)'
| c: '    71 Puffs 18.8s timestep'
| routine: 'Progress'

|
| [example] Progress
| i=0, j=0

```

```

a: 'example : Calculating'
b: '27-SEP-02 14:58:00Z (1.67 hr)'
c: '    69 Puffs 18.8s timestep'
routine: 'Progress'

[example] Progress
i=0, j=0
a: 'example : Writing output'
b: ''
c: ''
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: ''
c: 'Writing puff file'
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: ''
c: 'Writing project file'
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: ''
c: 'Writing surface file(s)'
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: ''
c: ''
routine: 'Progress'

|
|
[example] Progress
i=0, j=0
a: 'example : Calculating'
b: '27-SEP-02 15:03:00Z (1.75 hr)'
c: '    68 Puffs 18.8s timestep'
routine: 'Progress'

```

```

|
[example] Progress
  i=0, j=0
  a: 'example : Calculating'
  b: '27-SEP-02 15:08:00Z (1.83 hr)'
  c: '    78 Puffs 18.8s timestep'
  routine: 'Progress'

|
[example] Progress
  i=0, j=0
  a: 'example : Calculating'
  b: '27-SEP-02 15:13:00Z (1.92 hr)'
  c: '    76 Puffs 18.8s timestep'
  routine: 'Progress'

[example] Progress
  i=0, j=0
  a: 'example : Writing output'
  b: ''
  c: ''
  routine: 'Progress'

[example] Progress
  i=0, j=0
  a: ''
  b: ''
  c: 'Writing puff file'
  routine: 'Progress'

[example] Progress
  i=0, j=0
  a: ''
  b: ''
  c: 'Writing project file'
  routine: 'Progress'

[example] Progress
  i=0, j=0
  a: ''
  b: ''
  c: 'Writing surface file(s)'
  routine: 'Progress'

[example] Progress
  i=0, j=0
  a: ''
  b: ''

```

```

c: ''
routine: 'Progress'

|
| [example] Progress
|   i=0, j=0
|   a: 'example : Calculating'
|   b: '27-SEP-02 15:18:00Z (2.00 hr)'
|   c: '    82 Puffs 18.8s timestep'
|   routine: 'Progress'

|
| [example] Progress
|   i=0, j=0
|   a: 'example : Calculating'
|   b: '27-SEP-02 15:23:00Z (2.08 hr)'
|   c: '    87 Puffs 18.8s timestep'
|   routine: 'Progress'

|
| [example] Progress
|   i=0, j=0
|   a: 'example : Calculating'
|   b: '27-SEP-02 15:28:00Z (2.17 hr)'
|   c: '    87 Puffs 18.8s timestep'
|   routine: 'Progress'

[example] Progress
i=0, j=0
a: 'example : Writing output'
b: ''
c: ''
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: ''
c: 'Writing puff file'
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''

```

```

b: ''
c: 'Writing project file'
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: ''
c: 'Writing surface file(s)'
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: ''
c: ''
routine: 'Progress'

|
| [example] Progress
| i=0, j=0
| a: 'example : Calculating'
| b: '27-SEP-02 15:33:00Z (2.25 hr)'
| c: '    85 Puffs 18.8s timestep'
| routine: 'Progress'

|
| [example] Progress
| i=0, j=0
| a: 'example : Calculating'
| b: '27-SEP-02 15:38:00Z (2.33 hr)'
| c: '    86 Puffs 18.8s timestep'
| routine: 'Progress'

|
| [example] Progress
| i=0, j=0
| a: 'example : Calculating'
| b: '27-SEP-02 15:43:00Z (2.42 hr)'
| c: '    89 Puffs 18.8s timestep'
| routine: 'Progress'

[example] Progress
i=0, j=0

```

```

a: 'example : Writing output'
b: ''
c: ''
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: ''
c: 'Writing puff file'
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: ''
c: 'Writing project file'
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: ''
c: 'Writing surface file(s)'
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: ''
c: ''
routine: 'Progress'

|
| [example] Progress
| i=0, j=0
| a: 'example : Calculating'
| b: '27-SEP-02 15:48:00Z (2.50 hr)'
| c: '    87 Puffs 18.8s timestep'
| routine: 'Progress'

|
| [example] Progress
| i=0, j=0
| a: 'example : Calculating'
| b: '27-SEP-02 15:53:00Z (2.58 hr)'
| c: '    88 Puffs 18.8s timestep'
| routine: 'Progress'

```

```

|
|
[example] Progress
  i=0, j=0
  a: 'example : Calculating'
  b: '27-SEP-02 15:58:00Z (2.67 hr)'
  c: '    94 Puffs  18.8s timestep'
  routine: 'Progress'

[example] Progress
  i=0, j=0
  a: 'example : Writing output'
  b: ''
  c: ''
  routine: 'Progress'

[example] Progress
  i=0, j=0
  a: ''
  b: ''
  c: 'Writing puff file'
  routine: 'Progress'

[example] Progress
  i=0, j=0
  a: ''
  b: ''
  c: 'Writing project file'
  routine: 'Progress'

[example] Progress
  i=0, j=0
  a: ''
  b: ''
  c: 'Writing surface file(s)'
  routine: 'Progress'

[example] Progress
  i=0, j=0
  a: ''
  b: ''
  c: ''
  routine: 'Progress'

|
|
[example] Progress
  i=0, j=0

```

```

a: 'example : Calculating'
b: '27-SEP-02 16:03:00Z (2.75 hr)'
c: '    91 Puffs 18.8s timestep'
routine: 'Progress'

|
| [example] Progress
|   i=0, j=0
|   a: 'example : Calculating'
|   b: '27-SEP-02 16:08:00Z (2.83 hr)'
|   c: '    86 Puffs 18.8s timestep'
|   routine: 'Progress'

|
| [example] Progress
|   i=0, j=0
|   a: 'example : Calculating'
|   b: '27-SEP-02 16:13:00Z (2.92 hr)'
|   c: '    85 Puffs 18.8s timestep'
|   routine: 'Progress'

[example] Progress
i=0, j=0
a: 'example : Writing output'
b: ''
c: ''
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: ''
c: 'Writing puff file'
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: ''
c: 'Writing project file'
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: ''

```

```

c: 'Writing surface file(s)'
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: ''
c: ''
routine: 'Progress'

|
| [example] Progress
| i=0, j=0
| a: 'example : Calculating'
| b: '27-SEP-02 16:18:00Z (3.00 hr)'
| c: '    87 Puffs 18.8s timestep'
| routine: 'Progress'

|
| [example] Progress
| i=0, j=0
| a: 'example : Calculating'
| b: '27-SEP-02 16:23:00Z (3.08 hr)'
| c: '    96 Puffs 18.8s timestep'
| routine: 'Progress'

|
| [example] Progress
| i=0, j=0
| a: 'example : Calculating'
| b: '27-SEP-02 16:28:00Z (3.17 hr)'
| c: '   109 Puffs 18.8s timestep'
| routine: 'Progress'

[example] Progress
i=0, j=0
a: 'example : Writing output'
b: ''
c: ''
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''

```

```

b: ''
c: 'Writing puff file'
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: ''
c: 'Writing project file'
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: ''
c: 'Writing surface file(s)'
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: ''
c: ''
routine: 'Progress'

|
| [example] Progress
| i=0, j=0
| a: 'example : Calculating'
| b: '27-SEP-02 16:33:00Z (3.25 hr)'
| c: ' 112 Puffs 18.8s timestep'
| routine: 'Progress'

|
| [example] Progress
| i=0, j=0
| a: 'example : Calculating'
| b: '27-SEP-02 16:38:00Z (3.33 hr)'
| c: ' 111 Puffs 18.8s timestep'
| routine: 'Progress'

|
| [example] Progress
| i=0, j=0
| a: 'example : Calculating'

```

```
b: '27-SEP-02 16:43:00Z (3.42 hr)'
c: ' 107 Puffs 18.8s timestep'
routine: 'Progress'
```

```
[example] Progress
i=0, j=0
a: 'example : Writing output'
b: ''
c: ''
routine: 'Progress'
```

```
[example] Progress
i=0, j=0
a: ''
b: ''
c: 'Writing puff file'
routine: 'Progress'
```

```
[example] Progress
i=0, j=0
a: ''
b: ''
c: 'Writing project file'
routine: 'Progress'
```

```
[example] Progress
i=0, j=0
a: ''
b: ''
c: 'Writing surface file(s)'
routine: 'Progress'
```

```
[example] Progress
i=0, j=0
a: ''
b: ''
c: ''
routine: 'Progress'
```

|

|

```
[example] Progress
i=0, j=0
a: 'example : Calculating'
b: '27-SEP-02 16:48:00Z (3.50 hr)'
c: ' 112 Puffs 18.8s timestep'
routine: 'Progress'
```

|

```

| [example] Progress
|   i=0, j=0
|   a: 'example : Calculating'
|   b: '27-SEP-02 16:53:00Z (3.58 hr)'
|   c: ' 114 Puffs 18.8s timestep'
|   routine: 'Progress'
```

```

| [example] Progress
|   i=0, j=0
|   a: 'example : Calculating'
|   b: '27-SEP-02 16:58:00Z (3.67 hr)'
|   c: ' 114 Puffs 18.8s timestep'
|   routine: 'Progress'
```

```

[example] Progress
  i=0, j=0
  a: 'example : Writing output'
  b: ''
  c: ''
  routine: 'Progress'
```

```

[example] Progress
  i=0, j=0
  a: ''
  b: ''
  c: 'Writing puff file'
  routine: 'Progress'
```

```

[example] Progress
  i=0, j=0
  a: ''
  b: ''
  c: 'Writing project file'
  routine: 'Progress'
```

```

[example] Progress
  i=0, j=0
  a: ''
  b: ''
  c: 'Writing surface file(s)'
  routine: 'Progress'
```

```

[example] Progress
  i=0, j=0
  a: ''
  b: ''
  c: ''
```

```

routine: 'Progress'

|
| [example] Progress
|   i=0, j=0
|   a: 'example : Calculating'
|   b: '27-SEP-02 17:03:00Z (3.75 hr)'
|   c: ' 111 Puffs 18.8s timestep'
|   routine: 'Progress'

|
| [example] Progress
|   i=0, j=0
|   a: 'example : Calculating'
|   b: '27-SEP-02 17:08:00Z (3.83 hr)'
|   c: ' 119 Puffs 18.8s timestep'
|   routine: 'Progress'

/
| [example] Progress
|   i=0, j=0
|   a: 'example : Calculating'
|   b: '27-SEP-02 17:13:00Z (3.92 hr)'
|   c: ' 108 Puffs 9.38s timestep'
|   routine: 'Progress'

/
[example] Progress
  i=0, j=0
  a: 'example : Writing output'
  b: ''
  c: ''
  routine: 'Progress'

[example] Progress
  i=0, j=0
  a: ''
  b: ''
  c: 'Writing puff file'
  routine: 'Progress'

[example] Progress
  i=0, j=0

```

```

a: ''
b: ''
c: 'Writing project file'
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: ''
c: 'Writing surface file(s)'
routine: 'Progress'

[example] Progress
i=0, j=0
a: ''
b: ''
c: ''
routine: 'Progress'

[example] Progress
i=0, j=0
a: 'Outputting wind field at end of run'
b: ''
c: ''
routine: 'Progress'

[example] Progress
i=0, j=0
a: 'SCIPUFF finished dispersion calculation'
b: 'Project=example'
c: ''
routine: 'Progress'

[example] Calculation Complete!
[example] retrieving plot information

```

A.9 run.out

Standard output produced from execution of example with no command-line arguments.

```

[PLOT INFO]
class_count=3, choice_count=1, kind_count=0
class names:
    Surface Dosage
    Surface Deposition
    Concentration
choice names:
    ANTH2
kind names:
puff times:
    0,2002,9,27,13.55,0.25, 43, 27-Sep-02 13:33:00Z

```

0,2002,9,27,13.8,0.5, 54, 27-Sep-02 13:48:00Z
 0,2002,9,27,14.05,0.75, 58, 27-Sep-02 14:03:00Z
 0,2002,9,27,14.3,1, 60, 27-Sep-02 14:18:00Z
 0,2002,9,27,14.55,1.25, 66, 27-Sep-02 14:33:00Z
 0,2002,9,27,14.8,1.5, 74, 27-Sep-02 14:48:00Z
 0,2002,9,27,15.05,1.75, 68, 27-Sep-02 15:03:00Z
 0,2002,9,27,15.3,2, 82, 27-Sep-02 15:18:00Z
 0,2002,9,27,15.55,2.25, 85, 27-Sep-02 15:33:00Z
 0,2002,9,27,15.8,2.5, 87, 27-Sep-02 15:48:00Z
 0,2002,9,27,16.05,2.75, 91, 27-Sep-02 16:03:00Z
 0,2002,9,27,16.3,3, 87, 27-Sep-02 16:18:00Z
 0,2002,9,27,16.55,3.25, 112, 27-Sep-02 16:33:00Z
 0,2002,9,27,16.8,3.5, 112, 27-Sep-02 16:48:00Z
 0,2002,9,27,17.05,3.75, 111, 27-Sep-02 17:03:00Z
 0,2002,9,27,17.3,4, 89, 27-Sep-02 17:18:00Z

surface times:

0,2002,9,27,13.55,0.25, 3594, 27-Sep-02 13:33:00Z
 0,2002,9,27,13.8,0.5, 4098, 27-Sep-02 13:48:00Z
 0,2002,9,27,14.05,0.75, 4490, 27-Sep-02 14:03:00Z
 0,2002,9,27,14.3,1, 4762, 27-Sep-02 14:18:00Z
 0,2002,9,27,14.55,1.25, 5066, 27-Sep-02 14:33:00Z
 0,2002,9,27,14.8,1.5, 5394, 27-Sep-02 14:48:00Z
 0,2002,9,27,15.05,1.75, 5586, 27-Sep-02 15:03:00Z
 0,2002,9,27,15.3,2, 5778, 27-Sep-02 15:18:00Z
 0,2002,9,27,15.55,2.25, 6042, 27-Sep-02 15:33:00Z
 0,2002,9,27,15.8,2.5, 6226, 27-Sep-02 15:48:00Z
 0,2002,9,27,16.05,2.75, 6314, 27-Sep-02 16:03:00Z
 0,2002,9,27,16.3,3, 6394, 27-Sep-02 16:18:00Z
 0,2002,9,27,16.55,3.25, 6522, 27-Sep-02 16:33:00Z
 0,2002,9,27,16.8,3.5, 6602, 27-Sep-02 16:48:00Z
 0,2002,9,27,17.05,3.75, 6802, 27-Sep-02 17:03:00Z
 0,2002,9,27,17.3,4, 6914, 27-Sep-02 17:18:00Z

category-class availability:

category Surface

Surface Dosage: avail=1, type=1
 Surface Deposition: avail=1, type=1
 Concentration: avail=0, type=0

category Surface Slice

Surface Dosage: avail=0, type=0
 Surface Deposition: avail=0, type=0
 Concentration: avail=0, type=0

category Horizontal Slice

Surface Dosage: avail=0, type=0
 Surface Deposition: avail=0, type=0
 Concentration: avail=1, type=1

category Vertically Integrated Slice

Surface Dosage: avail=0, type=0
 Surface Deposition: avail=0, type=0
 Concentration: avail=1, type=1

category Vertical Slice

Surface Dosage: avail=0, type=0
 Surface Deposition: avail=0, type=0
 Concentration: avail=1, type=1

```

category Table
  Surface Dosage: avail=0, type=0
  Surface Deposition: avail=0, type=0
  Concentration: avail=0, type=0
class-choice availability:
  Surface Dosage-ANTH2: available=1, kind=0, ikind=0, -
nkind=0, itime=2, usertime=0, units=
  Surface Deposition-ANTH2: available=2, kind=0, ikind=1, -
nkind=0, itime=0, usertime=0, units=
  Concentration-ANTH2: available=4224040, kind=4224085, ikind=4224130, -
nkind=4224175, itime=4224222, usertime=4224290, units=fu@

[countour lines] line_count=3, point_count=953
0: index=1, start=1, number=305, mode=-1
-85.0009,32.4998
-85.0009,32.4998
-85.0009,32.4998
-85.0008,32.4997
-85.0008,32.4997
-85.0008,32.4997
-85.0007,32.4996
-85.0007,32.4996
-85.0007,32.4996
-85.0006,32.4996
-85.0006,32.4995
-85.0005,32.4995
-85.0005,32.4995
-85.0005,32.4995
-85.0004,32.4994
-85.0004,32.4994
-85.0004,32.4994
-85.0003,32.4994
-85.0003,32.4994
-85.0002,32.4993
-85.0002,32.4993
-85.0001,32.4993
-85.0001,32.4993
-85.0001,32.4993
-84.9999,32.4993
-84.9999,32.4993
-84.9999,32.4993
-84.9998,32.4993
-84.9998,32.4993
-84.9998,32.4993
-84.9997,32.4994
-84.9997,32.4994
-84.9996,32.4994
-84.9996,32.4994
-84.9995,32.4995
-84.9995,32.4995
-84.9994,32.4996

```

-84.9993,32.4996
-84.9992,32.4997
-84.9992,32.4997
-84.9991,32.4998
-84.9991,32.4998
-84.9991,32.4998
-84.999,32.4999
-84.9989,32.4999
-84.9988,32.5
-84.9988,32.5001
-84.9988,32.5002
-84.9987,32.5003
-84.9986,32.5004
-84.9985,32.5006
-84.9984,32.5006
-84.9984,32.5006
-84.9982,32.5009
-84.9982,32.5009
-84.998,32.5011
-84.9979,32.5013
-84.9979,32.5013
-84.9978,32.5015
-84.9977,32.5016
-84.9977,32.5017
-84.9976,32.5018
-84.9974,32.5021
-84.9974,32.5022
-84.9974,32.5022
-84.9973,32.5023
-84.9971,32.5027
-84.9969,32.5031
-84.9969,32.5032
-84.9968,32.5033
-84.9967,32.5037
-84.9965,32.504
-84.9964,32.5042
-84.9963,32.5044
-84.9962,32.5046
-84.9962,32.5048
-84.9961,32.5051
-84.9959,32.5056
-84.9959,32.5056
-84.9958,32.5057
-84.9957,32.5061
-84.9956,32.5064
-84.9955,32.5066
-84.9954,32.507
-84.9954,32.5071
-84.9952,32.5075
-84.9951,32.5078
-84.995,32.5082
-84.995,32.5084
-84.9948,32.5091

-84.9947,32.5092
-84.9947,32.5093
-84.9946,32.5096
-84.9944,32.5103
-84.9943,32.5106
-84.9941,32.5112
-84.9938,32.5121
-84.9937,32.5122
-84.9937,32.5126
-84.9935,32.5132
-84.9934,32.5135
-84.9931,32.5142
-84.9929,32.5149
-84.9929,32.5151
-84.9927,32.5159
-84.9926,32.5161
-84.9926,32.5162
-84.9924,32.5171
-84.9923,32.5174
-84.9922,32.5181
-84.9921,32.5186
-84.992,32.519
-84.9918,32.5199
-84.9918,32.52
-84.9917,32.5207
-84.9917,32.521
-84.9916,32.5211
-84.9914,32.522
-84.9913,32.5223
-84.9912,32.5229
-84.9912,32.5243
-84.9912,32.5244
-84.9912,32.5245
-84.9909,32.5264
-84.9908,32.5268
-84.9906,32.5283
-84.9904,32.5291
-84.9903,32.5303
-84.9902,32.5313
-84.9901,32.5322
-84.99,32.5334
-84.99,32.5342
-84.9899,32.5355
-84.9899,32.5361
-84.9898,32.5375
-84.9899,32.5381
-84.9899,32.5394
-84.99,32.54
-84.9901,32.5411
-84.9903,32.542
-84.9905,32.5427
-84.9909,32.5439
-84.991,32.5442

-84.9912, 32.5447
-84.9916, 32.5455
-84.9917, 32.5459
-84.9923, 32.5468
-84.9929, 32.5479
-84.993, 32.548
-84.9932, 32.5482
-84.994, 32.5489
-84.9951, 32.5498
-84.9951, 32.5498
-84.9951, 32.5498
-84.9964, 32.5505
-84.9971, 32.5509
-84.9978, 32.551
-84.999, 32.5512
-84.9995, 32.5512
-85.001, 32.5512
-85.0016, 32.5511
-85.0029, 32.5509
-85.0048, 32.5499
-85.0049, 32.5498
-85.0049, 32.5498
-85.0068, 32.5482
-85.0071, 32.5479
-85.0074, 32.5472
-85.0083, 32.5459
-85.0088, 32.5447
-85.0091, 32.5439
-85.0092, 32.5435
-85.0097, 32.542
-85.0099, 32.5409
-85.01, 32.54
-85.0101, 32.5387
-85.0101, 32.5381
-85.0101, 32.5368
-85.0101, 32.5361
-85.01, 32.5349
-85.01, 32.5342
-85.0099, 32.5331
-85.0099, 32.5322
-85.0097, 32.5313
-85.0097, 32.5303
-85.0095, 32.5295
-85.0094, 32.5283
-85.0093, 32.5278
-85.0091, 32.5264
-85.0091, 32.5261
-85.0088, 32.5244
-85.0088, 32.5244
-85.0088, 32.5243
-85.0088, 32.5229
-85.0087, 32.5226
-85.0086, 32.522

-85.0085,32.5217
-85.0083,32.521
-85.0083,32.521
-85.0083,32.5207
-85.0082,32.5201
-85.0082,32.52
-85.008,32.5193
-85.008,32.519
-85.0079,32.5185
-85.0078,32.5181
-85.0077,32.5177
-85.0076,32.5171
-85.0075,32.5169
-85.0074,32.5161
-85.0074,32.5161
-85.0073,32.5159
-85.0072,32.5153
-85.0071,32.5151
-85.0069,32.5145
-85.0069,32.5142
-85.0067,32.5138
-85.0065,32.5132
-85.0065,32.513
-85.0063,32.5126
-85.0063,32.5123
-85.0063,32.5122
-85.006,32.5116
-85.0059,32.5112
-85.0058,32.5108
-85.0056,32.5103
-85.0055,32.5101
-85.0054,32.5096
-85.0053,32.5094
-85.0053,32.5093
-85.0053,32.5092
-85.0051,32.5086
-85.005,32.5084
-85.005,32.5082
-85.0048,32.5078
-85.0048,32.5075
-85.0046,32.5071
-85.0046,32.5071
-85.0046,32.507
-85.0045,32.5067
-85.0045,32.5066
-85.0044,32.5064
-85.0043,32.5061
-85.0043,32.506
-85.0042,32.5057
-85.0041,32.5056
-85.0041,32.5056
-85.004,32.5053
-85.0039,32.5051

-85.0039,32.5049
-85.0038,32.5046
-85.0037,32.5046
-85.0037,32.5044
-85.0036,32.5042
-85.0036,32.5042
-85.0034,32.5039
-85.0033,32.5037
-85.0033,32.5035
-85.0032,32.5033
-85.0031,32.5032
-85.0031,32.5032
-85.003,32.5029
-85.0029,32.5027
-85.0028,32.5025
-85.0027,32.5023
-85.0027,32.5022
-85.0026,32.5022
-85.0026,32.5022
-85.0025,32.5019
-85.0024,32.5018
-85.0023,32.5017
-85.0023,32.5016
-85.0023,32.5016
-85.0022,32.5015
-85.0021,32.5013
-85.0021,32.5013
-85.0021,32.5013
-85.002,32.5012
-85.002,32.5011
-85.0019,32.501
-85.0018,32.5009
-85.0018,32.5009
-85.0018,32.5009
-85.0017,32.5007
-85.0016,32.5006
-85.0016,32.5006
-85.0015,32.5005
-85.0014,32.5004
-85.0014,32.5003
-85.0013,32.5003
-85.0013,32.5002
-85.0012,32.5002
-85.0012,32.5001
-85.0012,32.5001
-85.0012,32.5
-85.0011,32.5
-85.0011,32.4999
-85.001,32.4999
-85.001,32.4999
-85.0009,32.4998

```

-85.0009,32.4998
1: index=2, start=306, number=323, mode=-1
-85.0003,32.4995
-85.0003,32.4995
-85.0002,32.4995
-85.0002,32.4995
-85.0002,32.4995
-85.0002,32.4995
-85.0001,32.4995
-85.0001,32.4995
-85.0001,32.4995
-85,32.4995
-85,32.4995
-85,32.4995
-85,32.4995
-84.9999,32.4995
-84.9999,32.4995
-84.9999,32.4995
-84.9998,32.4995
-84.9998,32.4995
-84.9998,32.4995
-84.9998,32.4995
-84.9998,32.4995
-84.9997,32.4995
-84.9997,32.4995
-84.9997,32.4995
-84.9997,32.4996
-84.9996,32.4996
-84.9996,32.4996
-84.9996,32.4996
-84.9995,32.4996
-84.9995,32.4997
-84.9995,32.4997
-84.9995,32.4997
-84.9994,32.4997
-84.9994,32.4998
-84.9993,32.4998
-84.9993,32.4998
-84.9992,32.4999
-84.9992,32.5
-84.9991,32.5001
-84.9991,32.5001
-84.999,32.5002
-84.999,32.5003
-84.9989,32.5003
-84.9988,32.5004
-84.9988,32.5004
-84.9988,32.5004
-84.9987,32.5006
-84.9987,32.5006
-84.9987,32.5007
-84.9986,32.5009
-84.9984,32.5011
-84.9984,32.5011

```

-84.9984,32.5011
-84.9983,32.5013
-84.9982,32.5015
-84.9981,32.5016
-84.9981,32.5017
-84.998,32.5018
-84.9979,32.5019
-84.9978,32.5021
-84.9978,32.5022
-84.9977,32.5023
-84.9977,32.5025
-84.9976,32.5026
-84.9975,32.5028
-84.9974,32.5031
-84.9974,32.5032
-84.9974,32.5032
-84.9973,32.5033
-84.9972,32.5037
-84.9971,32.5039
-84.997,32.5042
-84.9968,32.5046
-84.9968,32.5046
-84.9968,32.5047
-84.9967,32.5051
-84.9966,32.5054
-84.9965,32.5056
-84.9963,32.5061
-84.9963,32.5061
-84.9963,32.5061
-84.9962,32.5066
-84.9961,32.5068
-84.996,32.5071
-84.9959,32.5075
-84.9959,32.5076
-84.9958,32.5078
-84.9958,32.5081
-84.9957,32.5082
-84.9957,32.5085
-84.9956,32.5088
-84.9955,32.509
-84.9954,32.5095
-84.9954,32.5095
-84.9954,32.5096
-84.9953,32.5103
-84.9952,32.5107
-84.995,32.5112
-84.9948,32.512
-84.9948,32.5122
-84.9946,32.5127
-84.9945,32.5132
-84.9945,32.5133
-84.9943,32.5142
-84.9942,32.5146

-84.9941,32.5151
-84.9939,32.5159
-84.9939,32.5161
-84.9937,32.517
-84.9937,32.5171
-84.9937,32.5175
-84.9935,32.5181
-84.9935,32.5182
-84.9934,32.519
-84.9933,32.5194
-84.9932,32.52
-84.9931,32.5205
-84.9931,32.521
-84.993,32.5217
-84.9929,32.522
-84.9928,32.5228
-84.9928,32.5229
-84.9927,32.5236
-84.9926,32.5239
-84.9925,32.5249
-84.9925,32.5251
-84.9926,32.5264
-84.9925,32.527
-84.9924,32.5283
-84.9923,32.5292
-84.9923,32.5303
-84.9922,32.5312
-84.9923,32.5322
-84.9922,32.5331
-84.9923,32.5342
-84.9923,32.5351
-84.9923,32.5361
-84.9924,32.5369
-84.9925,32.5381
-84.9925,32.5387
-84.9927,32.54
-84.9928,32.5404
-84.9931,32.542
-84.9931,32.542
-84.9932,32.5421
-84.9938,32.5433
-84.9943,32.5439
-84.9946,32.5445
-84.9951,32.5451
-84.9955,32.5455
-84.9958,32.5459
-84.9965,32.5465
-84.9971,32.5469
-84.9978,32.5471
-84.999,32.5474
-84.9994,32.5474
-85.001,32.5474
-85.0015,32.5473

-85.0029,32.5469
-85.0042,32.5459
-85.0049,32.5451
-85.0057,32.5439
-85.0068,32.5421
-85.0069,32.542
-85.0069,32.5419
-85.0073,32.54
-85.0073,32.5395
-85.0075,32.5381
-85.0076,32.5374
-85.0077,32.5361
-85.0076,32.5353
-85.0077,32.5342
-85.0077,32.5333
-85.0077,32.5322
-85.0077,32.5314
-85.0077,32.5303
-85.0076,32.5295
-85.0076,32.5283
-85.0075,32.5277
-85.0074,32.5264
-85.0073,32.5259
-85.0075,32.5249
-85.0074,32.5239
-85.0074,32.5239
-85.0073,32.5236
-85.0072,32.523
-85.0072,32.5229
-85.0071,32.5222
-85.0071,32.522
-85.007,32.5213
-85.0069,32.521
-85.0068,32.5205
-85.0068,32.52
-85.0067,32.5197
-85.0066,32.519
-85.0066,32.5188
-85.0065,32.5181
-85.0065,32.518
-85.0063,32.5175
-85.0063,32.5172
-85.0063,32.5171
-85.0062,32.5163
-85.0061,32.5161
-85.006,32.5155
-85.0059,32.5151
-85.0058,32.5147
-85.0057,32.5142
-85.0056,32.5139
-85.0055,32.5132
-85.0055,32.5131
-85.0054,32.5127

-85.0053,32.5123
-85.0052,32.5122
-85.005,32.5116
-85.005,32.5112
-85.0048,32.5108
-85.0047,32.5103
-85.0046,32.51
-85.0046,32.5096
-85.0046,32.5095
-85.0045,32.5092
-85.0045,32.509
-85.0044,32.5088
-85.0043,32.5085
-85.0043,32.5084
-85.0042,32.5081
-85.0042,32.508
-85.0042,32.5078
-85.0041,32.5076
-85.0041,32.5076
-85.004,32.5072
-85.004,32.5071
-85.0039,32.5069
-85.0038,32.5066
-85.0038,32.5065
-85.0037,32.5061
-85.0037,32.5061
-85.0035,32.5057
-85.0035,32.5056
-85.0034,32.5054
-85.0033,32.5051
-85.0033,32.505
-85.0032,32.5047
-85.0032,32.5047
-85.0032,32.5046
-85.003,32.5043
-85.003,32.5042
-85.0029,32.5039
-85.0028,32.5037
-85.0028,32.5036
-85.0027,32.5033
-85.0026,32.5032
-85.0026,32.5032
-85.0025,32.5029
-85.0025,32.5028
-85.0024,32.5026
-85.0023,32.5025
-85.0023,32.5025
-85.0023,32.5024
-85.0023,32.5023
-85.0022,32.5022
-85.0022,32.5021

-85.0021,32.502
-85.0021,32.5019
-85.002,32.5019
-85.002,32.5018
-85.002,32.5017
-85.0019,32.5016
-85.0019,32.5016
-85.0018,32.5015
-85.0018,32.5014
-85.0017,32.5013
-85.0017,32.5013
-85.0016,32.5011
-85.0016,32.5011
-85.0016,32.5011
-85.0015,32.501
-85.0014,32.5009
-85.0014,32.5008
-85.0013,32.5007
-85.0013,32.5007
-85.0013,32.5006
-85.0013,32.5006
-85.0012,32.5005
-85.0012,32.5004
-85.0012,32.5004
-85.0011,32.5004
-85.0011,32.5003
-85.0011,32.5003
-85.001,32.5003
-85.001,32.5002
-85.001,32.5001
-85.0009,32.5001
-85.0009,32.5001
-85.0009,32.5001
-85.0008,32.5
-85.0008,32.5
-85.0008,32.5
-85.0008,32.4999
-85.0007,32.4999
-85.0007,32.4998
-85.0007,32.4998
-85.0007,32.4998
-85.0006,32.4998
-85.0006,32.4998
-85.0006,32.4997
-85.0005,32.4997
-85.0005,32.4997
-85.0005,32.4997
-85.0005,32.4997
-85.0005,32.4996
-85.0004,32.4996
-85.0004,32.4996

```

-85.0004,32.4996
-85.0004,32.4996
-85.0004,32.4996
-85.0003,32.4996
-85.0003,32.4996
-85.0003,32.4995
-85.0003,32.4995
2: index=3, start=629, number=325, mode=-1
-85.0005,32.4998
-85.0005,32.4998
-85.0004,32.4998
-85.0004,32.4998
-85.0004,32.4998
-85.0004,32.4998
-85.0004,32.4998
-85.0004,32.4998
-85.0004,32.4998
-85.0004,32.4998
-85.0003,32.4997
-85.0003,32.4997
-85.0003,32.4997
-85.0003,32.4997
-85.0003,32.4997
-85.0003,32.4997
-85.0003,32.4997
-85.0002,32.4997
-85.0002,32.4997
-85.0002,32.4996
-85.0002,32.4996
-85.0001,32.4996
-85.0001,32.4996
-85.0001,32.4996
-85,32.4996
-85,32.4996
-85,32.4996
-85,32.4996
-85,32.4996
-85,32.4996
-84.9999,32.4996
-84.9999,32.4996
-84.9998,32.4996
-84.9998,32.4996
-84.9998,32.4997
-84.9998,32.4997
-84.9997,32.4997
-84.9997,32.4997
-84.9997,32.4997
-84.9996,32.4998
-84.9996,32.4998
-84.9996,32.4998
-84.9995,32.4998
-84.9995,32.4998
-84.9995,32.4999
-84.9995,32.4999
-84.9995,32.4999
-84.9994,32.5

```

-84.9994,32.5
-84.9994,32.5
-84.9994,32.5001
-84.9993,32.5001
-84.9993,32.5002
-84.9992,32.5003
-84.9992,32.5003
-84.9992,32.5004
-84.9991,32.5004
-84.9991,32.5005
-84.999,32.5005
-84.999,32.5007
-84.999,32.5007
-84.999,32.5007
-84.9989,32.5008
-84.9988,32.5009
-84.9988,32.5009
-84.9988,32.5009
-84.9987,32.5011
-84.9987,32.5011
-84.9987,32.5013
-84.9986,32.5013
-84.9986,32.5014
-84.9985,32.5016
-84.9984,32.5018
-84.9984,32.5018
-84.9984,32.5019
-84.9983,32.5021
-84.9982,32.5023
-84.9982,32.5023
-84.9981,32.5026
-84.998,32.5027
-84.998,32.5028
-84.9979,32.5029
-84.9979,32.5031
-84.9979,32.5031
-84.9978,32.5033
-84.9978,32.5035
-84.9977,32.5035
-84.9977,32.5037
-84.9976,32.5038
-84.9976,32.5038
-84.9976,32.5041
-84.9975,32.5042
-84.9975,32.5045
-84.9974,32.5046
-84.9973,32.505
-84.9973,32.5051
-84.9973,32.5052
-84.9972,32.5056
-84.9971,32.5058
-84.997,32.5061

-84.997, 32.5065
-84.9969, 32.5066
-84.9968, 32.507
-84.9968, 32.5071
-84.9968, 32.5071
-84.9967, 32.5076
-84.9967, 32.5077
-84.9966, 32.5081
-84.9965, 32.5084
-84.9965, 32.5085
-84.9964, 32.509
-84.9964, 32.509
-84.9963, 32.5093
-84.9963, 32.5095
-84.9963, 32.5096
-84.9962, 32.51
-84.9962, 32.5104
-84.9961, 32.5107
-84.9961, 32.5112
-84.996, 32.5118
-84.9959, 32.5122
-84.9958, 32.513
-84.9958, 32.5132
-84.9956, 32.5141
-84.9956, 32.5142
-84.9956, 32.5142
-84.9955, 32.5151
-84.9954, 32.5153
-84.9953, 32.5161
-84.9953, 32.5165
-84.9952, 32.5171
-84.9951, 32.5175
-84.9951, 32.5181
-84.995, 32.5187
-84.995, 32.519
-84.9949, 32.5197
-84.9949, 32.52
-84.9948, 32.5208
-84.9948, 32.521
-84.9948, 32.5218
-84.9948, 32.522
-84.9947, 32.5229
-84.9947, 32.5229
-84.9946, 32.5239
-84.9946, 32.5239
-84.9946, 32.524
-84.9946, 32.5249
-84.9946, 32.5249
-84.9946, 32.5259
-84.9945, 32.5268
-84.9945, 32.527
-84.9947, 32.5283
-84.9947, 32.5287

-84.9948,32.5303
-84.9948,32.5306
-84.995,32.5322
-84.995,32.5323
-84.9951,32.533
-84.9953,32.534
-84.9954,32.5342
-84.9957,32.5355
-84.9959,32.5361
-84.9962,32.537
-84.9965,32.5381
-84.9967,32.5385
-84.9971,32.5395
-84.9973,32.5398
-84.9976,32.54
-84.9984,32.5407
-84.999,32.5411
-84.9999,32.5411
-85.001,32.5411
-85.0024,32.54
-85.0029,32.5395
-85.0035,32.5381
-85.0037,32.5373
-85.0041,32.5361
-85.0045,32.5346
-85.0046,32.5342
-85.0049,32.533
-85.005,32.5322
-85.005,32.5321
-85.0052,32.5303
-85.0052,32.53
-85.0053,32.5283
-85.0053,32.5279
-85.0055,32.5268
-85.0054,32.5259
-85.0054,32.5258
-85.0054,32.5249
-85.0054,32.5249
-85.0054,32.524
-85.0054,32.5239
-85.0054,32.5239
-85.0053,32.523
-85.0053,32.5229
-85.0052,32.5221
-85.0052,32.522
-85.0052,32.5212
-85.0052,32.521
-85.0051,32.5203
-85.0051,32.52
-85.005,32.5194
-85.005,32.519
-85.0049,32.5185
-85.0049,32.5181

-85.0049,32.5176
-85.0048,32.5171
-85.0047,32.5168
-85.0047,32.5161
-85.0046,32.5159
-85.0045,32.5151
-85.0045,32.515
-85.0044,32.5142
-85.0044,32.5142
-85.0044,32.5141
-85.0042,32.5133
-85.0042,32.5132
-85.0041,32.5125
-85.0041,32.5122
-85.0039,32.5117
-85.0039,32.5112
-85.0038,32.5108
-85.0038,32.5104
-85.0038,32.51
-85.0038,32.5099
-85.0037,32.5095
-85.0037,32.5095
-85.0037,32.5093
-85.0036,32.5091
-85.0036,32.509
-85.0035,32.5087
-85.0035,32.5085
-85.0034,32.5083
-85.0034,32.5081
-85.0034,32.5079
-85.0033,32.5076
-85.0033,32.5075
-85.0032,32.5071
-85.0032,32.5071
-85.0032,32.507
-85.0031,32.5067
-85.0031,32.5066
-85.003,32.5063
-85.003,32.5061
-85.0029,32.5059
-85.0028,32.5056
-85.0028,32.5055
-85.0027,32.5051
-85.0027,32.5051
-85.0027,32.505
-85.0026,32.5047
-85.0026,32.5046
-85.0025,32.5044
-85.0025,32.5042
-85.0024,32.5041
-85.0024,32.5039
-85.0024,32.5038
-85.0023,32.5037

-85.0023,32.5036
-85.0023,32.5035
-85.0022,32.5034
-85.0022,32.5033
-85.0022,32.5032
-85.0021,32.5031
-85.0021,32.503
-85.0021,32.5029
-85.002,32.5028
-85.002,32.5028
-85.002,32.5027
-85.0019,32.5026
-85.0019,32.5025
-85.0018,32.5023
-85.0018,32.5023
-85.0018,32.5023
-85.0018,32.5022
-85.0017,32.5021
-85.0017,32.502
-85.0016,32.5018
-85.0016,32.5018
-85.0016,32.5018
-85.0015,32.5017
-85.0015,32.5016
-85.0014,32.5015
-85.0014,32.5013
-85.0014,32.5013
-85.0013,32.5013
-85.0013,32.5012
-85.0013,32.5011
-85.0013,32.5011
-85.0012,32.501
-85.0012,32.5009
-85.0012,32.5009
-85.0011,32.5008
-85.0011,32.5008
-85.0011,32.5008
-85.001,32.5007
-85.001,32.5007
-85.001,32.5007
-85.001,32.5006
-85.001,32.5005
-85.001,32.5005
-85.0009,32.5005
-85.0009,32.5004
-85.0009,32.5004
-85.0009,32.5004
-85.0008,32.5003
-85.0008,32.5003
-85.0008,32.5003
-85.0007,32.5002
-85.0007,32.5002
-85.0007,32.5001

-85.0007,32.5001
-85.0007,32.5001
-85.0006,32.5001
-85.0006,32.5
-85.0006,32.5
-85.0006,32.5
-85.0006,32.5
-85.0006,32.5
-85.0005,32.4999
-85.0005,32.4999
-85.0005,32.4999
-85.0005,32.4999
-85.0005,32.4999
-85.0005,32.4998
-85.0005,32.4998

Appendix B

Sample Detailed Services Client Programs

This is an example client to HPAC's detailed CORBA services. A single Java class accesses the Chemical-Biological Weapon incident source model to create releases fed to ScipuffServer. Plot data are retrieved.

This example was run under Windows XP and Linux using versions 1.3.1 and 1.4.0 of the J2SE from Sun Microsystems. The server was launched with the script listed in Section 3.2.1 under Windows XP. The DetailClient.java file was compiled with a command line option to set the extension directory list to the shared/ext and server/ext subdirectories under the HPAC installation as well as lib/ext under the J2SE directory, and the class path to the parent of the samples directory containing the source file. In the Windows example below, the shell variable "HD" refers to the HPAC installation directory, and "JH" refers to the J2SE directory.

```
java -J-Djava.ext.dirs=%HD%\shared\ext;%HD%\server\ext;%JH%\lib\ext DetailClient.java
```

B.1 DetailClient.java

```
// $Id$  
//-----  
//      NAME:          DetailClient.java  
//      PURPOSE:  
//                  Example detailed CORBA services client  
//-----  
package samples;  
  
import javax.naming.Context;  
import javax.naming.InitialContext;  
import javax.naming.NameClassPair;  
import javax.naming.NamingEnumeration;  
  
import org.omg.CORBA.Any;  
import org.omg.CORBA.AnyHolder;  
import org.omg.CORBA.IntHolder;  
import org.omg.CORBA.ORB;
```

```

import mil.dtra.hpac.models.cbwpn.server.CHEMBIOWEAPON_FACTORY_SERVICE_NAME;
import mil.dtra.hpac.models.cbwpn.server.ChemBioWeaponIncidentT;
import mil.dtra.hpac.models.cbwpn.server.ChemBioWeaponIncidentTHelper;
import mil.dtra.hpac.models.cbwpn.server.ChemBioWeaponServer;
import mil.dtra.hpac.models.cbwpn.server.ChemBioWeaponServerFactory;
import mil.dtra.hpac.models.cbwpn.server.ChemBioWeaponServerHelper;
import mil.dtra.hpac.models.cbwpn.server.ChemBioWeaponServerFactoryHelper;

import mil.dtra.hpac.material.server.HM_AEROSOL;
import mil.dtra.hpac.material.server.HM_GAS;
import mil.dtra.hpac.material.server.HM_LIQUID;
import mil.dtra.hpac.material.server.HM_PARTICLE;
import mil.dtra.hpac.material.server.MaterialT;

import mil.dtra.hpac.server.IncidentT;
import mil.dtra.hpac.server.IncidentTHolder;

import mil.dtra.hpac.server.files.FileReferenceT;
import mil.dtra.hpac.server.fileutils.FileReferenceTMgr;

import mil.dtra.hpac.server.plot.CLOSE_CONTOUR;
import mil.dtra.hpac.server.plot.HP_LEFTHAND;
import mil.dtra.hpac.server.plot.HP_ON;
import mil.dtra.hpac.server.plot.HP_OPEN;
import mil.dtra.hpac.server.plot.HP_RIGHTHAND;
import mil.dtra.hpac.server.plot.HP_SURF;
import mil.dtra.hpac.server.plot.HPACCategoryClassT;
import mil.dtra.hpac.server.plot.HPACClassChoiceT;
import mil.dtra.hpac.server.plot.HPACContourElementT;
import mil.dtra.hpac.server.plot.HPACContourElementTListHolder;
import mil.dtra.hpac.server.plot.HPACContourHeaderT;
import mil.dtra.hpac.server.plot.HPACFieldCoordinateT;
import mil.dtra.hpac.server.plot.HPACFieldCoordinateTHolder;
import mil.dtra.hpac.server.plot.HPACLineT;
import mil.dtra.hpac.server.plot.HPACLineTListHolder;
import mil.dtra.hpac.server.plot.HPACPlotFieldT;
import mil.dtra.hpac.server.plot.HPACPlotFieldTHolder;
import mil.dtra.hpac.server.plot.HPACPlotTypeT;
import mil.dtra.hpac.server.plot.HPACPointT;
import mil.dtra.hpac.server.plot.HPACPointTListHolder;
import mil.dtra.hpac.server.plot.HPACSliceT;
import mil.dtra.hpac.server.plot.HPACTimeT;
import mil.dtra.hpac.server.plot.HPACTimeTListHolder;
import mil.dtra.hpac.server.plot.LATLON_OUTPUT;
import mil.dtra.hpac.server.plot.PLOT_LIN;
import mil.dtra.hpac.server.plot.PLOT_ON;
import mil.dtra.hpac.server.plot.ReferenceT;

import mil.dtra.hpac.server.project.AuditT;
import mil.dtra.hpac.server.project.FlagsT;
import mil.dtra.hpac.server.project.HD_LATLON;
import mil.dtra.hpac.server.project.HF_DENSE;
import mil.dtra.hpac.server.project.HF_DYNAMIC;

```

```

import mil.dtra.hpac.server.project.HF_HAZARD;
import mil.dtra.hpac.server.project.HF_STATIC;
import mil.dtra.hpac.server.project.LimitT;
import mil.dtra.hpac.server.project.OptionsT;
import mil.dtra.hpac.server.project.SpatialDomainT;
import mil.dtra.hpac.server.project.SpatialDomainTHolder;
import mil.dtra.hpac.server.project.TemporalDomainT;
import mil.dtra.hpac.server.project.TemporalDomainTHolder;
import mil.dtra.hpac.server.project.TimeT;

import mil.dtra.hpac.server.release.ContinuousReleaseT;
import mil.dtra.hpac.server.release.HR_CONTINUOUS;
import mil.dtra.hpac.server.release.HR_FILE;
import mil.dtra.hpac.server.release.HR_INSTANTANEOUS;
import mil.dtra.hpac.server.release.HR_MOVING;
import mil.dtra.hpac.server.release.HR_POOL;
import mil.dtra.hpac.server.release.HR_STACK;
import mil.dtra.hpac.server.release.InstantaneousReleaseT;
import mil.dtra.hpac.server.release.ReleaseDataT;
import mil.dtra.hpac.server.release.ReleaseT;
import mil.dtra.hpac.server.release.RS_VALID;
import mil.dtra.hpac.server.release.RSI_CONT_STATUS_COUNT;
import mil.dtra.hpac.server.release.RSI_INST_STATUS_COUNT;

import mil.dtra.hpac.server.scipuff.CategoryClass2DHolder;
import mil.dtra.hpac.server.scipuff.ClassChoice2DHolder;
import mil.dtra.hpac.server.scipuff.DispersionCalculator;
import mil.dtra.hpac.server.scipuff.HM_DISPERSION_END;
import mil.dtra.hpac.server.scipuff.HM_ERROR;
import mil.dtra.hpac.server.scipuff.HM_INFO;
import mil.dtra.hpac.server.scipuff.HM_PROGRESSMSG;
import mil.dtra.hpac.server.scipuff.HM_REPLY;
import mil.dtra.hpac.server.scipuff.HPAC_AFFIRMATIVE_REPLY;
import mil.dtra.hpac.server.scipuff.HPAC_FAILURE;
import mil.dtra.hpac.server.scipuff.HPAC_NEGATIVE_REPLY;
import mil.dtra.hpac.server.scipuff.HPAC_SUCCESS;
import mil.dtra.hpac.server.scipuff.MessageT;
import mil.dtra.hpac.server.scipuff.PlotException;
import mil.dtra.hpac.server.scipuff.PlotGenerator;
import mil.dtra.hpac.server.scipuff.SCIPUFF_FACTORY_SERVICE_NAME;
import mil.dtra.hpac.server.scipuff.ScipuffServer;
import mil.dtra.hpac.server.scipuff.ScipuffServerFactory;
import mil.dtra.hpac.server.scipuff.ScipuffServerHelper;
import mil.dtra.hpac.server.scipuff.ScipuffServerFactoryHelper;
import mil.dtra.hpac.server.scipuff.StringListHolder;

import mil.dtra.models.CBFac.AgentsList.AgentsList;
import mil.dtra.models.CBFac.AgentsList.AgentsListHelper;
import mil.dtra.models.CBFac.AgentsList.SWFIAgentMatValues;
import mil.dtra.models.CBFac.AgentsList.SWFIAgents;

import mil.dtra.models.CBFac.SWFI.AgentSWFI;
import mil.dtra.models.CBFac.SWFI.FacilitySWFI;

```

```

import mil.dtra.models.CBFac.SWFI.InputsSWFI;
import mil.dtra.models.CBFac.SWFI.ResultsSWFI;
import mil.dtra.models.CBFac.SWFI.RunConditionsSWFI;
import mil.dtra.models.CBFac.SWFI.StateVarSWFI;
import mil.dtra.models.CBFac.SWFI.TypeSWFI;
import mil.dtra.models.CBFac.SWFI.WarHeadSWFI;

import mil.dtra.weather.shared.data.weatherT.*;

//-----
//      CLASS:          DetailClient
//*****-
* Mondo class providing default inputs, <tt>toString()</tt> methods for
* convenient feedback, a <tt>PollerThread</tt> inner class demonstrating
* how to poll <tt>ScipuffServer</tt>, and a <tt>main()</tt> method
* demonstrating the sequence of calls to <tt>ScipuffServer</tt> and
* an incident source model server.
*****/
public class
DetailClient
{
    // Constants
    //

    /**
     * Predefined category names, unfortunately
     * not specified in plot.idl (yet)
     */
    public final static
    String[]          PLOT_CATEGORY_NAMES =
    {
        "Surface", "Surface Slice", "Horizontal Slice",
        "Vertically Integrated Slice", "Vertical Slice",
        "Table",
    };

    /**
     * Plot type value selecting the mean,
     * should be defined in plot.idl
     */
    public final static
    int               PLOTTYPE_mean = 1;

    /**
     * Plot type value selecting a probability,
     * should be defined in plot.idl
     */
    public final static
    int               PLOTTYPE_prob = 2;

    /**
     * Plot type value selecting exceedance,

```

```

        * should be defined in plot.idl
        */
public final static
    int          PLOTTYPE_exceed = 3;

        /**
        * Plot type value selecting variance,
        * should be defined in plot.idl
        */
public final static
    int          PLOTTYPE_variance = 4;

        // Inner Classes
        //

//-----
// CLASS:          PollerThread
// *****
* Thread for polling ScipuffServer during calculations and plotting.
*****/
public static class
PollerThread
    extends Thread
{
private
    ScipuffServer      fServer;

private
    int              fStatus = HPAC_FAILURE.value;

//-----
// METHOD:          PollerThread.PollerThread()
// *****
*****/
public
PollerThread( ScipuffServer server )
{
    fServer = server;
} // PollerThread

//-----
// METHOD:          PollerThread.getStatus()
// *****
*****/
public final int
getStatus()
{
    return fStatus;
} // getStatus

```

```

//-----
//  METHOD:          PollerThread.processMessage()
// ****
* @return          true if terminate message received
*****/
protected boolean
processMessage( MessageT message )
{
    boolean end_flag = false;

    switch ( message.fMessageType )
    {
        case HM_DISPERSION_END.value:
            end_flag = true;
            fStatus = message.fMessageValue;
            break;

        case HM_ERROR.value:
            System.err.println(
                new StringBuffer( 512 ) .
                append( "\n[Server Message] Error\n" ) .
                append( message.fMessageString1.trim() ).append( "\n" ) .
                append( message.fMessageString2.trim() ).append( "\n" ) .
                append( message.fMessageString3.trim() ) .
                toString()
            );
            break;

        case HM_PROGRESSMSG.value:
        case HM_INFO.value:
            System.err.println(
                new StringBuffer( 512 ) .
                append( "\n[Server Message] Info\n" ) .
                append( message.fMessageString1.trim() ).append( "\n" ) .
                append( message.fMessageString2.trim() ).append( "\n" ) .
                append( message.fMessageString3.trim() ) .
                toString()
            );
            break;

        case HM_REPLY.value:
            try
            {
                fServer.reply(
                    message.fMessageParameter == 0 ?
                    HPAC_NEGATIVE_REPLY.value :
                    HPAC_AFFIRMATIVE_REPLY.value
                );
            }
            catch ( Exception ex ) {}
            break;
    }
}

```

```

        default:
            break;
    }

    return end_flag;
} // processMessage

//-----
//  METHOD:          PollerThread.run()  -
//***** ****
***** ****
public void
run()
{
    fStatus = HPAC_FAILURE.value;

    for ( boolean polling = true; polling; )
    {
        try
        {
            System.err.println( "[Poller] polling" );
            MessageT[] poll_messages = fServer.poll();

            if ( poll_messages != null )
            {
                for ( int i = 0; i < poll_messages.length; i++ )
                {
                    if ( processMessage( poll_messages[ i ] ) )
                        polling = false;
                }
            } // if messages

            Thread.sleep( 1000 );
        }

        catch ( Exception ex )
        {
            polling = false;
        }
    } // while polling
} // run
} // PollerThread

// Class Methods
//



//-----
//  METHOD:          createFlagsT()  -
//***** ****

```

```

* @return           a default <tt>FlagsT</tt> value
***** */
public static FlagsT
createFlagsT()
{
    return
new FlagsT(
    false,
    HF_DYNAMIC.value | HF_DENSE.value,
    HF_HAZARD.value,
    new AuditT(
        "Detailed CORBA Services Example",
        "Mojo the Analyst",
        "Unclassified",
        "4.0.1",
        "Today"
    )
);
} // createFlagsT

//-----
// METHOD:          createLimitT()          -
//*****
* @return           a default <tt>LimitT</tt> value
***** */
public static LimitT
createLimitT()
{
    return new LimitT( 20000, 25000, 1000 );
} // createLimitT

//-----
// METHOD:          createModelIncident()      -
//*****
* Creates a model incident object suitable for the
* ChemBioWeapon source model and as input for the <tt>initIncident()</tt>
* call. Note is necessary to initialize one of these because CORBA
* doesn't like nulls. Thus, all objects must be created. Strings
* must be the blank string, and arrays must be zero-length.
*
* @return           a default <tt>ChemBioWeaponIncidentT</tt> object
***** */
public static ChemBioWeaponIncidentT
createModelIncident()
{
    // Instead of calling the explicit constructor, we could
    // call the default constructor and explicitly set the
    // fModelVersion, fAgent, fDeliverySystem, and fMunition
    // fields
    //

    return
}

```

```

new ChemBioWeaponIncidentT(
    0,                                // fModelType
    "",                               // fModelVersion
    0,                                // fSeed
    0.f,                             // fAgentPurity
    0.f,                             // fBuoyancy
    0.f,                             // fEfficiency
    0.f,                             // fFallAngle
    0.f,                             // fHeading
    0.f,                             // fHob
    0.f,                             // fHorzUncertainty
    0.f,                             // fISize
    0.f,                             // fLength
    0.f,                             // fLiquidDryFraction
    0.f,                             // fMass
    0.f,                             // fMmd
    0.f,                             // fMomentum
    0,                                // fNumber
    0.f,                             // fReleaseMass
    0.f,                             // fSigmaD
    0.f,                             // fSpeed
    0.f,                             // fSpread
    0.f,                             // fVaporFraction
    0.f,                             // fVertUncertainty
    "",                               // fAgent
    "",                               // fDeliverySystem
    ""                                // fMunition
);
} // createModelIncident

-----
//      METHOD:          createOptionsT()          -
//*****************************************************************************
* @return           a default <tt>OptionsT</tt> value
//*****************************************************************************
public static OptionsT
createOptionsT()
{
    return
    new OptionsT(
        11, 2, 0, 1.e36f, 1.e-20f, 1.e36f,
        0.01f, 0.0004f, 10.0f, 0.25f, 1000.0f, 0.0f, 1.e36f, ""
    );
} // createOptionsT

-----
//      METHOD:          createSpatialDomainT()        -
//*****************************************************************************
* @return           a default <tt>SpatialDomainT</tt> value
//*****************************************************************************
public static SpatialDomainT

```

```

createSpatialDomainT()
{
    return
    new SpatialDomainT(
        true,
        -1.e36f, -1.e36f, -1.e36f, -1.e36f,
        1.e36f, 1.e36f, 1.e36f
    );
} // createSpatialDomainT

//-----
// METHOD:           createTemporalDomainT() -
// ****
* @return          a default <tt>TemporalDomainT</tt> value
* ****
public static TemporalDomainT
createTemporalDomainT()
{
    return
    new TemporalDomainT(
        true,
        new TimeT( (short)-65535, (short)-65535, (short)-65535, 0.0f ),
        new TimeT( (short)-65535, (short)-65535, (short)-65535, 4.0f )
    );
} // createTemporalDomainT

//-----
// METHOD:           createWeatherT() -
// ****
* @return          a default <tt>WeatherT</tt> value specifying
*                  fixed 10 mph winds from 180 degrees
* ****
public static WeatherT
createWeatherT()
{
    return
    new WeatherT(
        new MetFlagsT(
            HD_LATLON.value, false, HO_OUTPUT.value | HO_2D.value,
            0, -1.0e36f, 10.0f
        ),
        new MetMetT(
            HW_METFIX.value, 0, 0, 3600.0f, 0, 0,
            HU MPH.value, HU_DEG.value,
            10.0f, 180.0f,
            new String[ 0 ],
            FileReferenceTMgr.createDeferred(),
            FileReferenceTMgr.createDeferred()
        ),
        new MetBLT(
            HB_BLCALC.value, MST_NORMAL.value,

```

```

        1000.0f, 50.0f, 50.0f, 0.0f, 1.0f, 0.1f, 0.16f, 0.6f, 0.0f, 0.0f
    ),
    new MetLSVT( HL_LSVOPER.value, 0.0f, 100.0f ),
    new MetPrecipT( HP_PRCPINP.value, 0 ),
    new TerrainT(
        HT_SCIPUFF.value,
        new TerrainMCT(
            0.0f,
            new int[]{ 5, 5 },
            new float[]{ 1.e-2f, 1.e-2f },
            new float[]{ 1.e-2f, 1.0f },
            0,
            new float[ 0 ]
        ),
        FileReferenceTMgr.createDeferred()
    )
);
} // createWeatherT

//-----
//      METHOD:          main()          -
//*****-----*****-----*****-----*****-----*****-----*****-----*****/
public static void
main( String[] argv )
{
    boolean standalone = false;
    String
        incident_id = "xxxxxx",
        user_name = "007",
        project_name = "detail-example";

        // Initialize server references so we can know to
        // terminate and release when we're done
    ChemBioWeaponServer cbwpn_server = null;
    ChemBioWeaponServerFactory cbwpn_factory = null;

    ScipuffServerFactory scipuff_factory = null;
    ScipuffServer scipuff_server = null;
    DispersionCalculator calculator = null;
    PlotGenerator plot_gen = null;
    PollerThread poll_thread = null;

    try
    {
        // Get naming context, list service names for funzies
        //
        Context context = new InitialContext();

        // listBindings() returns Binding [getObject(), getClassName()]
        System.out.println( "[DetailClient] registered names" );
        for ( NamingEnumeration en = context.list( "" ); en.hasMore(); )

```

```

{
    NameClassPair pair = (NameClassPair)en.next();

    System.err.println(
        new StringBuffer( 128 ).
        append( " " ).append( pair.getName() ) .
        append( ", " ).append( pair.getClassName() ) .
        toString()
    );
}

// Get CBWPN server instance
//
cbwpn_factory = ChemBioWeaponServerFactoryHelper.narrow(
    (org.omg.CORBA.Object)
    context.lookup( CHEMBIOWEAPON_FACTORY_SERVICE_NAME.value )
);

cbwpn_server = ChemBioWeaponServerHelper.narrow(
    cbwpn_factory.getInstance( user_name, project_name, incident_id )
);
// Create CWPN model incident ready for
// initIncident() call
//
ORB singleton_orb = ORB.init();
Any model_incident_any = singleton_orb.create_any();
ChemBioWeaponIncidentT model_incident = createModelIncident();

ChemBioWeaponIncidentTHelper.insert( model_incident_any, model_incident );

// Create an IncidentT, must populate time and
// location (input) fields
//
IncidentT incident = new IncidentT();

incident.fAbsoluteTime =
    new TimeT( (short)2002, (short)10, (short)1, 1200.0f );
incident.fID = incident_id;
incident.fLLAlocation = new float[]{ -84.8f, 33.4f, 0.0f };
incident.fmodelName = "CBFacWeapon";
incident.fModelServiceName = CHEMBIOWEAPON_FACTORY_SERVICE_NAME.value;
incident.fName = "cbwpn incident";
incident.fReleaseList = new ReleaseT[ 0 ];

// Call initIncident() and updateIncident()
// (updateIncident() should be a redundant call if
// we don't change any model incident or incident
// parameters)
//
IncidentTHolder incident_holder = new IncidentTHolder( incident );
AnyHolder model_incident_holder = new AnyHolder( model_incident_any );

cbwpn_server.initIncident( incident_holder, model_incident_holder );
System.err.println(

```

```

"\n[DetailClient] after initIncident(), model incident\n" +
toString(
    ChemBioWeaponIncidentTHelper.
    extract( model_incident_holder.value )
)
);

cbwpn_server.updateIncident( 0, incident_holder, model_incident_holder );
incident = incident_holder.value;
model_incident =
    ChemBioWeaponIncidentTHelper.extract( model_incident_holder.value );
System.err.println(
    "\n[DetailClient] after updateIncident(), model incident\n" +
    toString( model_incident )
);

// For some reason, CBWPN and some other models
// (e.g., CBFAC) create releases with the status
// array one element short; this should be resolved
// in the next version, but for 4.0.1, it is
// necessary to add the missing status element
//

FixStatusArray:
{
    for ( int i = 0; i < incident.fReleaseList.length; i++ )
    {
        int[] status_array = incident.fReleaseList[ i ].fStatus;
        int[] new_array = new int[ RSI_INST_STATUS_COUNT.value ];

        System.err.println(
            new StringBuffer( 128 ).
            append( "[DetailClient] release status array length=" ).
            append( status_array.length ).
            append( ", RSI_INST_STATUS_COUNT=" ).
            append( RSI_INST_STATUS_COUNT.value ).
            append( ", fixing" ).
            toString()
        );
        for ( int j = 0; j < new_array.length; j++ )
            new_array[ j ] = RS_VALID.value;
        System.arraycopy(
            status_array, 0, new_array, 0, status_array.length
        );
        incident.fReleaseList[ i ].fStatus = new_array;
    } // for each release
}

// Feedback response
//
System.err.println( "[DetailClient]\n" + toString( incident ) );

// Gather inputs for the T&D calculation
//

```

```

ChemBioWeaponIncidentTHelper.insert( model_incident_any, model_incident );

    TemporalDomainTHolder temporal_holder =
        new TemporalDomainTHolder( createTemporalDomainT() );
    SpatialDomainTHolder spatial_holder =
        new SpatialDomainTHolder( createSpatialDomainT() );

            // Get a ScipuffServer
            //
scipuff_factory = ScipuffServerFactoryHelper.narrow(
    (org.omg.CORBA.Object)
    context.lookup( SCIPUFF_FACTORY_SERVICE_NAME.value )
);

scipuff_server = ScipuffServerHelper.narrow(
    scipuff_factory.getInstance( user_name, project_name )
);

            // Must start polling before anything else to
            // avoid deadlock if the server sends an HM_REPLY
            // message
            //
poll_thread = new PollerThread( scipuff_server );
poll_thread.start();

            // Run calculation
            //
System.err.println( "\n[DetailClient] calculating..." );
calculator = scipuff_server.getDispersionCalculator();
calculator.calculate2(
    new IncidentT[]{ incident },
    new Any[]{ model_incident_any },
    createWeatherT(),
    createLimitT(),
    createOptionsT(),
    createFlagsT(),
    temporal_holder,
    spatial_holder,
    300.0f, 0.25f
);

            // Wait for calculation to complete
            // (poll thread is running)
            //
poll_thread.join();
System.err.println(
    new StringBuffer( "\n[DetailClient] calculation complete with " ).append(
        poll_thread.getStatus() == HPAC_SUCCESS.value ?
        "SUCCESS" : "FAILURE"
    ).toString()
);

```

```

        // Must terminate DispersionCalculator prior to
        // getting a PlotGenerator
        //
System.err.println( "[DetailClient] terminating calculator" );
try { calculator.terminateDispersionCalculator(); }
catch ( Exception ex ) {}
finally
{
    calculator = null;
}

        // On success, we want plot data
        //
if ( poll_thread.getStatus() == HPAC_SUCCESS.value )
{
    // Restart polling thread <b>before</b>
    // calling getPlotGenerator()
    //
poll_thread.start();

        // Get a PlotGenerator
        //
System.err.println( "\n[DetailClient] retrieving plot generator..." );
plot_gen = scipuff_server.
getPlotGenerator( 25000, new IncidentT[ 0 ], new Any[ 0 ] );

        // Create plot data objects
        //
IntHolder rad_times = new IntHolder();
StringListHolder
    class_names = new StringListHolder(),
    choice_names = new StringListHolder(),
    kind_names = new StringListHolder();
CategoryClass2DHolder cat_classes = new CategoryClass2DHolder();
ClassChoice2DHolder class_choices = new ClassChoice2DHolder();
HPACFieldCoordinateTHolder
    project_coords = new HPACFieldCoordinateTHolder();
HPACTimeTListHolder
    puff_times = new HPACTimeTListHolder(),
    surface_times = new HPACTimeTListHolder(),
    met_times = new HPACTimeTListHolder();

        // Retrieve plot classes/choices/kinds
        // Note we don't have to call for the counts
        // first to allocate arrays because
        // ScipuffServer nicely does all that for us;
        // Note the 2D arrays are column major because
        // they come from Fortran
        //
plot_gen.getPlotClasses(
    class_names, choice_names, kind_names,
    cat_classes, class_choices, project_coords
}

```

```

) ;
        // Retrieve plot times
        //
plot_gen.getPlotTimes(
    puff_times, surface_times, met_times, rad_times
);

        // Dump the plot info
        //
System.out.println( "\n[PLOT INFO]\n  class names:" );
for ( int i = 0; i < class_names.value.length; i++ )
    System.out.println( "      " + class_names.value[ i ].trim() );

System.out.println( "  choice names:" );
for ( int i = 0; i < choice_names.value.length; i++ )
    System.out.println( "      " + choice_names.value[ i ].trim() );

System.out.println( "  kind names:" );
for ( int i = 0; i < kind_names.value.length; i++ )
    System.out.println( "      " + kind_names.value[ i ].trim() );

System.out.println( "  puff times:" );
for ( int i = 0; i < puff_times.value.length; i++ )
    System.out.println( "      " + toString( puff_times.value[ i ] ) );

System.out.println( "  surface times:" );
for ( int i = 0; i < surface_times.value.length; i++ )
    System.out.println( "      " + toString( surface_times.value[ i ] ) );

System.out.println( "  met times:" );
for ( int i = 0; i < met_times.value.length; i++ )
    System.out.println( "      " + toString( met_times.value[ i ] ) );

System.out.println( "  rad times:" + rad_times.value );

        // Column major
        //
System.out.println( "  category-class availability:" );
for ( int i = 0; i < cat_classes.value[ 0 ].length; i++ )
{
    System.out.println( "      category " + PLOT_CATEGORY_NAMES[ i ] );
    for ( int j = 0; j < cat_classes.value.length; j++ )
    {
        System.out.println(
            new StringBuffer( 80 ).append( "      " ).append( class_names.value[ j ].trim() ).append( "=" ).append( toString( cat_classes.value[ j ][ i ] ) ).toString()
        );
    } // for each class
} // for each category

```

```

                // Column major
                //
System.out.println( "  class-choice availability:" );
for ( int i = 0; i < class_choices.value[ 0 ].length; i++ )
{
    System.out.println( "      class " + class_names.value[ i ].trim() );
    for ( int j = 0; j < class_choices.value.length; j++ )
    {
        System.out.println(
            new StringBuffer( 80 ).
            append( "          " ).append( choice_names.value[ j ].trim() ).
            append( "=" ).
            append( toString( class_choices.value[ j ][ i ] ) ).
            toString()
        );
    } // for each choice
} // for each class

                // Build input objects for a call to
                // createField()
                //
System.err.println( "[DetailClient] creating plot field..." );

HPACPlotFieldT plot_field =
    new HPACPlotFieldT(
        HP_SURF.value,           // category index (1-based)
        1,                      // class index (1-based)
        1,                      // choice index (1-based)
        1,                      // field index (1-based)
        surface_times.value.length, // time index
        0.f,                    // user time
        0,                      // hazard (false)
        0,                      // max cells
        -1,                     // max refinement levels
        -1,                     // actual refinement levels (out)
        0.f,                    // resolution (out)
        0,                      // interpolation type (out)
                                // coords (out), but objects cannot
                                // be null or CORBA barfs
        new HPACFieldCoordinateT(
            HD_LATLON.value, 0,
            new ReferenceT(),
            new HPACSliceT( 0, new HPACPointT(), new HPACPointT() )
        ),
        "",                      // units (out)
        project_name
    );
HPACPlotFieldTHolder plot_field_holder =
    new HPACPlotFieldTHolder( plot_field );

                // Create the plot field
                //
int sag_id =

```

```

        plot_gen.createField( plot_field_holder, new float[ 0 ] );

plot_field = plot_field_holder.value;

        // Build input objects for calling
        // contourField()
        //
System.err.println( "[DetailClient] contouring plot field..." );

HPACPlotTypeT plot_type =
    new HPACPlotTypeT( PLOTTYPE_mean, 0, HP_ON.value );

        // Contour values with the exception of the
        // first are copied from the GB material file:
        // <hpac-root>/server/data/materials/gb.mtl
        //
HPACContourElementT[] contour_elements =
{
    new HPACContourElementT( 0.05f, 0.f, "Very Mild", 0.f, 0.f ),
    new HPACContourElementT( 0.3f, 0.f, "Mild_ECt10", 0.f, 0.f ),
    new HPACContourElementT( 0.5f, 0.f, "Mild_ECt50", 0.f, 0.f ),
    new HPACContourElementT( 25.f, 0.f, "Sev_ICt50", 0.f, 0.f ),
    new HPACContourElementT( 35.f, 0.f, "LCt50", 0.f, 0.f ),
    new HPACContourElementT( 45.f, 0.f, "LCt90", 0.f, 0.f ),
};

HPACContourElementTListHolder contour_list_holder =
    new HPACContourElementTListHolder( contour_elements );

HPACContourHeaderT contour_header =
    new HPACContourHeaderT(
        contour_elements.length, 16667.0f,
        PLOT_ON.value, PLOT_LIN.value, "mg-min/m3"
    );

HPACLineTListHolder line_holder = new HPACLineTListHolder();
HPACPointTListHolder point_holder = new HPACPointTListHolder();

        // Contour the field
        //
plot_gen.contourField(
    sag_id, plot_field, plot_type,
    contour_header, contour_list_holder,
    CLOSE_CONTOUR.value | LATLON_OUTPUT.value,
    line_holder, point_holder
);

        // Dump the contour polygons
        //
System.out.println( "\n[POLYGONS]" );
for ( int i = 0; i < line_holder.value.length; i++ )
{
    HPACLineT line = line_holder.value[ i ];
}

```

```

        System.out.println(
            new StringBuffer( 128 ).
            append( "line " ).append( i ).
            append( "=" ).append( toString( line ) ).
            toString()
        );

        for (
            int j = 0, pt_index = line.fstartIndex - 1;
            j < line.fNumberLocationPoints;
            j++, pt_index++
        )
    {
        System.out.println(
            " " + toString( point_holder.value[ pt_index ] )
        );
    } // for each point
} // for each line

        // Quit polling
        //

poll_thread.join();
} // if calculation succeeded
} // try

catch ( Exception ex )
{
    System.err.println( "[DetailClient] " + ex );
    ex.printStackTrace( System.err );
}

finally
{
    // It is critical that server objects be
    // closed and released to avoid cruft on the server
    //
    if ( poll_thread != null )
        poll_thread.interrupt();

    if ( cbwpn_factory != null && ! standalone )
        cbwpn_factory._release();

    if ( cbwpn_server != null )
    {
        try
        {
            cbwpn_server.terminate();

            if ( ! standalone )
                cbwpn_server._release();
        }
        catch ( Exception ex ) {}
    }
}

```

```

}

if ( calculator != null )
{
    System.err.println( "[DetailClient] terminating calculator" );
    try { calculator.terminateDispersionCalculator(); }
    catch ( Exception ex ) {}
    finally
    {
        if ( ! standalone )
            calculator._release();
    }
}

if ( plot_gen != null )
{
    System.err.println( "[DetailClient] terminating plot generator" );
    try { plot_gen.terminatePlotGenerator(); }
    catch ( Exception ex ) {}
    finally
    {
        if ( ! standalone )
            plot_gen._release();
    }
}

if ( scipuff_server != null )
{
    System.err.println( "[DetailClient] terminating scipuff server" );
    try { scipuff_server.terminate(); }
    catch ( Exception ex ) {}
    finally
    {
        if ( ! standalone )
            scipuff_server._release();
    }
}

if ( scipuff_factory != null && ! standalone )
    scipuff_factory._release();

System.exit( 0 );
} // finally
} // main

//-----
//      METHOD:          toString()
//*****-
//*****-
public static String
toString( float[] loc )
{

```

```

StringBuffer buffer = new StringBuffer( 64 );

for ( int i = 0; i < loc.length; i++ )
    buffer.append( loc[ i ] ).append( "," );

return buffer.toString();
} // toString

//-----
// METHOD:          toString()
// ****
public static String
toString( ChemBioWeaponIncidentT data )
{
    return
        new StringBuffer( 4096 ).
        append( "ChemBioWeaponIncidentT:" ).
        append( "\nfAgent=" ).append( data.fAgent ).
        append( "\nfAgentPurity=" ).append( data.fAgentPurity ).
        append( "\nfBuoyancy=" ).append( data.fBuoyancy ).
        append( "\nfDeliverySystem=" ).append( data.fDeliverySystem ).
        append( "\nfEfficiency=" ).append( data.fEfficiency ).
        append( "\nfFallAngle=" ).append( data.fFallAngle ).
        append( "\nfHeading=" ).append( data.fHeading ).
        append( "\nfHob=" ).append( data.fHob ).
        append( "\nfHorzUncertainty=" ).append( data.fHorzUncertainty ).
        append( "\nfISize=" ).append( data.fISize ).
        append( "\nfLength=" ).append( data.fLength ).
        append( "\nfLiquidDryFraction=" ).append( data.fLiquidDryFraction ).
        append( "\nfMass=" ).append( data.fMass ).
        append( "\nfMmd=" ).append( data.fMmd ).
        append( "\nfModelType=" ).append( data.fModelType ).
        append( "\nfModelVersion=" ).append( data.fModelVersion ).
        append( "\nfMomentum=" ).append( data.fMomentum ).
        append( "\nfMunition=" ).append( data.fMunition ).
        append( "\nfNumber=" ).append( data.fNumber ).
        append( "\nfReleaseMass=" ).append( data.fReleaseMass ).
        append( "\nfSeed=" ).append( data.fSeed ).
        append( "\nfSigmaD=" ).append( data.fSigmaD ).
        append( "\nfSpeed=" ).append( data.fSpeed ).
        append( "\nfSpread=" ).append( data.fSpread ).
        append( "\nfVaporFraction=" ).append( data.fVaporFraction ).
        append( "\nfVertUncertainty=" ).append( data.fVertUncertainty ).
        toString();
} // toString

//-----
// METHOD:          toString()
// ****

```

```

public static String
toString( HPACCCategoryClassT value )
{
    return
new StringBuffer( 64 ).  

    append( value.fIsAvailable ).append( "," ).  

    append( value.fIsTypeRequired ).  

    toString();
} // toString

//-----  

//      METHOD:          toString()          -  

//*****  

*****  

public static String
toString( HPACClassChoiceT value )
{
    return
new StringBuffer( 64 ).  

    append( value.fIsAvailable ).append( "," ).  

    append( value.fSupportsKindSelection ).append( "," ).  

    append( value.fKindIndex ).append( "," ).  

    append( value.fNumberKinds ).append( "," ).  

    append( value.fTimeId ).append( "," ).  

    append( value.fSupportsUserTime ).  

    toString();
} // toString

//-----  

//      METHOD:          toString()          -  

//*****  

*****  

public static String
toString( HPACLineT value )
{
    return
new StringBuffer( 128 ).  

    append( "index:" ).append( value.fContourIndex ).append( "," ).  

    append( "start:" ).append( value.fstartIndex ).append( "," ).  

    append( "count:" ).append( value.fNumberLocationPoints ).append( "," ).  

    append( "sense:" ).  

    append(
        value.fLineSense == HP_LEFTHAND.value ? "left" :
        value.fLineSense == HP_RIGHTHAND.value ? "right" :
        "open"
    ).  

    toString();
} // toString

//-----  


```

```

//      METHOD:          toString()           -
//****************************************************************************
*****public static String
toString( HPACPointT value )
{
    return
new StringBuffer( 64 ).append( value.fX ).append( "," ).append( value.fY ).toString();
} // toString

//-----
//      METHOD:          toString()           -
//****************************************************************************
*****public static String
toString( HPACTimeT value )
{
    return
new StringBuffer( 128 ).append( "\u0028" ).append( toString( value.fTime ) ).append( "\u0029," ).append( value.fNumberItems ).append( "," ).append( value.fTimeDisplay.trim() ).toString();
} // toString

//-----
//      METHOD:          toString()           -
//****************************************************************************
*****public static String
toString( IncidentT incident )
{
    StringBuffer buffer = new StringBuffer( 4096 );

    buffer.append( "Incident:" ).append( "\n fAbsoluteTime=" ).append( toString( incident.fAbsoluteTime ) ).append( "\n fAvailableEffects=" ).append( incident.fAvailableEffects ).append( "\n fHasCustomMaterials=" ).append( incident.fHasCustomMaterials ).append( "\n fHasCustomReleases=" ).append( incident.fHasCustomReleases ).append( "\n fID=" ).append( incident.fID ).append( "\n fLLALocation=" ).append( toString( incident.fLLALocation ) ).append( "\n fmodelName=" ).append( incident.fmodelName ).append( "\n fModelServiceName=" ).append( incident.fModelServiceName ).append( "\n fName=" ).append( incident.fName );
}

```

```

append( "\n  fReleaseList=\n" );

for ( int i = 0; i < incident.fReleaseList.length; i++ )
{
    ReleaseT release = incident.fReleaseList[ i ];
    int type = release.fReleaseData.discriminator();
    int mat_type = release.fMaterial.fMaterialData.discriminator();

    buffer.
        append( "[" ).append( i ).append( "]" ).
        append( "\n      fHorzSize=" ).append( release.fHorzSize ).
        append( "\n      fHorzUncertainty=" ).
            append( release.fHorzUncertainty ).
        append( "\n      fID=" ).append( release.fID ).
        append( "\n      fIncidentID=" ).append( release.fIncidentID ).
        append( "\n      fLLALocation=" ).
            append( toString( release.fLLALocation ) ).
        append( "\n      fLocationGroup=" ).append( release.fLocationGroup ).
        append( "\n      fMaterial isa " ).
            append(
                mat_type == HM_AEROSOL.value ? "AerosolMaterialT" :
                mat_type == HM_GAS.value ? "GasMaterialT" :
                mat_type == HM_LIQUID.value ? "LiquidMaterialT" :
                mat_type == HM_PARTICLE.value ? "ParticleMaterialT" :
                "*** unknown ***"
            ).
        append( "\n      fMaterial.fID=" ).append( release.fMaterial.fID ).
        append( "\n      fMaterial.fTypeMask=" ).
            append( release.fMaterial.fTypeMask ).
        append( "\n      fMaterial fName=" ).
            append( release.fMaterial.fName ).
        append( "\n      fMaterial.fLongName=" ).
            append( release.fMaterial.fLongName ).
        append( "\n      fMaterial.fUnits=" ).
            append( release.fMaterial.fUnits ).
        append( "\n      fMaterial.fAgentType=" ).
            append( release.fMaterial.fAgentType ).
        append( "\n      fMaterialCustomized=" ).
            append( release.fMaterialCustomized ).
        append( "\n      fPuffDuration=" ).append( release.fPuffDuration ).
        append( "\n" ).append( toString( release.fReleaseData ) ).
        append( "      fStatus=" );

    for ( int j = 0; j < release.fStatus.length; j++ )
        buffer.append( release.fStatus[ j ] ).append( "," );
}

buffer.
    append( "\n      fStartTime=" ).
        append( toString( release.fStartTime ) ).
    append( "\n      fVertSize=" ).append( release.fVertSize ).
    append( "\n      fVertUncertainty=" ).
        append( release.fVertUncertainty ).
    append( "\n" );

```

```

} // for each release

return buffer.toString();
} // toString

//-----  

// METHOD:           toString()          -  

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

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

public static String  

toString( ReleaseDataT data )  

{
    StringBuffer buffer = new StringBuffer( 512 );
    int type = data.discriminator();

    buffer.  

        append( "      fReleaseData isa " ).  

        append(  

            type == HR_CONTINUOUS.value ? "ContinuousReleaseT" :  

            type == HR_FILE.value ? "FileReleaseT" :  

            type == HR_INSTANTANEOUS.value ? "InstantaneousReleaseT" :  

            type == HR_MOVING.value ? "MovingReleaseT" :  

            type == HR_POOL.value ? "PoolReleaseT" :  

            type == HR_STACK.value ? "StackReleaseT" :  

            "*** unknown ***"  

        );
}

if ( type == HR_CONTINUOUS.value )
{
    ContinuousReleaseT cont = data.fContinuous();

    buffer.  

        append( "\n      fReleaseData.fBuoyancy=" ).append( cont.fBuoyancy ).  

        append( "\n      fReleaseData.fDistribution=" ).  

        append( cont.fDistribution ).  

        append( "\n      fReleaseData.fDryMassFraction=" ).  

        append( cont.fDryMassFraction ).  

        append( "\n      fReleaseData.fDuration=" ).  

        append( cont.fDuration ).  

        append( "\n      fReleaseData.fMassMeanDiameter=" ).  

        append( cont.fMassMeanDiameter ).  

        append( "\n      fReleaseData.fMassRate=" ).append( cont.fMassRate ).  

        append( "\n      fReleaseData.fMassSigma=" ).  

        append( cont.fMassSigma ).  

        append( "\n      fReleaseData.fMomentum=" ).append( cont.fMomentum ).  

        append( "\n      fReleaseData.fSigmaY=" ).append( cont.fSigmaY ).  

        append( "\n      fReleaseData.fSigmaZ=" ).append( cont.fSigmaZ );
} // if continuous

if ( type == HR_INSTANTANEOUS.value )
{
    InstantaneousReleaseT inst = data.fInstantaneous();
}

```

```

buffer.
    append( "\n      fReleaseData.fBuoyancy=" ).append( inst.fBuoyancy ).
    append( "\n      fReleaseData.fDistribution=" ).
        append( inst.fDistribution ).
    append( "\n      fReleaseData.fDryMassFraction=" ).
        append( inst.fDryMassFraction ).
    append( "\n      fReleaseData.fMass=" ).
        append( inst.fMass ).
    append( "\n      fReleaseData.fMassMeanDiameter=" ).
        append( inst.fMassMeanDiameter ).
    append( "\n      fReleaseData.fMassSigma=" ).
        append( inst.fMassSigma ).
    append( "\n      fReleaseData.fMomentum=" ).append( inst.fMomentum ).
    append( "\n      fReleaseData.fRandomCount=" ).
        append( inst.fRandomCount ).
    append( "\n      fReleaseData.fRandomSeed=" ).
        append( inst.fRandomSeed ).
    append( "\n      fReleaseData.fRandomSpread=" ).
        append( inst.fRandomSpread ).
    append( "\n      fReleaseData.fSigmaX=" ).append( inst.fSigmaX ).
    append( "\n      fReleaseData.fSigmaY=" ).append( inst.fSigmaY ).
    append( "\n      fReleaseData.fSigmaZ=" ).append( inst.fSigmaZ );
} // if continuous

buffer.append( "\n" );

return buffer.toString();
} // toString

-----
// METHOD:          toString()
***** /
***** /
public static String
toString( TimeT time )
{
    return
new StringBuffer( 64 ).  

    append( time.fYear ).append( "," ).  

    append( time.fMonth ).append( "," ).  

    append( time.fDay ).append( "," ).  

    append( time.fHour ).  

    toString();
} // toString
} // DetailClient

```

B.2 DetailClient.run.bat

```

@echo off
rem -----

```

```

rem -      NAME:          DetailClient.run.bat
rem -      PURPOSE:        -
rem -                  Windows batch script to run samples.DetailClient
rem -----
set JRE_BASE=d:\j2sdk1.4.0
rem set JRE_BASE=c:\jdk1.3.1

set path=%JRE_BASE%\bin;%path%
set JRE_HOME=%JRE_BASE%\jre

set classpath=..
set ExtFlags=-Djava.ext.dirs=c:\hpac4\shared\ext;c:\hpac4\server\ext;%JRE_HOME%\lib\ext

set S01=-Djava.naming.factory.initial=com.sun.jndi.cosnaming.CNCtxFactory
set S02=-Djava.naming.provider.url=iiop://localhost:1400/

java -Xms16m -classpath .. %ExtFlags% %S01% %S02% samples.DetailClient */

```

B.3 DetailClient.run.sh

```

#!/bin/sh -av
#-----
#-      NAME:          DetailClient.run.sh
#-      PURPOSE:        -
#-                  Bourne shell script to run samples.DetailClient
#-----

#      -- Check Command Line for a ServerURL
#      --
if [ $# -gt 0 ]; then
  ServerURL=$1
else
  ServerURL=iiop://localhost:1400/
fi

#      -- Point to J2SE
#      --
export JRE_HOME
JRE_HOME=/usr/local/j2sdk1.4.0/jre
# JRE_HOME=/usr/local/jdk1.3.1

export PATH
PATH=$JRE_HOME/bin:$PATH

export HPAC_DIR
HPAC_DIR=$HOME/src/hpacdev

java \
-Xms16m \
-classpath .. \
-Djava.ext.dirs=$HPAC_DIR/shared/ext:$HPAC_DIR/server/ext:$JRE_HOME/lib/ext \
-Djava.naming.factory.initial=com.sun.jndi.cosnaming.CNCtxFactory \

```

```
-Djava.naming.provider.url=$ServerURL \
samples.DetailClient $*
```

B.4 run.log

This is the standard error log from the execution of DetailClient containing progress and feedback messages.

```
[DetailClient] registered names
NwpnFactory, mil.dtra.hpac.models.nwpn.server._NwpnServerFactoryStub
RadFileMerger, mil.dtra.hpac.server.radfile._RadFileMergerStub
MTIMatServer, mil.dtra.models.CBFac.AgentsList._AgentsListStub
DamCatCalc, mil.dtra.models.CBFac.DamCat._DamCatStub
RWPNFFactory, com.sun.corba.se.internal.iiop.CDRInputStream_1_0$1
StcalcServer, mil.dtra.hpac.server.stcalc._StcalcServerStub
CloudTransCalc, mil.dtra.models.CBFac.CloudTrans._CloudTransStub
SmokeWeaponFactory, com.sun.corba.se.internal.iiop.CDRInputStream_1_0$1
NWIFactory, com.sun.corba.se.internal.iiop.CDRInputStream_1_0$1
SWFICalc, mil.dtra.models.CBFac.SWFI._SWFISTub
DWFICalc, mil.dtra.models.CBFac.DWFI._DWFISTub
MissileInterceptFactory, com.sun.corba.se.internal.iiop.CDRInputStream_1_0$1
ChemBioWeaponFactory, com.sun.corba.se.internal.iiop.CDRInputStream_1_0$1
ScipuffServer-1-1033486053939, mil.dtra.hpac.server.scipuff._ScipuffAllStub
CBFacServerFactory, mil.dtra.hpac.models.CBFac.server._CBFacServerFactoryStub
ScipuffServer-1-1033501455115, mil.dtra.hpac.server.scipuff._ScipuffAllStub
NfacFactory, mil.dtra.hpac.models.nfac.server._NfacServerFactoryStub
MaterialServer, mil.dtra.hpac.material.server._MaterialServerStub
WeaponServer, mil.dtra.models.CBFac.WeaponServer._WeaponListStub
ScipuffServerFactory, mil.dtra.hpac.server.scipuff._ScipuffServerFactoryStub

[DetailClient] after initIncident(), model incident
ChemBioWeaponIncidentT:
fAgent=GB
fAgentPurity=100.0
fBuoyancy=-1.0E36
fDeliverySystem=4 AC * 8 Bombs
fEfficiency=100.0
fFallAngle=0.0
fHeading=45.0
fHob=2.0
fHorzUncertainty=0.0
fISize=12.0
fLength=0.0
fLiquidDryFraction=40.0
fMass=100.0
fMmd=500.0
fModelType=1
fModelVersion=1.0
fMomentum=-1.0E36
fMunition=500 kg Bomb
fNumber=32
fReleaseMass=80.0
fSeed=18942363
```

```

fSigmaD=2.0
fSpeed=0.0
fSpread=400.0
fVaporFraction=40.0
fVertUncertainty=0.0

[DetailClient] after updateIncident(), model incident
ChemBioWeaponIncidentT:
fAgent=GB
fAgentPurity=100.0
fBuoyancy=-1.0E36
fDeliverySystem=4 AC * 8 Bombs
fEfficiency=100.0
fFallAngle=0.0
fHeading=45.0
fHob=2.0
fHorzUncertainty=0.0
fISize=12.0
fLength=0.0
fLiquidDryFraction=40.0
fMass=100.0
fMmd=500.0
fModelType=1
fModelVersion=1.0
fMomentum=-1.0E36
fMunition=500 kg Bomb
fNumber=32
fReleaseMass=80.0
fSeed=18942363
fSigmaD=2.0
fSpeed=0.0
fSpread=400.0
fVaporFraction=40.0
fVertUncertainty=0.0
[DetailClient] release status array length=21, RSI_INST_STATUS_COUNT=22, fixing
[DetailClient] release status array length=21, RSI_INST_STATUS_COUNT=22, fixing
[DetailClient]
Incident:
  fAbsoluteTime=2002,10,1,1200.0
  fAvailableEffects=0
  fHasCustomMaterials=false
  fHasCustomReleases=false
  fID=xxxxxx
  fLLALocation=-84.8,33.4,0.0,
  fmodelName=CBFacWeapon
  fModelServiceName=ChemBioWeaponFactory
  fName=cbwpn incident
  fReleaseList=
[0]
  fHorzSize=50.0
  fHorzUncertainty=0.0
  fID=first_phase-1033501481443
  fIncidentID=xxxxxx

```

```

fLLALocation=-84.8,33.4,2.0,
fLocationGroup=0
fMaterial isa LiquidMaterialT
fMaterial.fID=place holder
fMaterial.fTypeMask=1048580
fMaterial.fName=gb
fMaterial.fLongName=Sarin
fMaterial.fUnits=kg
fMaterial.fAgentType=CHM
fMaterialCustomized=false
fPuffDuration=4.0
fReleaseData isa InstantaneousReleaseT
fReleaseData.fBuoyancy=-1.0E36
fReleaseData.fDistribution=-1
fReleaseData.fDryMassFraction=-1.0E36
fReleaseData.fMass=40.0
fReleaseData.fMassMeanDiameter=5.0E-4
fReleaseData.fMassSigma=2.0
fReleaseData.fMomentum=-1.0E36
fReleaseData.fRandomCount=32
fReleaseData.fRandomSeed=18942363
fReleaseData.fRandomSpread=200.0
fReleaseData.fSigmaX=12.0
fReleaseData.fSigmaY=12.0
fReleaseData.fSigmaZ=12.0
fStatus=1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
fStartTime=2002,10,1,1200.0
fVertSize=5000.0
fVertUncertainty=0.0
[1]
fHorzSize=50.0
fHorzUncertainty=0.0
fID=vapor_phase-4603882208183887824
fIncidentID=xxxxxx
fLLALocation=-84.8,33.4,2.0,
fLocationGroup=0
fMaterial isa LiquidMaterialT
fMaterial.fID=place holder
fMaterial.fTypeMask=1048580
fMaterial.fName=gb
fMaterial.fLongName=Sarin
fMaterial.fUnits=kg
fMaterial.fAgentType=CHM
fMaterialCustomized=false
fPuffDuration=4.0
fReleaseData isa InstantaneousReleaseT
fReleaseData.fBuoyancy=-1.0E36
fReleaseData.fDistribution=0
fReleaseData.fDryMassFraction=-1.0E36
fReleaseData.fMass=40.0
fReleaseData.fMassMeanDiameter=-1.0E36
fReleaseData.fMassSigma=-1.0E36
fReleaseData.fMomentum=-1.0E36

```

```

fReleaseData.fRandomCount=32
fReleaseData.fRandomSeed=18942363
fReleaseData.fRandomSpread=200.0
fReleaseData.fSigmaX=12.0
fReleaseData.fSigmaY=12.0
fReleaseData.fSigmaZ=12.0
fStatus=1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
fStartTime=2002,10,1,1200.0
fVertSize=5000.0
fVertUncertainty=0.0

[DetailClient] calculating...
[Poller] polling
[Poller] polling

[Server Message] Info
Preparing to create project
Project=C:\hpac4\server\users\007\detail-example\detail-example
Validating input data

[Server Message] Info
Preparing to create project
Project=C:\hpac4\server\users\007\detail-example\detail-example
Deleting existing project files

[Server Message] Info
Creating project input files
Project=C:\hpac4\server\users\007\detail-example\detail-example
Writing INP file

[Server Message] Info

Writing SCN file

[Server Message] Info

Writing MSC file

[Server Message] Info
SCIPUFF starting creation of project files
Project=C:\hpac4\server\users\007\detail-example\detail-example

[Server Message] Info
Initializing SCIPUFF

[Server Message] Info
detail-example : Initializing

```

```
[Server Message] Info  
detail-example : Initializing  
Reading project input data
```

```
[Server Message] Info
```

```
Initializing data
```

```
[Server Message] Info  
Validating meteorology input
```

```
[Server Message] Info
```

```
Initializing meteorology parameters
```

```
[Server Message] Info
```

```
Reading meteorology input file
```

```
[Server Message] Info  
detail-example : Initializing  
Validating meteorology input  
Preparing meteorology
```

```
[Server Message] Info  
detail-example : Initializing
```

```
[Server Message] Info
```

```
Creating project output files
```

```
[Server Message] Info
```

```
Creating surface output files
```

```
[Server Message] Info
```

```
Creating surface deposition file
```

```
[Server Message] Info
```

```
Creating surface dosage file
```

```
[Server Message] Info
```

```
Creating project puff file
```

```
[Server Message] Info
```

```
Writing output file(s)
```

```
[Server Message] Info  
detail-example : Initialized
```

```
[Server Message] Info  
SCIPUFF finished creating project files  
Project=C:\hpac4\server\users\007\detail-example\detail-example
```

```
[Server Message] Info  
SCIPUFF starting dispersion calculation  
Project=C:\hpac4\server\users\007\detail-example\detail-example
```

```
[Server Message] Info  
Initializing SCIPUFF
```

```
[Server Message] Info  
detail-example : Initializing
```

```
[Server Message] Info
```

```
Reading project data files
```

```
[Server Message] Info
```

```
Initializing data
```

```
[Server Message] Info
```

```
Initializing meteorology
```

[Server Message] Info

Initializing meteorology parameters

[Server Message] Info

Reading meteorology input file

[Server Message] Info
detail-example : Initializing
Initializing meteorology
Preparing meteorology
[Poller] polling

[Server Message] Info
detail-example : Initializing
Initializing meteorology
Continuing meteorology preparation

[Server Message] Info
detail-example : Initializing

[Server Message] Info
detail-example : Initializing

[Server Message] Info

Initializing sources

[Poller] polling

[Server Message] Info

Preparing source 1 (Release 0 (Incident cbwpn incident)))

[Server Message] Info

Preparing source 2 (Release 1 (Incident cbwpn incident)))

[Server Message] Info

Initializing 672 new puffs


```
[Server Message] Info
detail-example : Calculating
01-NOV-02 07:02:13Z (5.00 min)
220 Puffs 2.34s timestep
[Poller] polling
[Poller] polling
[Poller] polling
[Poller] polling
[Poller] polling

[Server Message] Info
detail-example : Calculating
01-NOV-02 07:07:13Z (10.0 min)
46 Puffs 2.34s timestep
[Poller] polling

[Server Message] Info
detail-example : Writing output
```

```
[Server Message] Info
```

```
Writing puff file
```

```
[Server Message] Info
```

```
Writing project file
```

```
[Server Message] Info
```

```
Writing surface file(s)
```

```
[Server Message] Info
```

```
[Server Message] Info
detail-example : Calculating
01-NOV-02 07:12:13Z (15.0 min)
13 Puffs 18.8s timestep
[Poller] polling
```

```
[Server Message] Info
detail-example : Calculating
01-NOV-02 07:17:13Z (20.0 min)
10 Puffs 37.5s timestep
```

```
[Server Message] Info  
detail-example : Calculating  
01-NOV-02 07:22:13Z (25.0 min)  
9 Puffs 75.0s timestep
```

```
[Server Message] Info  
detail-example : Writing output
```

```
[Server Message] Info
```

```
Writing puff file
```

```
[Server Message] Info
```

```
Writing project file
```

```
[Server Message] Info
```

```
Writing surface file(s)
```

```
[Server Message] Info
```

```
[Server Message] Info  
detail-example : Calculating  
01-NOV-02 07:27:13Z (30.0 min)  
12 Puffs 75.0s timestep
```

```
[Server Message] Info  
detail-example : Calculating  
01-NOV-02 07:32:13Z (0.583 hr)  
14 Puffs 150.0s timestep
```

```
[Server Message] Info  
detail-example : Calculating  
01-NOV-02 07:37:13Z (0.667 hr)  
15 Puffs 150.0s timestep
```

```
[Server Message] Info  
detail-example : Writing output
```

```
[Server Message] Info
```

Writing puff file

[Server Message] Info

Writing project file

[Server Message] Info

Writing surface file(s)

[Server Message] Info

[Server Message] Info
detail-example : Calculating
01-NOV-02 07:42:13Z (0.750 hr)
13 Puffs 150.0s timestep

[Server Message] Info
detail-example : Calculating
01-NOV-02 07:47:13Z (0.833 hr)
14 Puffs 150.0s timestep

[Server Message] Info
detail-example : Calculating
01-NOV-02 07:52:13Z (0.917 hr)
13 Puffs 300.0s timestep

[Server Message] Info
detail-example : Writing output

[Server Message] Info

Writing puff file

[Server Message] Info

Writing project file

[Server Message] Info

Writing surface file(s)

[Server Message] Info

```
[Poller] polling

[Server Message] Info
detail-example : Calculating
01-NOV-02 07:57:13Z (1.00 hr)
10 Puffs 300.0s timestep

[Server Message] Info
detail-example : Calculating
01-NOV-02 08:02:13Z (1.08 hr)
8 Puffs 300.0s timestep

[Server Message] Info
detail-example : Calculating
01-NOV-02 08:07:13Z (1.17 hr)
8 Puffs 300.0s timestep

[Server Message] Info
detail-example : Writing output

[Server Message] Info

Writing puff file

[Server Message] Info

Writing project file

[Server Message] Info

Writing surface file(s)

[Server Message] Info

[Server Message] Info
detail-example : Calculating
01-NOV-02 08:12:13Z (1.25 hr)
10 Puffs 300.0s timestep

[Server Message] Info
detail-example : Calculating
01-NOV-02 08:17:13Z (1.33 hr)
```

```
9 Puffs  300.0s timestep

[Server Message] Info
detail-example : Calculating
01-NOV-02 08:22:13Z (1.42 hr)
9 Puffs  300.0s timestep

[Server Message] Info
detail-example : Writing output

[Server Message] Info

Writing puff file

[Server Message] Info

Writing project file

[Server Message] Info

Writing surface file(s)

[Server Message] Info

[Server Message] Info
detail-example : Calculating
01-NOV-02 08:27:13Z (1.50 hr)
10 Puffs  300.0s timestep

[Server Message] Info
detail-example : Calculating
01-NOV-02 08:32:13Z (1.58 hr)
10 Puffs  300.0s timestep

[Server Message] Info
detail-example : Calculating
01-NOV-02 08:37:13Z (1.67 hr)
6 Puffs  300.0s timestep

[Server Message] Info
detail-example : Writing output

[Server Message] Info
```

Writing puff file

[Server Message] Info

Writing project file

[Server Message] Info

Writing surface file(s)

[Server Message] Info

[Server Message] Info
detail-example : Calculating
01-NOV-02 08:42:13Z (1.75 hr)
8 Puffs 300.0s timestep

[Server Message] Info
detail-example : Calculating
01-NOV-02 08:47:13Z (1.83 hr)
5 Puffs 300.0s timestep

[Server Message] Info
detail-example : Calculating
01-NOV-02 08:52:13Z (1.92 hr)
5 Puffs 300.0s timestep

[Server Message] Info
detail-example : Writing output

[Server Message] Info

Writing puff file

[Server Message] Info

Writing project file

[Server Message] Info

Writing surface file(s)

```
[Server Message] Info
```

```
[Server Message] Info
detail-example : Calculating
01-NOV-02 08:57:13Z (2.00 hr)
4 Puffs 300.0s timestep
```

```
[Server Message] Info
detail-example : Calculating
01-NOV-02 09:02:13Z (2.08 hr)
3 Puffs 300.0s timestep
```

```
[Server Message] Info
detail-example : Calculating
01-NOV-02 09:07:13Z (2.17 hr)
3 Puffs 300.0s timestep
```

```
[Server Message] Info
detail-example : Writing output
```

```
[Server Message] Info
```

```
Writing puff file
```

```
[Server Message] Info
```

```
Writing project file
```

```
[Server Message] Info
```

```
Writing surface file(s)
[Poller] polling
```

```
[Server Message] Info
```

```
[Server Message] Info
detail-example : Calculating
01-NOV-02 09:12:13Z (2.25 hr)
4 Puffs 300.0s timestep
```

```
[Server Message] Info
```

```
detail-example : Calculating
01-NOV-02 09:17:13Z (2.33 hr)
2 Puffs 300.0s timestep

[Server Message] Info
detail-example : Calculating
01-NOV-02 09:22:13Z (2.42 hr)
2 Puffs 300.0s timestep

[Server Message] Info
detail-example : Writing output
```

```
[Server Message] Info
```

```
Writing puff file
```

```
[Server Message] Info
```

```
Writing project file
```

```
[Server Message] Info
```

```
Writing surface file(s)
```

```
[Server Message] Info
```

```
[Server Message] Info
detail-example : Calculating
01-NOV-02 09:27:13Z (2.50 hr)
1 Puffs 300.0s timestep
```

```
[Server Message] Info
detail-example : Calculating
01-NOV-02 09:32:13Z (2.58 hr)
1 Puffs 300.0s timestep
```

```
[Server Message] Info
detail-example : Calculating
01-NOV-02 09:37:13Z (2.67 hr)
1 Puffs 300.0s timestep
```

```
[Server Message] Info
detail-example : Writing output
```

```
[Server Message] Info
```

```
Writing puff file
```

```
[Server Message] Info
```

```
Writing project file
```

```
[Server Message] Info
```

```
Writing surface file(s)
```

```
[Server Message] Info
```

```
[Server Message] Info  
detail-example : Calculating  
01-NOV-02 09:42:13Z (2.75 hr)  
1 Puffs 300.0s timestep
```

```
[Server Message] Info  
Stopping run  
No more puffs at Time = 2.833 hrs
```

```
[Server Message] Info
```

```
Writing puff file
```

```
[Server Message] Info
```

```
Writing project file
```

```
[Server Message] Info
```

```
Writing surface file(s)
```

```
[Server Message] Info
```

```
[Server Message] Info  
Outputting wind field at end of run
```

```
[Server Message] Info
SCIPUFF finished dispersion calculation
Project=C:\hpac4\server\users\007\detail-example\detail-example

[DetailClient] calculation complete with SUCCESS
[DetailClient] terminating calculator

[DetailClient] retrieving plot generator...
[DetailClient] creating plot field...
[DetailClient] contouring plot field...
[DetailClient] terminating plot generator
[DetailClient] terminating scipuff server
```

B.5 run.out

This is standard output listing the plot data from the execution of DetailClient.

```
[PLOT INFO]
class names:
    Surface Dosage
    Surface Deposition
    Concentration
    Dosage Duration
    CODA Human Performance
    Probability of Injury
    Probability of Mortality
    Casualty Table
choice names:
    GB
kind names:
    Vapor
    Liquid
    Total
    Performance
    Performance Decrement
    Probability if Hit
    Probability
    Probability if Hit
    Probability
    Casualty Table
puff times:
    (2002,10,1,7.203529),13,01-Nov-02 07:12:13Z
    (2002,10,1,7.453529),12,01-Nov-02 07:27:13Z
    (2002,10,1,7.703529),13,01-Nov-02 07:42:13Z
    (2002,10,1,7.953529),10,01-Nov-02 07:57:13Z
    (2002,10,1,8.203529),10,01-Nov-02 08:12:13Z
    (2002,10,1,8.453529),10,01-Nov-02 08:27:13Z
    (2002,10,1,8.703529),8,01-Nov-02 08:42:13Z
```

```

(2002,10,1,8.953529),4,01-Nov-02 08:57:13Z
(2002,10,1,9.203529),4,01-Nov-02 09:12:13Z
(2002,10,1,9.453529),1,01-Nov-02 09:27:13Z
(2002,10,1,9.703529),1,01-Nov-02 09:42:13Z
surface times:
(2002,10,1,7.203529),17316,01-Nov-02 07:12:13Z
(2002,10,1,7.453529),19348,01-Nov-02 07:27:13Z
(2002,10,1,7.703529),20716,01-Nov-02 07:42:13Z
(2002,10,1,7.953529),21528,01-Nov-02 07:57:13Z
(2002,10,1,8.203529),21648,01-Nov-02 08:12:13Z
(2002,10,1,8.453529),21812,01-Nov-02 08:27:13Z
(2002,10,1,8.703529),21840,01-Nov-02 08:42:13Z
(2002,10,1,8.953529),21872,01-Nov-02 08:57:13Z
(2002,10,1,9.203529),21916,01-Nov-02 09:12:13Z
(2002,10,1,9.453529),21916,01-Nov-02 09:27:13Z
(2002,10,1,9.703529),21916,01-Nov-02 09:42:13Z
(2002,10,1,9.786862),21916,01-Nov-02 09:47:13Z
met times:
rad times:0
category-class availability:
  category Surface
    Surface Dosage=true,true
    Surface Deposition=true,true
    Concentration=false,false
    Dosage Duration=true,false
    CODA Human Performance=true,false
    Probability of Injury=true,false
    Probability of Mortality=true,false
    Casualty Table=false,false
  category Surface Slice
    Surface Dosage=false,false
    Surface Deposition=false,false
    Concentration=false,false
    Dosage Duration=false,false
    CODA Human Performance=false,false
    Probability of Injury=false,false
    Probability of Mortality=false,false
    Casualty Table=false,false
  category Horizontal Slice
    Surface Dosage=false,false
    Surface Deposition=false,false
    Concentration=true,true
    Dosage Duration=false,false
    CODA Human Performance=false,false
    Probability of Injury=false,false
    Probability of Mortality=false,false
    Casualty Table=false,false
  category Vertically Integrated Slice
    Surface Dosage=false,false
    Surface Deposition=false,false
    Concentration=true,true
    Dosage Duration=false,false
    CODA Human Performance=false,false

```

```

Probability of Injury=false,false
Probability of Mortality=false,false
Casualty Table=false,false
category Vertical Slice
  Surface Dosage=false,false
  Surface Deposition=false,false
  Concentration=true,true
  Dosage Duration=false,false
  CODA Human Performance=false,false
  Probability of Injury=false,false
  Probability of Mortality=false,false
  Casualty Table=false,false
category Table
  Surface Dosage=false,false
  Surface Deposition=false,false
  Concentration=false,false
  Dosage Duration=false,false
  CODA Human Performance=false,false
  Probability of Injury=false,false
  Probability of Mortality=false,false
  Casualty Table=true,false
class-choice availability:
  class Surface Dosage
    GB=true,true,1,3,2,false
  class Surface Deposition
    GB=true,true,1,3,2,false
  class Concentration
    GB=true,true,1,3,1,false
  class Dosage Duration
    GB=true,true,1,3,2,false
  class CODA Human Performance
    GB=true,true,4,2,2,true
  class Probability of Injury
    GB=true,true,6,2,2,false
  class Probability of Mortality
    GB=true,true,8,2,2,false
  class Casualty Table
    GB=true,true,10,1,2,false

[POLYGONS]
line 0=index:1,start:1,count:699,sense:left
-84.80067,33.398552
-84.80066,33.39855
-84.80059,33.398537
-84.80057,33.398533
-84.80051,33.398533
-84.80048,33.398533
-84.8004,33.39855
-84.80039,33.39855
-84.80038,33.398556
-84.80031,33.398598
-84.80023,33.398643
-84.80022,33.39866

```

-84.800156,33.39873
-84.80013,33.39879
-84.8001,33.39882
-84.800095,33.398857
-84.800064,33.398907
-84.80005,33.398983
-84.80005,33.398994
-84.800064,33.399055
-84.80007,33.399082
-84.80008,33.399136
-84.800095,33.39917
-84.8001,33.399197
-84.80013,33.399258
-84.80013,33.399258
-84.80013,33.399258
-84.80015,33.39933
-84.80015,33.399345
-84.80016,33.399403
-84.80017,33.399433
-84.80018,33.399475
-84.80019,33.39952
-84.80019,33.399544
-84.80021,33.39961
-84.80021,33.399616
-84.80022,33.399643
-84.80023,33.399685
-84.80023,33.399696
-84.80023,33.39977
-84.80024,33.399784
-84.80022,33.399868
-84.80022,33.39987
-84.80013,33.39991
-84.8001,33.399902
-84.80004,33.39991
-84.80001,33.399906
-84.79996,33.399887
-84.79995,33.399883
-84.799934,33.39987
-84.7999,33.399845
-84.799866,33.399796
-84.79986,33.399788
-84.79986,33.399784
-84.79985,33.399715
-84.79985,33.399696
-84.79986,33.399616
-84.79986,33.39961
-84.799866,33.399582
-84.79988,33.39952
-84.79989,33.399498
-84.79991,33.399433
-84.799934,33.39937
-84.79994,33.399345
-84.79996,33.399303

-84.79997,33.399258
-84.79999,33.399227
-84.8,33.39917
-84.80001,33.399113
-84.80002,33.399082
-84.80003,33.399002
-84.80003,33.398994
-84.800026,33.398926
-84.800026,33.398907
-84.80002,33.398846
-84.80001,33.39882
-84.799995,33.39878
-84.79999,33.39873
-84.79998,33.398712
-84.79996,33.398643
-84.79996,33.398643
-84.79996,33.398643
-84.79993,33.398586
-84.79991,33.398556
-84.79989,33.398533
-84.799866,33.39851
-84.79984,33.398495
-84.7998,33.398468
-84.79979,33.398464
-84.79978,33.39846
-84.79969,33.398464
-84.79969,33.398464
-84.79968,33.398468
-84.79961,33.398525
-84.79955,33.398556
-84.799515,33.398582
-84.79946,33.39861
-84.79943,33.398613
-84.79939,33.39859
-84.79936,33.398556
-84.79935,33.39855
-84.79934,33.39854
-84.79931,33.3985
-84.79929,33.398468
-84.79927,33.39845
-84.799255,33.39843
-84.79922,33.39842
-84.799164,33.3984
-84.79914,33.398407
-84.79908,33.398422
-84.79901,33.398468
-84.79899,33.398495
-84.79896,33.398556
-84.798904,33.398643
-84.798904,33.398643
-84.79886,33.39869
-84.79881,33.39873
-84.79881,33.39873

-84.798805,33.39874
-84.79873,33.39881
-84.79872,33.39882
-84.79871,33.398834
-84.798676,33.398907
-84.79865,33.39898
-84.798645,33.398994
-84.79864,33.39908
-84.79864,33.399082
-84.79865,33.39916
-84.79865,33.39917
-84.79867,33.39923
-84.798676,33.399258
-84.79868,33.3993
-84.798706,33.399345
-84.79871,33.39936
-84.79873,33.39939
-84.798744,33.399418
-84.79875,33.399433
-84.798775,33.399475
-84.7988,33.39952
-84.7988,33.399532
-84.79881,33.39956
-84.79884,33.39958
-84.79888,33.39961
-84.7989,33.39962
-84.798904,33.399628
-84.79896,33.39961
-84.79899,33.399597
-84.79907,33.39952
-84.79908,33.399517
-84.799126,33.399433
-84.799164,33.399364
-84.79919,33.399345
-84.799255,33.399284
-84.7993,33.399296
-84.79934,33.39931
-84.79935,33.399338
-84.799355,33.399345
-84.79938,33.399395
-84.79939,33.399433
-84.7994,33.39946
-84.79942,33.39952
-84.79942,33.399525
-84.79942,33.39961
-84.7994,33.399643
-84.799355,33.399696
-84.79934,33.399708
-84.79927,33.399784
-84.799255,33.399815
-84.79922,33.39987
-84.79922,33.39991
-84.799194,33.39996

-84.79918,33.400036
-84.79918,33.40005
-84.79917,33.40013
-84.79917,33.40014
-84.799164,33.40016
-84.79915,33.400227
-84.79915,33.40024
-84.79913,33.400314
-84.79913,33.400345
-84.79912,33.400402
-84.79911,33.40046
-84.7991,33.40049
-84.799095,33.40056
-84.799095,33.400578
-84.79909,33.400658
-84.79909,33.400665
-84.79908,33.400715
-84.799065,33.400753
-84.799034,33.400795
-84.799,33.40084
-84.79899,33.400856
-84.79897,33.400852
-84.79897,33.40084
-84.79895,33.40079
-84.79894,33.400753
-84.798935,33.400723
-84.79893,33.400665
-84.79892,33.400646
-84.79891,33.400578
-84.79891,33.400574
-84.798904,33.40056
-84.7989,33.4005
-84.79889,33.40049
-84.79888,33.40042
-84.798874,33.400402
-84.79887,33.400352
-84.79885,33.400314
-84.79884,33.400288
-84.79881,33.400265
-84.7988,33.400246
-84.798775,33.400227
-84.79876,33.400196
-84.79873,33.400192
-84.79869,33.40018
-84.79864,33.40016
-84.79862,33.400154
-84.79855,33.40014
-84.79855,33.40014
-84.79855,33.40014
-84.7985,33.4001
-84.79846,33.400066
-84.798454,33.40006
-84.79845,33.40005

-84.79842,33.4
-84.79841,33.39996
-84.7984,33.399937
-84.79839,33.39987
-84.79839,33.39986
-84.798386,33.399784
-84.798386,33.399773
-84.79839,33.399696
-84.79839,33.399677
-84.79839,33.39961
-84.79839,33.399593
-84.79839,33.39952
-84.79839,33.399506
-84.79838,33.39948
-84.79836,33.39945
-84.798355,33.399433
-84.798325,33.399395
-84.79829,33.399364
-84.79828,33.399357
-84.798256,33.399345
-84.79822,33.39933
-84.7982,33.39933
-84.79812,33.399338
-84.79811,33.399338
-84.7981,33.399345
-84.79802,33.39943
-84.79802,33.399433
-84.79802,33.399437
-84.798,33.39952
-84.79799,33.399555
-84.79798,33.39961
-84.79797,33.399654
-84.79797,33.399696
-84.797935,33.39977
-84.79793,33.399784
-84.797844,33.39986
-84.79783,33.39987
-84.79776,33.399925
-84.79773,33.39996
-84.79769,33.40003
-84.797676,33.40005
-84.79767,33.400078
-84.79765,33.40014
-84.79765,33.400158
-84.79764,33.400227
-84.79764,33.400257
-84.79764,33.400314
-84.79764,33.400345
-84.797646,33.400402
-84.797646,33.400425
-84.79765,33.40049
-84.79766,33.4005
-84.79767,33.40056

-84.79767,33.400574
-84.797676,33.400578
-84.79769,33.400642
-84.7977,33.400665
-84.79771,33.40071
-84.79772,33.400753
-84.79773,33.400784
-84.797745,33.40084
-84.797745,33.400856
-84.79776,33.40093
-84.79776,33.400932
-84.79776,33.400955
-84.79777,33.40101
-84.79777,33.401016
-84.79778,33.40108
-84.79778,33.401104
-84.79779,33.401157
-84.7978,33.40119
-84.797806,33.401234
-84.797806,33.40128
-84.79781,33.401314
-84.79782,33.401367
-84.79782,33.40139
-84.79783,33.401455
-84.79783,33.401466
-84.797844,33.401543
-84.797844,33.401543
-84.797844,33.40155
-84.79786,33.40162
-84.79786,33.40163
-84.79787,33.40169
-84.797874,33.40172
-84.79788,33.401768
-84.79789,33.401806
-84.7979,33.401844
-84.797905,33.401894
-84.79791,33.40192
-84.79792,33.40198
-84.79792,33.401993
-84.797935,33.40207
-84.797935,33.40207
-84.797935,33.40208
-84.79794,33.402145
-84.79795,33.402157
-84.79796,33.402218
-84.79796,33.402245
-84.797966,33.402298
-84.79797,33.402332
-84.79798,33.402374
-84.798,33.40242
-84.798,33.402443
-84.79801,33.402508
-84.79801,33.402515

-84.79801,33.402596
-84.79801,33.402603
-84.79802,33.402668
-84.79802,33.40268
-84.79802,33.402683
-84.79805,33.40274
-84.798065,33.40277
-84.79807,33.402805
-84.79809,33.40286
-84.79809,33.40288
-84.79808,33.402946
-84.79808,33.402973
-84.798096,33.403034
-84.7981,33.40304
-84.79811,33.403053
-84.79815,33.403084
-84.798164,33.403122
-84.79817,33.403152
-84.79819,33.40321
-84.79818,33.40323
-84.7982,33.403255
-84.79822,33.403282
-84.798225,33.403297
-84.79825,33.403336
-84.79828,33.403385
-84.79828,33.40339
-84.79829,33.4034
-84.79831,33.40345
-84.79838,33.403442
-84.798416,33.403435
-84.79846,33.403427
-84.79851,33.403385
-84.79853,33.403316
-84.79854,33.403297
-84.79855,33.403263
-84.79857,33.40321
-84.79858,33.403183
-84.7986,33.403122
-84.79861,33.40306
-84.798615,33.403034
-84.79864,33.40296
-84.798645,33.402946
-84.798645,33.40294
-84.798676,33.40286
-84.79869,33.402805
-84.798706,33.40277
-84.798706,33.40271
-84.798706,33.402683
-84.79872,33.402603
-84.79872,33.402596
-84.79873,33.402584
-84.79877,33.402508
-84.79877,33.402466

-84.798775,33.40242
-84.798775,33.402374
-84.798775,33.402332
-84.79878,33.402275
-84.79879,33.402245
-84.79881,33.402157
-84.79881,33.402157
-84.79881,33.40215
-84.79884,33.40207
-84.79886,33.402023
-84.79886,33.40198
-84.798904,33.4019
-84.79893,33.401962
-84.79894,33.40198
-84.79896,33.40201
-84.79899,33.402058
-84.79899,33.402065
-84.79899,33.40207
-84.79901,33.402138
-84.79902,33.402157
-84.79903,33.402206
-84.79906,33.402245
-84.799065,33.40226
-84.79908,33.402306
-84.79908,33.40233
-84.79909,33.402332
-84.79908,33.402405
-84.79908,33.40242
-84.79908,33.40242
-84.79908,33.402424
-84.799095,33.402496
-84.799095,33.402508
-84.79912,33.402554
-84.79914,33.402596
-84.79914,33.402615
-84.799164,33.402683
-84.799164,33.402687
-84.79914,33.40277
-84.79915,33.40279
-84.79916,33.40286
-84.799164,33.402863
-84.799164,33.402866
-84.799194,33.402912
-84.7992,33.402946
-84.79921,33.40299
-84.79922,33.403034
-84.79921,33.403076
-84.799225,33.403122
-84.799225,33.40315
-84.79924,33.40321
-84.79925,33.403217
-84.799255,33.403267
-84.79926,33.403294

-84.79926,33.403297
-84.79927,33.403366
-84.79928,33.403385
-84.799286,33.40344
-84.79929,33.403473
-84.7993,33.403515
-84.79931,33.40356
-84.79932,33.403587
-84.799324,33.40365
-84.79933,33.40366
-84.79934,33.403713
-84.79935,33.403732
-84.79935,33.403736
-84.79936,33.403805
-84.79938,33.403824
-84.799385,33.403866
-84.79941,33.403915
-84.799416,33.403934
-84.799416,33.404003
-84.799416,33.40402
-84.79943,33.404083
-84.79943,33.404087
-84.79943,33.40409
-84.79947,33.40414
-84.7995,33.40418
-84.7995,33.40419
-84.799515,33.404224
-84.79953,33.404255
-84.79953,33.404266
-84.79953,33.404343
-84.79954,33.404354
-84.79956,33.404396
-84.79959,33.40444
-84.7996,33.404446
-84.79961,33.40446
-84.799644,33.40452
-84.79967,33.404556
-84.79973,33.404655
-84.79973,33.40466
-84.799736,33.404663
-84.799736,33.404667
-84.799835,33.404736
-84.79991,33.4048
-84.79994,33.4048
-84.80009,33.404797
-84.80019,33.40466
-84.80026,33.40458
-84.8003,33.404484
-84.80033,33.40442
-84.80038,33.40431
-84.80044,33.404156
-84.800446,33.404133
-84.800446,33.404125

-84.80049,33.403957
-84.800514,33.40388
-84.800545,33.40378
-84.800575,33.403645
-84.80059,33.403606
-84.80061,33.4035
-84.80063,33.403435
-84.80063,33.403427
-84.80063,33.403416
-84.800644,33.40335
-84.80065,33.403297
-84.80066,33.40325
-84.80067,33.40321
-84.80067,33.403202
-84.80068,33.403122
-84.80068,33.4031
-84.80069,33.403034
-84.80069,33.403
-84.8007,33.402946
-84.800705,33.402897
-84.80071,33.40286
-84.80072,33.402798
-84.80072,33.40277
-84.80073,33.4027
-84.80073,33.402683
-84.80074,33.4026
-84.80074,33.402596
-84.80074,33.402576
-84.80076,33.402508
-84.80076,33.402493
-84.80078,33.40242
-84.80078,33.402386
-84.80079,33.402332
-84.8008,33.40228
-84.800804,33.402245
-84.800835,33.40216
-84.80084,33.402157
-84.80092,33.402092
-84.80094,33.402134
-84.80097,33.402157
-84.800995,33.402176
-84.80101,33.40219
-84.801025,33.402233
-84.80103,33.402245
-84.80105,33.402294
-84.80106,33.402332
-84.80107,33.402355
-84.801094,33.402393
-84.80111,33.40241
-84.801125,33.40242
-84.801155,33.402454
-84.801186,33.402473
-84.801216,33.402477

-84.80127,33.402477
-84.801315,33.40242
-84.80136,33.40237
-84.80138,33.402332
-84.801384,33.40231
-84.80141,33.402245
-84.801445,33.40217
-84.80145,33.402157
-84.80145,33.40215
-84.80148,33.40207
-84.80152,33.401997
-84.80153,33.40198
-84.80154,33.40196
-84.80155,33.401894
-84.80155,33.40188
-84.80156,33.401806
-84.80157,33.401775
-84.801575,33.40172
-84.801575,33.40168
-84.801575,33.40163
-84.801575,33.40159
-84.801575,33.401543
-84.80157,33.401512
-84.80155,33.401455
-84.80155,33.40144
-84.80154,33.40141
-84.80152,33.401382
-84.801506,33.401367
-84.80148,33.40133
-84.801445,33.401302
-84.80143,33.401295
-84.801414,33.40128
-84.801384,33.401253
-84.80136,33.401245
-84.80132,33.401234
-84.80127,33.401215
-84.80125,33.401215
-84.80121,33.40119
-84.8012,33.40118
-84.801186,33.401165
-84.80113,33.401154
-84.80112,33.401104
-84.80112,33.40108
-84.801125,33.401016
-84.80113,33.40098
-84.801155,33.40093
-84.801186,33.40087
-84.80119,33.40084
-84.8012,33.40082
-84.80123,33.400753
-84.80126,33.400677
-84.80126,33.400665
-84.80127,33.40065

-84.80131,33.400578
-84.80135,33.4005
-84.80135,33.40049
-84.80136,33.40048
-84.80141,33.400402
-84.801445,33.400333
-84.80146,33.400314
-84.80153,33.400234
-84.80154,33.400227
-84.80154,33.400227
-84.80154,33.400223
-84.80162,33.400192
-84.80168,33.40014
-84.80171,33.40012
-84.80175,33.40005
-84.801796,33.399986
-84.8018,33.39996
-84.80181,33.399944
-84.80185,33.39987
-84.80189,33.399788
-84.80189,33.399784
-84.80189,33.399784
-84.8019,33.399696
-84.80191,33.399673
-84.801926,33.39961
-84.80193,33.399567
-84.80194,33.39952
-84.80195,33.399464
-84.80195,33.399433
-84.80195,33.399372
-84.801956,33.399345
-84.80194,33.399292
-84.80194,33.399258
-84.80193,33.399216
-84.80192,33.39917
-84.80191,33.39915
-84.80189,33.3991
-84.80188,33.39909
-84.80187,33.399082
-84.801834,33.399048
-84.801796,33.399002
-84.80179,33.399
-84.80179,33.398994
-84.80174,33.398964
-84.80171,33.398937
-84.80169,33.39893
-84.80165,33.398907
-84.80163,33.398895
-84.80162,33.39889
-84.80156,33.39888
-84.80154,33.398876
-84.80146,33.39889
-84.801445,33.398895

-84.80143,33.398907
-84.80138,33.398975
-84.80136,33.398994
-84.80136,33.399006
-84.80134,33.399082
-84.80133,33.39911
-84.801315,33.39917
-84.80129,33.39923
-84.80129,33.399258
-84.80127,33.399323
-84.80126,33.399345
-84.801186,33.399433
-84.801186,33.399433
-84.801094,33.39948
-84.80104,33.39952
-84.80101,33.399555
-84.800934,33.39961
-84.80092,33.399616
-84.800896,33.399628
-84.80087,33.39961
-84.80084,33.399597
-84.800835,33.39959
-84.800804,33.399555
-84.80081,33.39952
-84.80082,33.399445
-84.80083,33.399433
-84.80082,33.399357
-84.80083,33.399345
-84.800835,33.399315
-84.80084,33.399258
-84.80084,33.39925
-84.80085,33.39917
-84.80085,33.399155
-84.80086,33.399082
-84.80086,33.39906
-84.80086,33.398994
-84.80086,33.39897
-84.80086,33.398907
-84.80086,33.398888
-84.80085,33.39882
-84.80084,33.398808
-84.800835,33.398754
-84.80083,33.39874
-84.80083,33.39873
-84.8008,33.39868
-84.80077,33.398643
-84.80076,33.398624
-84.80074,33.398605
-84.80071,33.398582
-84.800674,33.398556
-84.80067,33.398552

Appendix C

Sample IHPACServer Client Program

This is an example client to IHPACServer's CORBA services. It was tested under the same conditions as the sample detailed services client.

C.1 IHPACClient.java

```
// $Id$  
//-----  
//      NAME:          IHPACClient.java  
//      PURPOSE:       -  
//                  Example IHPACServer client  
//-----  
package samples;  
  
import javax.naming.Context;  
import javax.naming.InitialContext;  
import javax.naming.NameClassPair;  
import javax.naming.NamingEnumeration;  
  
import org.omg.CORBA.Any;  
import org.omg.CORBA.AnyHolder;  
import org.omg.CORBA.ORB;  
  
import mil.dtra.hpac.ihpacserver.dispersion.IDispersionServer;  
import mil.dtra.hpac.ihpacserver.dispersion.MessageListHolder;  
import mil.dtra.hpac.ihpacserver.dispersion.MessageS;  
  
import mil.dtra.hpac.ihpacserver.incident.IIncident;  
import mil.dtra.hpac.ihpacserver.incident.IIncidentMgr;  
  
import mil.dtra.hpac.ihpacserver.models.chembioweapon.CHEMBIOWEAPON_MODEL;  
  
import mil.dtra.hpac.ihpacserver.parameters.IWeather;  
  
import mil.dtra.hpac.ihpacserver.plot.IPlot;  
  
import mil.dtra.hpac.ihpacserver.project.IProject;
```

```

import mil.dtra.hpac.ihpacserver.release.IReleaseMgr;

import mil.dtra.hpac.ihpacserver.server.IHPACSERVER_FACTORY_SERVICE_NAME;
import mil.dtra.hpac.ihpacserver.server.IHPACServer;
import mil.dtra.hpac.ihpacserver.server.IHPACServerFactory;
import mil.dtra.hpac.ihpacserver.server.IHPACServerFactoryHelper;
import mil.dtra.hpac.ihpacserver.server.IHPACServerHelper;

import mil.dtra.hpac.server.IncidentModelServer;
import mil.dtra.hpac.server.IncidentModelServerFactory;
import mil.dtra.hpac.server.IncidentModelServerFactoryHelper;
import mil.dtra.hpac.server.IncidentModelServerHelper;
import mil.dtra.hpac.server.IncidentT;
import mil.dtra.hpac.server.IncidentTHolder;

import mil.dtra.hpac.server.project.TimeT;

import mil.dtra.hpac.server.release.HR_CONTINUOUS;
import mil.dtra.hpac.server.release.HR_FILE;
import mil.dtra.hpac.server.release.HR_INSTANTANEOUS;
import mil.dtra.hpac.server.release.HR_MOVING;
import mil.dtra.hpac.server.release.HR_POOL;
import mil.dtra.hpac.server.release.HR_STACK;
import mil.dtra.hpac.server.release.ReleaseDataT;
import mil.dtra.hpac.server.release.ReleaseT;

//-----
//      CLASS:          IHPACClient
//***** /*****
public class
IHPACClient
{
    // Class Methods
    //

//-----
//      METHOD:          main()
//***** /*****
public static void
main( String[] argv )
{
    boolean  standalone = false;
    String
        incident_id = "xxxxx",
        user_name = System.getProperty( "user.name" ),
        project_name = "ihpac-example";

    IHPACServerFactory  ihpac_factory = null;
    IHPACServer  ihpac_server = null;
}

```

```

IDispersionServer dispersion = null;
IProject project = null;
IIIncidentMgr incident_mgr = null;
IIIncident incident = null;
IReleaseMgr release_mgr = null;
IPlot plot = null;
IWeather weather = null;

try
{
    // Get Naming Context, List Names
    //
    Context context = new InitialContext();

    // Get Server Instances
    //
    ihpac_factory = IHPACServerFactoryHelper.narrow(
        (org.omg.CORBA.Object)
        context.lookup( IHPACSERVER_FACTORY_SERVICE_NAME.value )
    );
    ihpac_server = IHPACServerHelper.narrow( ihpac_factory.getInstance() );

    // Create Incident
    //
    System.err.println( "[IHPACServer] creating project" );
    project = ihpac_server.createProject(
        new StringBuffer( "c:\\\\tmp\\\\" ).
        append( project_name ).
        append( ".hpac" ).
        toString()
    );

    System.err.println( "[IHPACServer] building incident" );
    incident_mgr = project.getIncidentMgr();
    incident = incident_mgr.createIncident( CHEMBIOWEAPON_MODEL.value );

    incident.setName( "cbwpn-ex" );
    incident.update();

    System.err.println(
        new StringBuffer( 128 ).
        append( "[IHPACServer] " ).
        append( incident.getReleaseMgr().getReleaseCount() ).
        append( " releases" ).
        toString()
    );

    // Must Have Releases
    //
    if ( incident.getReleaseMgr().getReleaseCount() > 0 )
    {
        // Set Weather

```

```

        //
weather = project.getWeather();
weather.setFixedWindsSpeed( 5.0 );
weather.setFixedWindsDirection( 180.0 );

        // Save Project
        //
System.err.println( "[IHPACServer] saving project" );
project.save();

        // Calculate Dispersion
        //
MessageListHolder message_list_holder = new MessageListHolder();

System.err.println( "[IHPACServer] calculating..." );
dispersion = project.getDispersionServer();
if ( dispersion.startSynchCalculation( message_list_holder ) )
{
    System.err.println(
        "[IHPACServer] calculation succeeded, generating plot"
    );
        // Generate a Plot
        //
plot = project.getPlotMgr().createPlot();

int
option = plot.valueOfOption( "Surface Deposition" ),
type = plot.valueOfType( "Mean Value (M)" ),
choice = plot.valueOfChoice( "vx" ),
category = plot.valueOfCategory( "Surface Data" ),
kind = plot.valueOfKind( "Total", "Surface Deposition", "vx" ),
time_index = plot.getAvailTimes( option, choice ).length - 1;

plot.setPlotByTime(
    option, choice, kind, category, type, time_index
);
plot.compute();
project.save();

String[] plots = project.getPlotMgr().getPlotList();

System.err.println( "[IHPACServer] plots" );
for ( int i = 0; i < plots.length; i++ )
    System.err.println( " " + plots[ i ] );
} // if calculation succeeded

else
{
    MessageS[] messages = message_list_holder.value;

System.err.println( "[IHPACServer] calculation failed" );

for ( int i = 0; i < messages.length; i++ )

```

```

        System.err.println( " " + messages[ i ].fMessage );
    }
} // if releases
} // try

catch ( Exception ex )
{
    System.err.println( "[IHPACClient] " + ex );
    ex.printStackTrace( System.err );
}

finally
{
    if ( ! standalone )
    {
        if ( dispersion != null ) dispersion._release();
        if ( ihpac_factory != null ) ihpac_factory._release();
        if ( incident != null ) incident._release();
        if ( incident_mgr != null ) incident_mgr._release();
        if ( plot != null ) plot._release();
        if ( project != null ) project._release();
        if ( release_mgr != null ) release_mgr._release();
        if ( weather != null ) weather._release();

        if ( ihpac_server != null )
        {
            //ihpac_server.shutdown();
            ihpac_server._release();
        }
    } // if not standalone
} // finally
} // main
} // IHPACClient

```

C.2 IHPACClient.run.bat

```

@echo off
rem -----
rem - NAME:          IHPACClient.run.bat
rem - PURPOSE:
rem -           Windows batch script to run samples.IHPACClient
rem -----
set JRE_BASE=d:\j2sdk1.4.0
rem set JRE_BASE=c:\jdk1.3.1

set path=%JRE_BASE%\bin;%path%
set JRE_HOME=%JRE_BASE%\jre

set classpath=..
set ExtFlags=-Djava.ext.dirs=c:\hpac4\shared\ext;c:\hpac4\server\ext;%JRE_HOME%\lib\ext

set S01=-Djava.naming.factory.initial=com.sun.jndi.cosnaming.CNCtxFactory

```

```
set S02=-Djava.naming.provider.url=iiop://localhost:1400/
java -Xms16m -classpath .. %ExtFlags% %S01% %S02% samples.IHPACClient %*
```

C.3 IHPACClient.run.sh

```
#!/bin/sh -av
#-----
#-      NAME:          IHPACClient.run.sh
#-      PURPOSE:        -
#-                  Bourne shell script to run samples.IHPACClient
#-----

#      -- Check Command Line for a ServerURL
#
#      --
if [ $# -gt 0 ]; then
  ServerURL=$1
else
  ServerURL=iiop://localhost:1400/
fi

#      -- Point to J2SE
#
#      --
export JRE_HOME
JRE_HOME=/usr/local/j2sdk1.4.0/jre
# JRE_HOME=/usr/local/jdk1.3.1

export PATH
PATH=$JRE_HOME/bin:$PATH

export HPAC_DIR
HPAC_DIR=$HOME/src/hpacdev

java \
-Xms16m \
-classpath .. \
-Djava.ext.dirs=$HPAC_DIR/shared/ext:$HPAC_DIR/server/ext:$JRE_HOME/lib/ext \
-Djava.naming.factory.initial=com.sun.jndi.cosnaming.CNCtxFactory \
-Djava.naming.provider.url=$ServerURL \
samples.IHPACClient $*
```

C.4 run.out

This is standard output and standard error from the execution of IHPACClient.

```
[IHPACServer] creating project
[IHPACServer] building incident
[IHPACServer] 2 releases
[IHPACServer] saving project
[IHPACServer] calculating...
[IHPACServer] calculation succeeded, generating plot
[IHPACServer] plots
```

Sarin(Total);Surface Deposition;01-Oct-02 23:14:31Z