Advanced Use of the SEDRIS SDK

http://www.sedris.org/sdk

SEDRIS™ Technology Conference 2004
Lake Buena Vista, Florida
7 January 2004

Warren Macchi
Accent Geographic
wmacchi@accentgeographic.com

Kevin Wertman
SAIC
kevin.m.wertman@saic.com

Jesse Campos
SAIC
jesse.j.campos@saic.com

About this Tutorial

- **DESCRIPTION**
  - **This tutorial covers the more advanced aspects of the SEDRIS APIs, such as the C++ API, traversal techniques, sharing of objects, object IDs, the Syntax and Rules Checker applications, the TCRS, and Inter Transmittal Referencing. This workshop-styled tutorial provides attendees with an opportunity to exchange their views and learn directly from SEDRIS API developers. Participants are given exercises that address many aspects of the consumption and production of SEDRIS transmittals, and given an opportunity to try their solutions interactively.**
- **WHO SHOULD ATTEND**
  - **Experienced SEDRIS developers, developers dealing with large environmental data sets, and developers looking for an understanding of advanced traversal techniques for accessing SEDRIS transmittals.**
- **PREREQUISITES**
  - **Environmental modeling, and familiarity with the C++ language is helpful. Prior attendance at either the "Fundamentals of the DRM" or "Fundamentals for Accessing Transmittals" tutorial is recommended.**
- **WHAT TO EXPECT**
  - **At completion, the attendee will have an understanding of the various how-to techniques for use of SEDRIS resources and APIs in the creation and extraction of transmittals, and an understanding of the role of a TCRS for both producers and consumers of environmental data.**

Workshop Outline

8:30 – Overview of Workshop

8:40 – Introduction to the SEDRIS Transmittal Access C++ API

9:00 – Break up into groups

9:15 – Exercise Set 1

9:45 – Discussion

10:00 – Coffee break

10:30 – Exercise Set 2

11:00 – Discussion

11:15 – Exercise Set 3

11:45 – Discussion

12:00 – End of Workshop

Overview of Workshop

What You Will Learn

- **During the course of the workshop, we will be discussing:**
  - **The use of SEDRIS documentation resources and tools**
  - **Traversal techniques**
  - **Searching**
  - **DRM traversal branching and decision making**
  - **The efficiency rule of "don't touch me more than once"**
  - **Instancing and sharing**
  - **Choosing organization hierarchies**
  - **Inter-Transmittal References**
  - **The use of TCRS tools**

Methodology

- **Work through a list of exercises, building up in complexity.**
- **We have several notebooks that can compile and run SEDRIS code. They contain:**
  - **SEDRIS SDK**
  - **SEDRIS SDK Windows Help file**
  - **A beta version of the Transmittal Access C++ API**
  - **Sample code and transmittals**
  - **Tools:**
    - **FOCUS**
    - **Side-by-Side**
    - **EDCS Query Tool**
- **We also have a few CD-ROMs with most of the above software, so that you can edit and compile in your own machine (only headers and documentation for the C++ API are provided, no libraries).**

Methodology (cont'd)

- **Using your own notebook:**
  - **Copy the CD-ROM contents to your own machine, and if needed add "write" permissions to sample directories and files.**
  - **Pass the CD-ROM around.**
  - **Verify compilation environment.**
  - **Test distribution of your source files using the USB memories or the diskettes provided.**
  - **If possible, be prepared to share your machine with a few other attendees.**
- **Be prepared to stop working on the exercises during the discussion sessions, so that you don't miss important tips and hints.**
- **SEDRIS API developers will be available for consultation and feedback.**

A Word On Developer Resources

- SEDRIS developers have many resources available to them:
  - SEDRIS SDK documentation (included in releases)
  - Windows Help-based SEDRIS SDK documentation (available through the SDK download pages)
  - Proceedings and multimedia tutorials (video + presentation slides) of past SEDRIS Technology Conferences
  - Sample code and transmittals
  - SEDRIS tools at http://tools.sedris.org
  - Email help through help@sedris.org
- You can also become a SEDRIS Associate and participate in SEDRIS technology development activities (see the SEDRIS web site for more details).
- This conference! Complete with a talented mix of experienced SEDRIS developers and data producers.

Introduction to the SEDRIS Transmittal Access C++ API

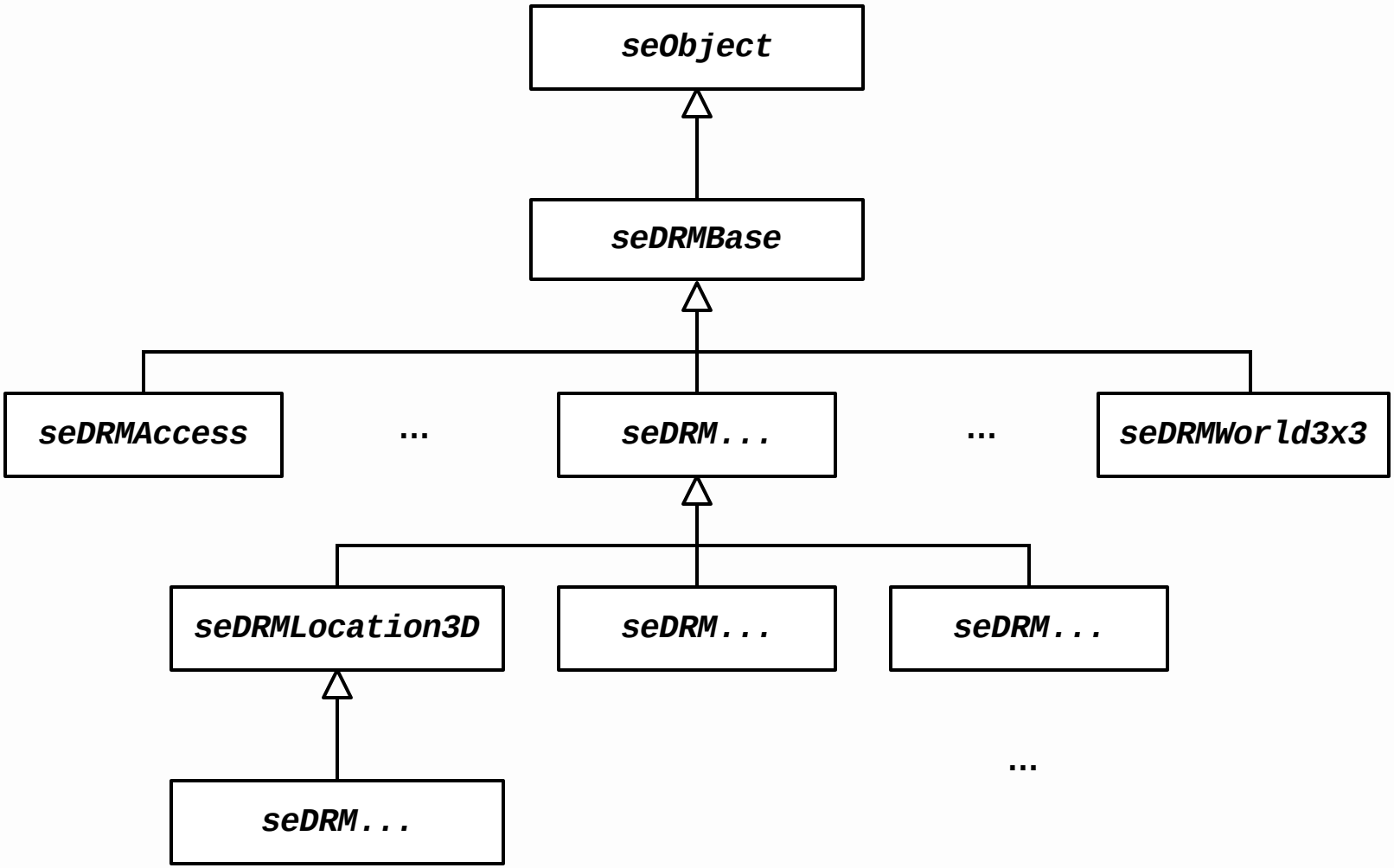
C++ API Basics

- **Similarly to the C API, the OOI C++ API is a set of header files and shared/static libraries for all supported platforms.**
- **All C++ classes are accessed through the "sedris" namespace to avoid name collisions with other libraries.**
- **All C++ API functions are documented in the header files, and also contain sample code that demonstrate their use.**

C++ API Classes

- ***seWorkspace***: Groups access to transmittals in a common setup.
- ***seTransmittal***: Access to transmittal data, creates DRM objects.
- ***seObject***: Manipulates DRM object instances. The DRM hierarchy of classes is derived from ***seDRMBase***, which is in turn derived from ***seObject***.
- ***seIterator***: Sequential access to DRM objects.
- ***seException***: Exception handling.

seObject Subclasses



Code Samples

Code Samples Topics

- **Opening a transmittal and displaying Transmittal Root's fields**
- **Iterating through a list of objects**
- **Creating a transmittal and objects**
- **Setting the fields of an object**
- **Using ITR and unresolved objects**

Sample – Open Transmittal

- **Open transmittal and display the Transmittal Root fields:**

```
seWorkspace wksp;

try
{
    seTransmittal xmtal;
    seDRMTransmittalRoot troot_obj;

    wksp.openTransmittalByFile( "test.stf", xmtal );
    xmtal.getRootObject();
    troot_obj.print();
}
catch ( seException &e )
{
    cerr << "Error - " << e.getWhat() << endl;
    return false;
}
```

Sample – Iterator

- **Display all model names:**

```
seIterator iter;
seDRMModelLibrary model_lib;
seDRMModel model;

if ( !troot_obj.getComponent( model_lib ) )
{
    cerr << "Model Library not found" << endl;
    return false;
}

model_lib.getComponentIterator( iter, SE_DRM_CLS_MODEL );

while ( iter.getNext( model ) )
{
    cout << "Model = "
        << model.get_name().characters
        << endl;
}
```

Sample – Creating Transmittal

- **Simple object creation:**

```
seDRMTransmittalRoot troot_obj;
seDRMAccess access;
seDRMCitation citation;
seDRMAbsoluteTimePoint abs_time_point;

wksp.createTransmittal( "create_cpp.stf", xmtal );

xmtal.createObject( troot_obj );
xmtal.setRootObject( troot_obj );
// set root_obj fields

xmtal.createObject( access );
// set access fields
troot_obj.addComponent( access );

xmtal.createObject( citation );
// set citation fields
troot_obj.addComponent( citation );

xmtal.createObject( abs_time_point );
// set abs_time_point fields
citation.addComponent( abs_time_point );
```

Sample – Setting Fields

- **Strings and basic-type parameters:**

```
seDRMTransmittalRoot troot;

troot.set_name("My transmittal name");
troot.set_credits("Credits go to the DRM");

…

seDRMUnionOfPrimitiveGeometry uofpg;
seDRMPolygon poly;

xmtal.createObject( uofpg );
uofpg.set_unique_descendants( SE_TRUE );

xmtal.createObject( poly );
poly.set_polygon_flags( SE_POLY_FLAG_TERRAIN
                      | SE_POLY_FLAG_SUN_ILLUMINATED );

uofpg.addComponent( poly );
```

Sample – ITR

- Unresolved objects in the C++ API are created by the `seWorkspace` class. You can treat unresolved objects like regular objects, so you can add them as components/associates of other objects using the same methods of regular objects. You can call the `seObject::resolve` method to make the API attempt to resolve an unresolved object.

- Sample ITR usage:

```
seDRMTransmittalRoot troot_obj;
seDRMModelLibrary model_lib;
seObject itr_model;

wksp.createTransmittal( "create_itr.stf", xmtal );
xmtal.createObject( troot_obj );
xmtal.setRootObject( troot_obj );
xmtal.createObject( model_lib );
troot_obj.addComponent( model_lib );

wksp.createUnresolvedObject( xmtal_urn, model_label, itr_model );
model_lib.addComponent( itr_model );
```

Break up into groups

Exercise 1 - Consumption

“How many <Polygon>s are under the <Environment Root>?”

Exercise 1 – Why?

- **We may need to understand the performance requirements for the visualization of the geometry in a SEDRIS transmittal.**

- **Some questions we may need to answer:**
  - **How many polygons are there in the transmittal?**
  - **How many polygons are trees/vegetation?**
  - **How many are buildings?**
  - **Is the data DRM compliant?**
  - **Does the data have LOD levels?**
  - **How large is the image data/number of images?**
  - **Can we make assumptions on the data organization?**

Exercise 1 - Task

- **Create an application that retrieves the <Environment Root> object in "belle31.stf", and returns the number of <Polygon> objects under its hierarchy.**

- **Hints:**
  - **Use source file "sample_iterator.cpp" to get you started.**
  - **To find all <Polygon> objects, consider using a recursive function, or see the *`seSearchIterator`* utility class.**
  - **"Fundamentals for Accessing Transmittals" tutorial.**
  - **Consider other object you may find in the hierarchy, for example <Model> objects.**
  - **If you want to keep track of objects you have seen, consider using Object IDs (may be <Model>s? why?).**

Exercise 1 - Discussion

- **What you learned:**
  - **Compiling/running SEDRIS applications**
  - **Opening transmittals**
  - **Handling exceptions**
  - **Use of SEDRIS documentation**
  - **Use of iterator-based classes**
  - **Model instancing**
  - **Using Object IDs to keep track of objects**
  - **Object reuse, advantages and disadvantages**
- **Other topics:**
  - **How would you determine the number of polygons used for vegetation/buildings/water bodies?**
  - **What was the hardest part of the exercise?**

Exercise 2 - Producing

“Is my transmittal valid?”

Exercise 2 – Why?

- **To produce transmittals that contain valid DRM object hierarchies and that satisfy customer requirements.**

- **Some considerations:**
  - **How do I verify that the transmittal I produced is "valid"?**
  - **What does it mean for a transmittal to be "valid"?**
  - **Is there a "right" way of producing transmittals?**
  - **How can my customers certify that a transmittal meets their requirements?**
  - **What tools are available to help me diagnose transmittal problems?**

Exercise 2 - Task

- **Identify the problems in the "test.stf" transmittal found in the "broken_test" directory, and fix it as needed.**

- **Hints:**
  - **Use the Syntax and Rules Checker applications.**
  - **Preview the data using Side-by-Side, is there anything missing?**
  - **Consider using the FOCUS tool for small, simple edits.**

- **Extra challenge:**
  - **Move the instanced model 5 units up.**

Exercise 2 - Discussion

- **What you learned:**
  - **Not all transmittals are created equal**
  - **All transmittals should be verified against the Syntax and Rules Checker applications**
  - **Object sharing implications**
  - **The TCRS application will allow you to check a transmittal against a set of rules that must be satisfied**
- **Other topics:**
  - **How do you describe the contents of your transmittal?**
  - **What would you do if you can't find a mapping from your internal format to a SEDRIS DRM hierarchy?**

Exercise 3 - Editing

“Why isn’t my transmittal displaying correctly?”

Exercise 3 – Why?

- **Sometimes the data sets we receive are not in the organization we want them, data is missing, or there is another problem. We want to learn to identify these and correct them if possible.**

- **Some considerations:**
  - **Is it possible that I receive an invalid transmittal?**
  - **Is there a "right" way of producing transmittals?**
  - **How do I know that my transmittal may have problems?**
  - **What do I do if I receive a transmittal that has a defect, or if I need the data in some alternative hierarchy?**
  - **How do I edit a transmittal to modify its hierarchy?**
  - **If my code cannot handle a particular DRM organization, what should I do?**

Exercise 3 - Task

- **Create an application that edits the "belle31.stf" transmittal in the "broken_belle" directory and fixes the water and terrain polygon display.**

- **Hints:**
  - **Use <Classification Data> to identify the proper geometry branches.**
  - **The Model Viewer application can be used to display the <Image> objects in a transmittal.**

- **Extra challenge:**
  - **Instead of using the <Image>s in the Bellevue transmittal, refer to the <Image>s in another transmittal using ITR.**

Exercise 3 - Discussion

- **What you learned:**
  - **How to edit transmittals**
  - **Finding a problem in your source data can be time consuming, but there are many tools to help you**
  - **Transmittals can be composed of other transmittals using ITR**
  - **Object reuse can reduce the size of transmittals, in some cases by large amounts**
- **Other topics:**
  - **Is it always possible/the right choice to modify your source data?**
  - **Is there a way of specifying/verifying whether my code will handle a specific transmittal?**

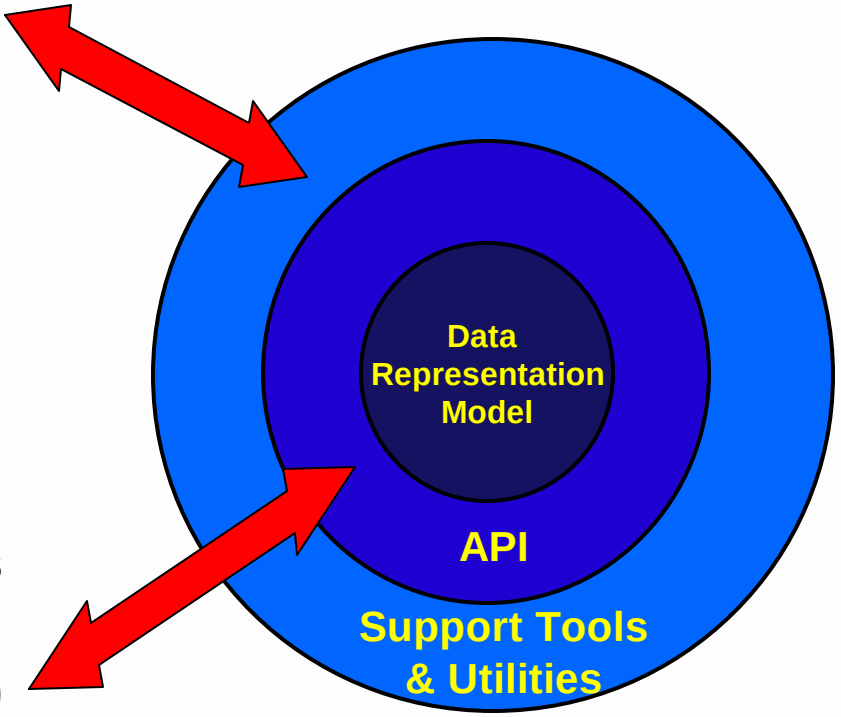
Where to Go From Here

- **To take a look at what others are doing with SEDRIS: "Panels and Presentations" - Thursday 8:30 AM through Friday. Both tracks have an excellent selection of activities and projects using SEDRIS technologies.**
- **On Thursday at 11:30 AM, Greg Hull will be giving a presentation on TCRS (see "Panels and Presentations, Track 1").**
- **"How to Produce and Consume Transmittals" – Friday 8:30 AM**
- **Read the Technical Guides in the SEDRIS SDK Documentation to learn more about Data Tables, Control Links, Images, and other SEDRIS technologies (docs/tech/index.htm).**
- **Use the SEDRIS help line for general SEDRIS questions help@sedris.org.**

Give me more!

Development of SEDRIS

- ***SEDRIS Associates*** (key environmental database developers/users)
  - **Review and feedback**
    - **DRM**
    - **EDCS**
    - **SRM**
    - **Transmittal Access**
  - **Native-model mapping**
  - **Interchange experiments**
  - **Value-added tools/utilities**
- ***Core Team***
  - **Manage evolution of components**
  - **Reference implementations**
  - **SEDRIS Transmittal Format (STF)**
  - **Common tools & applications**



Applying SEDRIS Technologies

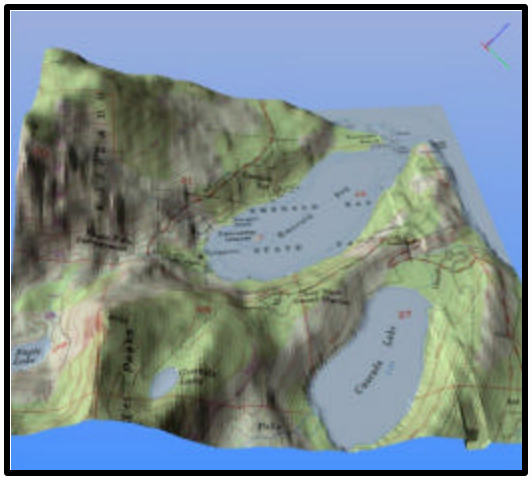
- **Use:**
  - **the DRM, EDCS, and the SRM to model environmental data**
  - **the DRM, EDCS, and the SRM to specify environmental database content**
  - **the EDCS as a stand-alone component**
  - **the SRM as a stand-alone component**
  - **all SEDRIS technology components as an interchange mechanism**
  - **SEDRIS tools to examine environmental data**
  - **SEDRIS Technologies as the basis to develop new tools**

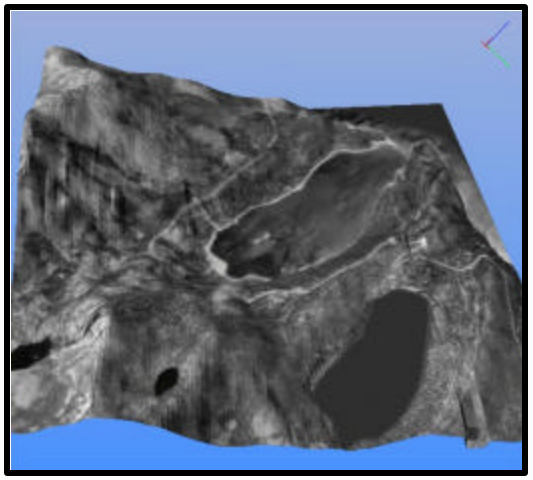
Developing SEDRIS Expertise

- Learn to speak SEDRIS: learn the DRM
- Generate mapping documents
  - For native format(s) or assigned government format(s)
  - To ensure the DRM and EDCS can handle all data requirements
- Develop software: to convert native data into SEDRIS and back to check completeness of the interchange
- Become a SEDRIS Associate and participate in SEDRIS Associates Meetings (SAMs) and interchange experiments
  - Exchange ideas
  - Cooperatively define and develop SEDRIS technology
  - Share non-proprietary (native format) utilities and applications that support SEDRIS interchange

Data Set Views

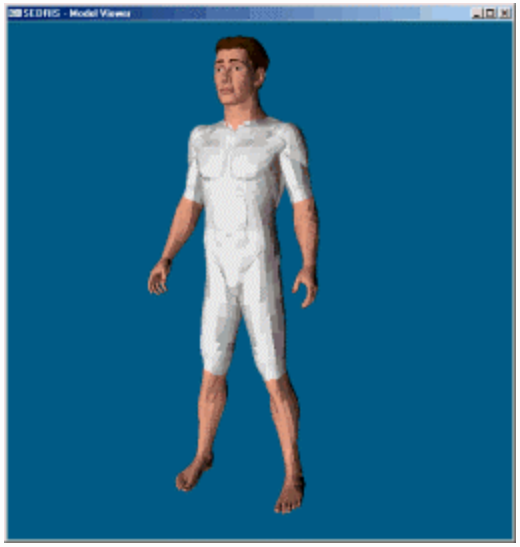
- **Gridded Data:**
  - **An example of gridded data is a DTED data set giving terrain elevation in a regular latitude / longitude (Geodetic) grid.**
  - **Displaying "height" Property Grids involves creating triangles for the data. We are considering the inclusion of a Level 1 API function that can turn this kind of tables into polygonal data.**





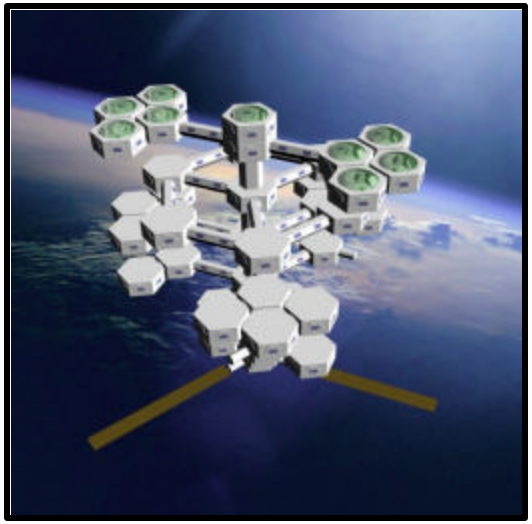
Other Data Sets (cont'd)

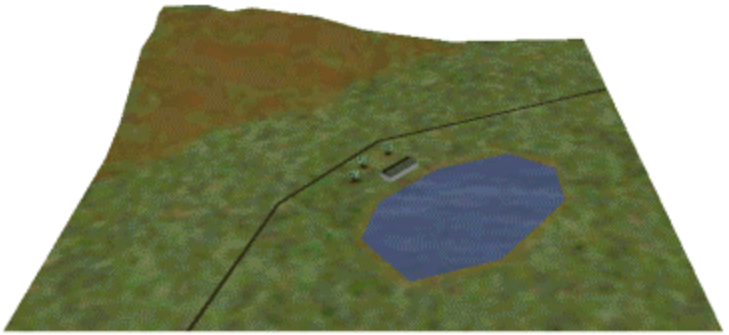
- **Human models:**
  - **The Humanoid Animation (H-Anim) Working Group (www.h-anim.org) was formed in 1997, with the goal of establishing a standard way of representing humanoids, such that they may be interchanged among modeling, authoring, and run-time applications.**
  - **We created a mapping and converted H-Anim 1.0 models in VRML97 into SEDRIS 3.1 STF files.**
  - **The mapping is meant to preserve the hierarchy and the rotational components of the humanoids.**



Other Data Sets (cont'd)

- **SEDRIS Space Station Gamma:**
  - **This model was created to provide a data set that can be used for demonstrating and testing the DRM classes.**
- **SEDRIS smallVille:**
  - **Demonstrates alternate gridded/polygonal data sets, geotypical images, instancing, stamp behavior.**





End Of Workshop

Appendix

C and C++ API Function List Comparison

C and C++ API – List

Transmittal Functions

C API Function	Comments	C++ API
CloseTransmittal	Automatic close on workspace handle going out of scope (or explicit seWorkspace::close).	seTransmittal::close
FreeTransmittal	Handles are freed automatically when they go out of scope, but method is still available for memory management purposes.	seTransmittal::release
GetObjectForID	Functionality is now assigned to a Transmittal. Hence, you need to open the transmittal before you can get a handle to the object. Previously you could get a handle to an object without any other notion of how you opened the transmittal.	seTransmittal::getObjectFromID
GetPublishedObjectList	Now takes an iterator rather than a store and allocating an array.	seTransmittal::getPublishedObjectsIterator
GetReferencedTransmittalList	Now you can get the count, and then ask for individual references rather than a store and allocating an array. Also allows for retrieval of labels used for each referenced transmittal.	seTransmittal:: getITRReferenceCount seTransmittal:: getITRReference seTransmittal:: getITRReferenceLabelCount seTransmittal:: getITRReferenceLabel
GetRootObject		seTransmittal::getRootObject
GetTransmittalFile		seTransmittal:: getFileName
GetTransmittalFromID	Functionality of object IDs is now limited to an open transmittal.	
GetTransmittalName		seTransmittal::getURN
GetTransmittalVersionInformation	Removed because we don't enforce the setting of a Transmittal Root object. Users can get the root object and check themselves.	
GetUniqueTransmittalID	Returns a string pointer.	seTransmittal::getUniqueID
ObjectIDsSupported	All implementations required to support object IDs.	
OpenTransmittalByFile	Implementation ID now handled at Workspace level.	seWorkspace::openTransmittalByFile
OpenTransmittalByName	Implementation ID now handled at Workspace level.	seWorkspace::openTransmittalByURN
ResolveTransmittalName		seWorkspace:: resolveTransmittalURN
RemoveFromTransmittal		seTransmittal::removeObject
SetRootObject	No previous root object is returned, since one can call seTransmittal::getRootObject.	seTransmittal::setRootObject
SetTransmittalName		seTransmittal::setURN
StringToObjectID	OBE.	
TransmittalsAreSame		seTransmittal::isSameAs

C and C++ API – List (cont'd)

Object Manipulation Functions

C API Function	Comments	C++ API
AddAssociateRelationship	"make_two_way" removed since it is not needed for normal operation.	seObject::addAssociate
AddComponentRelationship		seObject::addComponent
CloneObject	Objects are handles that can be cloned (for DRM class compatible classes). drmBase-derived classes can use copy constructor.	seObject::cloneTo
CompareObjectIDs	ID are returned as strings, and can be compared as such.	
CreateObject		seTransmittal::createObject
FreeObject	Handles freed automatically when they go out of scope.	seObject::release
GetAggregate		seObject::getAggregate
GetAssociate		seObject::getAssociate
GetComponent		seObject::getComponent
GetContextTransformation	Not implemented, delegated to higher-level functionality.	
GetFields		seObject::getFields and specific drmBase-derived classes
GetIDForObject		seObject::getID
GetImplementationIdentifier	Used to be for SE_Objects.	seTransmittal::getImplementationID
GetNthAssociateOfDRMClass	Use iterators.	
GetNthComponentOfDRMClass	Use iterators.	
GetNumberOfPathsToTransmittalRoot	Not implemented, delegated to higher-level functionality.	
GetObjectReferenceCount	Removed since user can keep top level object references.	
GetPublishedLabels	Used to use store and array of labels.	seObject:: getPublishedLabelCount seObject:: getPublishedLabel

C and C++ API – List (cont'd)

Object Manipulation Functions (cont'd)

C API Function	Comments	C++ API
GetRelationCounts	Use iterators (ITR issues involved).	
GetSortKey	Removed, can use object IDs.	
GetSRFParameters	Not implemented, delegated to higher-level functionality.	
GetTransmittalFromObject		seObject::getTransmittal
GetUnresolvedObjectFromPublishedLabel	Functionality moved to the Transmittal to be more consistent with related Object methods (addComponent, addAssociate, etc).	seWorkspace::createUnresolvedObject
GetUserData		seObject::getUserData
HasAggregates		seObject::hasAggregates
HasAssociations		seObject::hasAssociates
HasComponents		seObject::hasComponents
IdentifyObject		seObject::getDRMClass
ObjectIsPublished		seObject::isPublished
ObjectIsResolved		seObject::isResolved
ObjectsAreSame		seObject::isSameAs
PublishObject		seObject::publish
PutFields		seObject::setFields and specific drmBase-derived classes
RemoveAssociateRelationship		seObject::removeAssociate
RemoveComponentRelationship		seObject::removeComponent
ResolveObject		seObject::resolve
SetUserData		seObject::setUserData
UnpublishObject		seObject::unpublish

C and C++ API – List (cont'd)

Iterator Functions

C API Function	Comments	C++ API
CreateSearchFilter	Not implemented, delegated to higher-level functionality.	
CreateSpatialSearchBoundary	Not implemented, delegated to higher-level functionality.	
DetermineSpatialInclusion	Not implemented, delegated to higher-level functionality.	
FreeSpatialSearchBoundary	Not implemented, delegated to higher-level functionality.	
FreeIterator		seIterator::release
FreePackedHierarchy	Not used.	
FreeRemainingObjectsList	Not used.	
FreeRemainingPackedHierarchiesList	Not used.	
FreeSearchFilter	Not implemented, delegated to higher-level functionality.	
GetIterationLengthRemaining	Enhanced to return number of objects left of a specific type.	seIterator::getCount
GetNextObject		seIterator::getNext
GetPackedHierarchy	Not used.	
GetRemainingObjectsList	Not used.	
GetRemainingPackedHierarchiesList	Not used.	
InitializeAggregateIterator	No complex search filters.	seObject:: getAggregateIterator
InitializeAssociateIterator	No complex search filters.	seObject:: getAssociateIterator
InitializeComponentIterator	Not implemented, delegated to higher-level functionality:	seObject:: getComponentIterator
	Directly attach table components.	
	Process inheritance.	
	Transform locations.	
	Follow model instances.	
	Evaluate static control links.	
	Hierarchy select and order parameters.	
	Traversal pattern.	
	Complex search filters.	
InitializeInheritedComponentIterator	Not implemented, delegated to higher-level functionality.	

C and C++ API – List (cont'd)

Miscellaneous Functions

C API Function	Comments	C++ API
CreateStore	Not used.	
FreeStore	Not used.	
GetDataTable		seDRMDataTable::getDataTableData
GetColourModel	Not implemented, delegated to higher-level functionality.	
GetElementOfDataTable	To be added to a Data Table Helper class.	
GetErrorDescription		seException
GetImageData	No sub-extents (functionality was not implemented).	drmImage::getImageData
GetPackedDataTable	OBE. Data returned in specific type arrays, by extents.	
ObjectIDToString	Not used, IDs are now strings.	
PutDataTable		seDRMDataTable::setDataTableData
PutDataTableSubExtent	OBE.	
PutElementOfDataTable	To be added to a Data Table Helper class.	
PutElementOfDataTableSubExtent	OBE.	
PutImageData	No sub-extents (functionality was not implemented).	drmImage::setImageData
PutPackedDataTable	OBE.	
PutPackedDataTableSubExtent	OBE.	
SetCallbackForOneFunctionOneStatusCode	Using exception handling.	
SetFirstErrorMessage	Using exception handling.	
SetGeneralCallback	Using exception handling.	
SetGeneralCallbackForOneFunction	Using exception handling.	
SetSecondErrorMessage	Using exception handling.	
SetColourModel	Not implemented, delegated to higher-level functionality.	
SetSRFParameters	Not implemented, delegated to higher-level functionality.	
UseDefaultColourModel	Not implemented, delegated to higher-level functionality.	
UseDefaultSRFParameters	Not implemented, delegated to higher-level functionality.	
CreateUnresolvedObject		seWorkspace::createUnresolvedObject

C and C++ API – List (cont'd)

Level 1/Utility Functions

C API Function	Comments	C++ API
AddObjectToAggregate	Quick create/add/release. Possibly not needed.	
AllocDataTableData	Delayed.	
AllocElementOfDataTableData	Delayed.	
AllocPackedDataTableData	Delayed.	
FreeDataTableConstantSignature	Delayed.	
FreeDataTableData	Delayed.	
FreeDataTableDataStrings	Delayed.	
FreeDataTableSignature	Delayed.	
FreeDataTableSubExtent	Delayed.	
FreeElementOfDataTableData	Delayed.	
FreeElementOfDataTableDataStrings	Delayed.	
FreePackedDataTableData	Delayed.	
FreePackedDataTableDataStrings	Delayed.	
GetDataTableConstantSignature	Delayed.	
GetDataTableSignature	Delayed.	
GetDataTableSubExtent	Delayed.	
GetFeatureSameAsID	Could be added to a util package.	
GetGeometrySameAsID	Could be added to a util package.	
GetRearrangedImageData	To be added to a Image Helper class.	
GetSizeOfDataTableData	Delayed.	
GetSizeOfElementOfDataTable		
GetSizeOfImageData		seDRMImage::getImageDataSize
GetSizeOfPackedDataTableData	Delayed.	
ImageNameFromImageMappingFunction	Could be added to a util package.	
ModelNameFromFMI	Could be added to a util package.	
ModelNameFromGMI	Could be added to a util package.	
PrintDesiredImageParameters	Needs to be added to a util package.	
RearrangeImageData	To be added to a Image Helper class.	
RemoveObjectAndLinks	Needs to be added to a util package.	
RemoveObjectTree	Needs to be added to a util package.	
SetErrorHandlers	Custom seExceptions?	
SimpleCreateComponentIterator	Not needed.	
SoundNameFromSoundInstance	Could be added to a util package.	
ValidDesiredImageParameters	Needs to be added to a util package.	