Access Grid Virtual Venues Programmer's Manual

Release 2.0alpha1

Robert Olson

December 4, 2002

olson@mcs.anl.gov

Abstract

This document is a programmer's guide for modifying and extending the Access Grid software environment.

Contents

1 Introduction

2 Package breakdown

The VV software is broken into a number of different Python packages. These packages are rooted in a hierarchy based at package `AG`.

Module name	Purpose
`AG.services.xmlrpc`	XMLRPC-based web services library
`AG.services.venue`	The AG Venues service implementation
`AG.apps.venue`	The AG Venues client
`AG.apps.shared_web`	AG shared web browser
`AG.apps.docking_tools`	Standard collection of workspace docking tools
`AG.tools.agsd`	AG service description processing tool

3 Venue description language

In a number of locations in the virtual venues system it is useful to provide the user with a free-form abstract mechanism for describing objects (users, venues, services, etc). We define a description model which allows ample expressiveness in the representation of description data while using simple data structures. We desire the use of simple data structures to enable the use of high-level scripting languages. It also allows us to effectively leverage the use of the XMLRPC protocol for remote procedure invocation. XMLRPC supports the following data types (from *XML-RPC HOWTO*):

Data type	Description
int	A signed, 32-bit integer.
string	An ASCII string, which may contain NULL bytes.
boolean	Either true or false.
double	A double-precision floating point number. (Accuracy may be limited in some implementations.)
dateTime.iso8601	A date and time.
base64	Raw binary data of any length; encoded using Base64 on the wire.
array	An one-dimensional array of values. Individual values may be of any type.
struct	A collection of key-value pairs. The keys are strings; the values may be of any type.

Given this background, we can now define a description. It is simply an unordered set of tag/value pairs, where the tags are strings and the values are any of the basic types enumerated above. This maps naturally to the XMLRPC struct data type. In Python, a description maps to a dictionary.

Given this background, the real meaning and utility of a description lies in a common definition of tag names and their meaning. The virtual venues software uses the following standard descriptions.

3.1 User

A user description provides the information about a logged-in user. This description is present in the venue when the user is present. The following information is provided by the venue itself.

Tag name	Definition
secure_flag	True (1) if user has connected using an authenticated channel.
identity	User's authenticated identity.
entry_id	The unique identifier of this user's connection to the venue.
preferences	User's client preferences, as advertised by the user's client.
description	User's description, as advertised by the user's client.

The *description* contains the following information, as provided by the reference implementation of the venues client:

Tag name	Definition
name	User's name
email_address	User's email address
phone_number	User's phone number
institution	User's institution
url	User's personal url
toolbar_handle	Service handle for the client's toolbar service

3.2 Venue

A venue description is generated by the venue service and returned to its clients upon execution of the `enter` or `refresh_state` operations.

Tag name	Definition
user_list	Array of user descriptions, one for each logged-in user
media	Array of media descriptions
connections	Array of connection descriptions
service_registry	Handle of the service registry for this venue.
description	Description defining the persistent attributes of this venue
.	

The persistent description mentioned above contains the following tags.

Tag name	Definition
name	Venue name.
description	Textual description of the venue.

4 `AG.services.venue` — Access Grid Venue Server

This package contains the reference implementation of the Access Grid Virtual Venues server. It is composed of several modules:

`AG.services.venue.client` Tools for venues clients

4.1 `AG.services.venue.VenueService` — The Venue service interface module

4.2 `AG.services.venue.VenueServiceImpl` — Venue service implementation module

4.3 `AG.services.venue.VenueServer` — The Venue server itself.

4.4 AG.services.venue.VenueUser — User state for Venues users

4.5 AG.services.venue.client — Tools for clients

The client module provides a set of classes and methods that abstract away the details of the communication between the venue client and service. Via the mechanisms provided in the client module, the application writer can concentrate on the details of the application itself and manipulate the venue data at an appropriately high level.

A Venue provides several types of data to the client. Each of these data types is made available through the client interface encapsulated in a Python object. *However, we currently only wrap users in objects.*

class ClientVenueInterface()
The ClientVenueInterface class is the primary interface the client uses to talk to the venue.

class User(*user_desc, is_local*)
An instance of the User class contains the information maintained by a venu about that user.

ClientVenueInterface objects

The ClientVenueInterface class provides the following methods.

get_description()
Returns the dictionary used to construct the description of this client. See Section 3 for more information on how descriptions are defined.

get_preferences()
Returns the dictionary used to construct the preferences for this client. *Preferences are not currently implemented.*

connect(*service_handle*)
Connect to the Venue service at handle *service_handle* by invoking the enter service operation.

This method messages the venue service with the current description and preferences as configured in the client. If the client is allowed to enter the venue, the venue returns the description of the venue. The ClientVenueInterface digests this description and makes the data available to its users via the get_service_registry, get_connections, get_user_list, get_media_channels, and get_venue_description methods.

disconnect()
Disconnect the client from the venue by invoking the exit service operation on the venue.

reconnect()
If the client is already connected, disconnect. Then connect to the service previously connected to.

keepalive()
Invoke the keepalive operation on the venue to keep this client's soft state registration from timing out.

refresh()
Invoke the refresh_state operation on the venue service to obtain a new description for the venue. Digest the description in the same manner as enter.

get_service_registry()
Return the service registry handle for the venue, if present.

get_connections()
Return the list of connections present in the venue. The connection list is a python list of tuples (connection_name, connection_handle).

get_user_list()
Return the list of users in the venue. Each user is an instance of the User class.

get_media_channels()
Return the list of media channels in the venue. Each channel is (currently) a tuple `(media, address, port)`, where media is `video` for a video channel and `udaio` for an audio channel.

get_venue_description()
Returns the full venue description dictionary.

get_my_id()
Returns the unique identifier for this client.

`User` objects

The `User` object provides the following methods.

is_local_user()
Returns true if this user object corresponds to the currently-executing application.

get_DN()
If the user authenticated to the venue using a secure connection, return the distinguished name for that authentication. Other, return *None*.

get_id()
Return the user's unique identifier (as assigned by the venue).

get_name()
Return the user's name.

get_attributes()
Return the user description as a Python dictionary.

get_attribute(*key*)
Look up and return the user description entry for *key*.

Module Index

A

Index

A

C

D

G

I

K

R

U

X