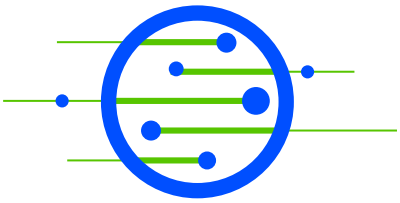

PlanetLab Core Specification 4.0

Aaron Klingaman, Mark Huang, Steve Muir, Larry Peterson

Princeton University

PDN–06–032
June 2006

Status: Ongoing Draft.

Table of Contents

Chapter 1. Core Specification

1. Introduction

This document describes the interfaces and protocols that collectively define the PlanetLab Core, also known as PlanetLab Central (PLC). Any entity that wishes to operate a *slice* or *management* authority, and federate with the publically deployed PlanetLab, must be compliant with this interface.

This document assumes that the reader is familiar with the overall PlanetLab architecture, as described in the companion PDN-06-031 [http://www.planet-lab.org/PDN/PDN-06-031/].

2. Version

This document defines Version 4.0 of the PlanetLab Core Specification. The current statis is: Draft.

3. Remote Procedure Call Mechanism

To ensure interoperable access to federated PlanetLab installations, XML-RPC will serve as the transport mechanism for all calls specified in this document. More information about XML-RPC can be found at [http://www.xmlrpc.com/].

Examples of actual calls and the XML data transmitted can be found in the concluding section of this document.

4. Extending Interfaces

The set of interfaces specified here may be extended as necessary to support site-specific functionality, so long as the extentions do not break compatibility with the specification. Common extensions could include such items as:

1. Additional authentication and authorization mechanisms, including adding role and/or capability based restrictions on certain method calls. More details on extending the authentication function can be found in the Authentication Interface chapter.

2. New abstractions, such as a group of principals organized into common geographic site, or organization.

3. Extending node related calls to include node-specific data, such as network addresses or node state.

The functions specified here have been designed so that adding additional values to the parameters can be done without breaking compatibility with this specification. This is most often done by adding additional key/value pairs to parameters that are of type struct, or associative array. Clients of systems that interface to these functions should be designed so that unknown key/value pairs are ignored.

The fault code 902, "Omitted a required extension.", can be returned to the user if the implementation of a function requires additional parameters that are not specified by this document, and were omitted by the user.

5. Return Values

For all functions, if the call is successful and the desired operation performed, the return value should be the type and value specified by the function. Calls that result in failure and that did not modify any internal state, due to authentication, authorization, or other failure, should instead return an XML-RPC fault.

For the common failures specified in each function, the assigned XML-RPC fault must be returned. Additional faults may be returned if implementation specific additions result in new failures.

6. Namespaces

Throughout this specification, three primary entities are referenced: slices, nodes, and principals. Each has its own method for identification, and an associated namespace. These three are outlined below.

6.1. Slices

A slice is a collection of virtual machines on a set of nodes, identified by a string name. For a full dicussion on slices and the slice namespace, see the PlanetLab Architecture Document, Section 4.4.

6.2. Nodes

A node is a machine capable of hosting one or more virtual machines. Machines located on networks are typically identified by their hostname, however, in the core PlanetLab specification, they are identified by a unique integer value referred to as node_id. Within a management authority (MA), each node will have a unique node_id value. To identify which MA a node belongs to, the names of the MA may be prepended to the node id, separated by a period. This notation is not used within the MA functions, but can be useful to identify a machine and its respective MA outside of the system.

6.3. Principals

Principals, or users of the system, are identified by their email address.

Chapter 2. Interface Selection

1. Overview

The interfaces defined in this specification can be selected and implemented as a whole, or in several different combinations, depending on the desired functionality that is to be supported. The operations defined in this document are grouped according to the general functionalilty they provide; each operation group is presented in its own Chapter. Four configurations, each corresponding to a subset of the operation groups, are viable.

2. Management Authority

A *management authority* (MA) is responsible for configuring and managing a set of nodes. A compliant MA must support the following operation groups:

1. Authentication Interface
2. Namespace Identification Interface
3. Principal Management Interface
4. Node Management Interface

3. Slice Authority

A *slice authority* (SA) is responsible for creating and managing a set of slices. A compliant SA must support the following operation groups:

1. Authentication Interface
2. Namespace Identification Interface
3. Principal Management Interface
4. Slice Management Interface
5. Slice Attribute Management Interface
6. Slice Principal Assignment Interface
7. Slice Node Assignment Interface

4. Naming Authority

Although not officially defined by the PlanetLab architecture, it is possible to implement a subset of complete SA that only returns tickets for valid slices, but does not serve as a front-end to a slice creation service. We refer to such a subset as a *naming authority* (NA). A compliant NA must support the following operation groups:

1. Authentication Interface
2. Namespace Identification Interface
3. Slice Management Interface

5. Full PlanetLab Central

A ful PlanetLab Central (PLC) implements all the operation groups:

1. Authentication Interface
2. Namespace Identification Interface
3. Principal Management Interface
4. Slice Management Interface
5. Slice Attribute Management Interface
6. Slice Principal Assignment Interface
7. Slice Node Assignment Interface

Chapter 3. Authentication Interface

The functions specified here are the top level authentication entry points. Before any other MA or SA calls can be made, this function must be called, and the resultant session key used for subsequent calls. The session key returned (if unencrypted) should be an alphanumeric string, unique (never before used), and at least 32 characters. Most implementations of this specification will want to extend the AuthenticatePrincipal and GetCertificate functions, to to take additional values to validate the caller. See the next section for examples, and further details on the session keys.

1. AuthenticatePrincipal

Prototype:

AuthenticatePrincipal (auth)

Description:

Authenticate a principal, and return a session key that can be used for other function calls.

Parameters:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call
email	string	false		The email address for the principal we want to authenticate

Successful return value:

Name	Type	Description
	string	The session key if authentication/authorization succesful

Return failure value:

Fault ID	Description
100	Authentication or authorization failure.
101	Authentication structure invalid.
102	Principal may not authenticate.
103	Principal not authorized.
901	Argument type incorrect.
902	Omitted a required extension.

2. GetCertificate

Prototype:

GetCertificate (auth, public_key)

Description:

Return a certificate for the specified public key.

Parameters:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call
email	string	false		The email address for the principal we want to authenticate
public_key	string	false		The user's public key this certificate is for.

Successful return value:

Name	Type	Description
	string	The certificate, if user authentication successful, in XML format

Return failure value:

Fault ID	Description
100	Authentication or authorization failure.
101	Authentication structure invalid.
102	Principal may not authenticate.
103	Principal not authorized.
901	Argument type incorrect.
902	Omitted a required extension.

3. AuthenticateCertificate

Prototype:

AuthenticateCertificate (auth)

Description:

Authenticate a previously issued certificate, and return an encrypted session key that can be used for other function calls.

Parameters:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call
certificate	string	false		The contents of the certificate in XMLSEC format

Successful return value:

Name	Type	Description
	string	The session key, encrypted with the public key specified if authentication/authorization successful

Return failure value:

Fault ID	Description
100	Authentication or authorization failure.
101	Authentication structure invalid.
102	Principal may not authenticate.
103	Principal not authorized.
901	Argument type incorrect.

4. InvalidateSession

Prototype:

InvalidateSession (auth)

Description:

Immediately invalidate the specified session, instead of waiting for it to expire. Afterwards, no calls can be made with the session, and the user will have to reauthenticate to obtain a new one

Parameters:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call
session	string	false		The session to invalidate.

Successful return value:

Name	Type	Description
	integer	Always return 1, regardless of whether or not session was initially valid.

Return failure value:

Fault ID	Description
101	Authentication structure invalid.

5. Supporting Password Authentication for Principals

As one concrete example of extending the authentication functions, we will add the commonly used password based authentication scheme to the AuthenticatePrincipal function. This involves only adding one value to the auth parameter of this function. The new function prototype would look like:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call
email	string	false		The email address for the principal we want to authenticate

Name	Type	Optional	Default	Description
password	string	false		New: The password of the principal.

The password could either be set upon registration in the AddPrincipal call, or, through an additional function to handle resetting and assigning passwords. Once the password is assigned to the principal, a session key can be obtained:

```
auth_val['email']= "user_email@domain.edu"
auth_val['password']= "mypassword"

session_key= AuthenticatePrincipal( auth )
```

Now, the caller could take their session key to any other function to make calls as desired. This same modification could be made to the GetCertificate call, to validate a principal before handing them a certificate.

6. Session Key Lifetime

Not specified in this document is the lifetime of session keys returned from the two authentication functions, AuthenticatePrincipal and AuthenticateCertificate. It is left up to the implementor to decide how long to make the session keys valid for. Some options include:

- Adding a 'requested lifetime' parameter to the authentication functions. If local policy allows the user the specified lifetime, then the key will be valid for that long.
- Enforcing a fixed lifetime on all session keys, set to a number of hours or days.
- Making session keys valid for an unlimited amount of time, or until explicitly invalidated through a new function.

7. Differing Caller Roles

Although the notion of caller roles has been explicitly left out of this document, there are times when one caller will have different levels of access to the system than another, and as a result will affect the returned data from a function call. Or, these levels of access could prevent callers from making some calls at all. There are fundamentally two different potential users of the specification functions: principals, or real people, and nodes, or the services running on those nodes.

One example of this would be the set of slices returned by the function GetSlices. Certain Slice Authority implementations may want to make this set of slices private, or only available to administrators of the system. Therefor, administrators would be able to see all slices, but a user of a slice, when calling the function, may only see the slices they are a member of. Except where explicitly mentioned in the function documentation, this left of access is left up to the implementors.

Chapter 4. Namespace Identification Interface

Returns the unique namespace for this system.

1. GetNamespace

Prototype:

GetNamespace (auth)

Description:

Return the namespace identifier for this system.

Parameters:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call.
session	string	false		The session value of the principal

Successful return value:

Name	Type	Description
	string	The namespace for this system

Return failure value:

Fault ID	Description
100	Authentication or authorization failure.

Chapter 5. Principal Management Interface

This interface provides basic principal management functions.

1. AddPrincipal

Prototype:

AddPrincipal (auth, principal_values)

Description:

Add a new principal record.

Parameters:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call.
session	string	false		The session value of the principal
principal_values	struct	false		Values for the new principal record.
first_name	string	false		The first name of the principal
last_name	string	false		The last name of the principal
email	string	false		The email address of the principal

Successful return value:

Name	Type	Description
	integer	1 if addition successful

Return failure value:

Fault ID	Description
100	Authentication or authorization failure.
101	Authentication structure invalid.
103	Principal not authorized.
203	The principal email address is not a valid email address.
204	A principal record with a matching email address already exists.
901	Argument type incorrect.
902	Omitted a required extension.

2. UpdatePrincipal

Prototype:

UpdatePrincipal (auth, email, update_values)

Description:

Update an existing principal record.

Parameters:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call.
session	string	false		The session value of the principal
email	string	false		The principal to update
update_values	struct	false		New values for the principal record. Those not specified are not updated
first_name	string	true		New first name
last_name	string	true		New last name
email	string	true		New email address

Successful return value:

Name	Type	Description
	integer	1 if update successful

Return failure value:

Fault ID	Description
100	Authentication or authorization failure.
101	Authentication structure invalid.
103	Principal not authorized.
200	One or more of the specified principals do not exist.
203	The principal email address is not a valid email address.
204	A principal record with a matching email address already exists.
901	Argument type incorrect.
902	Omitted a required extension.

3. GetPrincipals

Prototype:

GetPrincipals (auth, principal_email_list, return_fields)

Description:

Return details for the specified principal records. If return_fields is specified, return only those named principal record values (accepted values are the keys in the return type). If return_fields is not specified, all principal record values should be returned.

Parameters:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call.

Name	Type	Optional	Default	Description
session	string	false		The session value of the principal
principal_email_list	array of type string	true		Principals to get details for
return_fields	array of type string	true		Record field names that should be returned

Successful return value:

Name	Type	Description
	array of type struct	All principal details
first_name	string	Principal first name
last_name	string	Principal last name
email	string	Principal email address

Return failure value:

Fault ID	Description
100	Authentication or authorization failure.
101	Authentication structure invalid.
103	Principal not authorized.
200	One or more of the specified principals do not exist.
901	Argument type incorrect.
902	Omitted a required extension.

4. DeletePrincipal

Prototype:

DeletePrincipal (auth, email)

Description:

Delete a principal record.

Parameters:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call.
session	string	false		The session value of the principal
email	string	false		The principal to delete

Successful return value:

Name	Type	Description
	integer	1 if principal deleted

Return failure value:

Fault ID	Description
100	Authentication or authorization failure.
101	Authentication structure invalid.
103	Principal not authorized.
200	One or more of the specified principals do not exist.
901	Argument type incorrect.
902	Omitted a required extension.

Chapter 6. Principal Key Management Interface

This interface provides basic functions to handle associating a set of public keys used primarily for node access, though their use may include verifying signed packages or code for use in slice deployment.

1. AddPrincipalKey

Prototype:

AddPrincipalKey (auth, email, type, key)

Description:

Add a new public key to the specified principal record. If the key already exists, the call should also return successful with the key_id already assigned to that key.

Parameters:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call.
session	string	false		The session value of the principal
email	string	false		The principal to add this key for
type	string	false		The type of the key; see GetAllKeyTypes
key	string	false		The value of the key

Successful return value:

Name	Type	Description
	integer	Resultant new key_id, >0

Return failure value:

Fault ID	Description
100	Authentication or authorization failure.
101	Authentication structure invalid.
103	Principal not authorized.
200	One or more of the specified principals do not exist.
201	Principal key is invalid, or does not match specified type.
205	The principal key type is not recognized.
901	Argument type incorrect.
902	Omitted a required extension.

2. GetPrincipalKeys

Prototype:

GetPrincipalKeys (auth, email)

Description:

Return all public keys the principal has assigned.

Parameters:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call.
session	string	false		The session value of the principal
email	string	false		The id of the principal to return keys for

Successful return value:

Name	Type	Description
	array of type struct	All the principal keys; should be empty array if principal has no assigned keys
key_id	integer	The unique identifier for this key
type	string	The type of this key
key	string	The actual key value

Return failure value:

Fault ID	Description
100	Authentication or authorization failure.
101	Authentication structure invalid.
103	Principal not authorized.
200	One or more of the specified principals do not exist.
901	Argument type incorrect.
902	Omitted a required extension.

3. DeletePrincipalKey

Prototype:

DeletePrincipalKey (auth, email, key_id)

Description:

Delete the principal's specified key.

Parameters:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call.
session	string	false		The session value of the principal

Name	Type	Optional	Default	Description
email	string	false		The principal record to delete keys from
key_id	integer	false		Principal key to delete

Successful return value:

Name	Type	Description
	integer	1 if specified key was deleted

Return failure value:

Fault ID	Description
100	Authentication or authorization failure.
101	Authentication structure invalid.
103	Principal not authorized.
200	One or more of the specified principals do not exist.
202	One or more of the specified principal keys do not exist.
901	Argument type incorrect.
902	Omitted a required extension.

4. GetAllKeyTypes

Prototype:

GetAllKeyTypes (auth)

Description:

Return all current supported principal key types.

Parameters:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call.
session	string	false		The session value of the principal

Successful return value:

Name	Type	Description
	array of type string	All recognized key types

Return failure value:

Fault ID	Description
100	Authentication or authorization failure.
101	Authentication structure invalid.

Fault ID	Description
103	Principal not authorized.
902	Omitted a required extension.

Chapter 7. Node Management Interface

This interface provides basic node management functions.

1. AddNode

Prototype:

AddNode (auth, node_values)

Description:

Add a new node record.

Parameters:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call.
session	string	false		The session value of the principal
node_values	struct	false		Values for the new node record
hostname	string	false		The fully qualified hostname
ipaddress	string	false		The primary IP address

Successful return value:

Name	Type	Description
	integer	Resultant new node_id, >0

Return failure value:

Fault ID	Description
100	Authentication or authorization failure.
101	Authentication structure invalid.
103	Principal not authorized.
300	A node record with a matching hostname already exists.
302	Node hostname is not a valid hostname.
901	Argument type incorrect.
902	Omitted a required extension.

2. UpdateNode

Prototype:

UpdateNode (auth, node_id, update_values)

Description:

Update an existing node record.

Parameters:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call.
session	string	false		The session value of the principal
node_id	integer	false		The identifier of the node to update
update_values	struct	true		New values for the node record; those not specified are not updated
hostname	string	true		A new fully qualified hostname
ipaddress	string	false		The primary IP address

Successful return value:

Name	Type	Description
	integer	1 if update successful

Return failure value:

Fault ID	Description
100	Authentication or authorization failure.
101	Authentication structure invalid.
103	Principal not authorized.
300	A node record with a matching hostname already exists.
301	One or more of the specified nodes do not exist.
302	Node hostname is not a valid hostname.
901	Argument type incorrect.
902	Omitted a required extension.

3. GetManagedNodes

Prototype:

GetManagedNodes (auth)

Description:

Return a list of all node_ids cooresponding to the nodes this Management Authority manages.

Parameters:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call.
session	string	false		The session value of the principal

Successful return value:

Name	Type	Description
	array of type integer	All node_ids managed by this MA

Return failure value:

Fault ID	Description
100	Authentication or authorization failure.
101	Authentication structure invalid.
103	Principal not authorized.

4. GetNodes

Prototype:

GetNodes (auth, node_id_list, return_fields)

Description:

Return details about the nodes specified. If no node_ids are given, return details for all nodes. If return_fields is set, return only those field values (accepted values are the keys in the return type). If return_fields is not set, all node record values should be returned.

Parameters:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call.
session	string	false		The session value of the principal
node_id_list	array of type integer	true		Node ids to get details for
return_fields	array of type string	true		Record field names that should be returned

Successful return value:

Name	Type	Description
	array of type struct	Details of the specified nodes
node_id	integer	The node_id for this node
hostname	string	The hostname for this node
ipaddress	string	The primary IP address

Return failure value:

Fault ID	Description
100	Authentication or authorization failure.
101	Authentication structure invalid.
103	Principal not authorized.

Fault ID	Description
301	One or more of the specified nodes do not exist.
901	Argument type incorrect.
902	Omitted a required extension.

5. DeleteNode

Prototype:

DeleteNode (auth, node_id)

Description:

Delete a node record.

Parameters:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call.
session	string	false		The session value of the principal
node_id	integer	false		The node to delete

Successful return value:

Name	Type	Description
	integer	1 if node deleted.

Return failure value:

Fault ID	Description
100	Authentication or authorization failure.
101	Authentication structure invalid.
103	Principal not authorized.
301	One or more of the specified nodes do not exist.
901	Argument type incorrect.
902	Omitted a required extension.

Chapter 8. Slice Management Interface

Handles basic slice management, including creation, deletion, and renewal. Also manages set of peered Management Authorities

1. CreateSlice

Prototype:

CreateSlice (auth, slice_name)

Description:

Register a new slice name. Once the slice name is registered, the slice can be configured by assigning principals, setting slice attributes, and optionally assigning nodes (dependendent on SA implementation).

Parameters:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call.
session	string	false		The session value of the principal
slice_name	string	false		Name of the slice to register

Successful return value:

Name	Type	Description
	integer	Returns 1 if slice name registered

Return failure value:

Fault ID	Description
100	Authentication or authorization failure.
101	Authentication structure invalid.
103	Principal not authorized.
404	A slice with that name already exists.
901	Argument type incorrect.
902	Omitted a required extension.

2. DeleteSlice

Prototype:

DeleteSlice (auth, slice_name)

Description:

Delete a slice. The specified slice should be deleted, making the slice name available for reuse.

Parameters:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call.
session	string	false		The session value of the principal
slice_name	string	false		Name of the slice to delete.

Successful return value:

Name	Type	Description
	integer	1 if slice deleted

Return failure value:

Fault ID	Description
100	Authentication or authorization failure.
101	Authentication structure invalid.
103	Principal not authorized.
400	One or more of the specified slices do not exist.
901	Argument type incorrect.
902	Omitted a required extension.

3. GetSlices

Prototype:

GetSlices (auth)

Description:

Return names of existing slices. Services that are responsible for slice creation on nodes that call this function should be returned all the slices to be created on the calling node.

Parameters:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call.
session	string	false		The session value of the principal

Successful return value:

Name	Type	Description
	array of type string	Existing slice names

Return failure value:

Fault ID	Description
100	Authentication or authorization failure.

Fault ID	Description
101	Authentication structure invalid.
103	Principal not authorized.
902	Omitted a required extension.

4. UpdateSliceInfo

Prototype:

UpdateSliceInfo (auth, slice_name, info_values)

Description:

Update slice information values that are not critical to the behaviour or operation of the slice, like url and description.

Parameters:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call.
session	string	false		The session value of the principal
slice_name	string	false		The name of the slice to update
info_values	struct	false		New values for the slice. Those not specified are not updated.
url	string	true		The new URL for the slice
description	string	true		The new description for the slice

Successful return value:

Name	Type	Description
	integer	1 if slice updated

Return failure value:

Fault ID	Description
100	Authentication or authorization failure.
101	Authentication structure invalid.
103	Principal not authorized.
400	One or more of the specified slices do not exist.
901	Argument type incorrect.
902	Omitted a required extension.

5. GetSliceInfo

Prototype:

GetSliceInfo (auth, slice_name)

Description:

Return slice information values that are not critical to the behaviour or operation of the slice.

Parameters:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call.
session	string	false		The session value of the principal
slice_name	string	false		The name of the slice to update

Successful return value:

Name	Type	Description
	struct	Slice information values
url	string	URL for the slice
description	string	Description for the slice

Return failure value:

Fault ID	Description
100	Authentication or authorization failure.
101	Authentication structure invalid.
103	Principal not authorized.
400	One or more of the specified slices do not exist.
901	Argument type incorrect.
902	Omitted a required extension.

6. StartSlice

Prototype:

StartSlice (auth, slice_name)

Description:

Allow a new slice to begin running, or restart a stopped slice. If the slice has not be started before, after this call, it will be created through normal Slice Authority/Slice Creation Service communication paths.

Parameters:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call.
session	string	false		The session value of the principal
slice_name	string	false		Name of the slice to start

Successful return value:

Name	Type	Description
	integer	1 if slice started, or pending being started

Return failure value:

Fault ID	Description
100	Authentication or authorization failure.
101	Authentication structure invalid.
103	Principal not authorized.
400	One or more of the specified slices do not exist.
405	The specified slice has already started.
901	Argument type incorrect.
902	Omitted a required extension.

7. StopSlice

Prototype:

StopSlice (auth, slice_name)

Description:

Stop a slice. Generally, the slice should cease running, and no longer be available to principals. Slices started with either StartSlice or GetSliceTicket can be stopped using this call.

Parameters:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call.
session	string	false		The session value of the principal
slice_name	string	false		Name of the slice to stop

Successful return value:

Name	Type	Description
	integer	1 if slice stopped, or pending being stopped

Return failure value:

Fault ID	Description
100	Authentication or authorization failure.
101	Authentication structure invalid.
103	Principal not authorized.
400	One or more of the specified slices do not exist.
406	The specified slice has already stopped.

Fault ID	Description
901	Argument type incorrect.
902	Omitted a required extension.

8. GetSliceTicket

Prototype:

GetSliceTicket (auth, slice_name)

Description:

Start a slice by obtain a ticket for the named slice. A slice ticket is a full XML document containing all details of the slice that can be directly taken to the Slice Creation Service on a node. The format of this XML is defined in the PlanetLab Architecture PDN (see Introduction).

Parameters:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call.
session	string	false		The session value of the principal
slice_name	string	false		Name of the slice to get a ticket for

Successful return value:

Name	Type	Description
	string	Full slice ticket in XML format

Return failure value:

Fault ID	Description
100	Authentication or authorization failure.
101	Authentication structure invalid.
103	Principal not authorized.
400	One or more of the specified slices do not exist.
901	Argument type incorrect.
902	Omitted a required extension.

Chapter 9. Slice Attribute Management Interface

Handles slice resources, or attributes and their types. An implementation may choose to provide additional functions that allow for a default set of slice attributes to be automatically assigned to newly created slices, such as bandwidth limits, or VM types.

1. SetSliceAttribute

Prototype:

SetSliceAttribute (auth, slice_name, attributetype_name, attribute_value)

Description:

Set a slice attribute to a particular value. To list available slice attribute types, refer to the function SliceAttributeTypeGet. The attribute value must match the type of the named attribute. This method can only be used to set attributes of singular kind - that is those attribute of which there can be at most one per slice. If the named attribute is already set to a value for the named slice, the attribute will be assigned the new value.

Parameters:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call.
session	string	false		The session value of the principal
slice_name	string	false		Name of slice to set attribute on
attributetype_name	string	false		Name of the attribute type
attribute_value	struct	false		Value to set the attribute to, with name being the keys, and values being the attribute value

Successful return value:

Name	Type	Description
	integer	1 if attribute was set

Return failure value:

Fault ID	Description
100	Authentication or authorization failure.
101	Authentication structure invalid.
103	Principal not authorized.
400	One or more of the specified slices do not exist.
401	No slice attribute type matches that name.
402	The specified attribute value does not match the type.
901	Argument type incorrect.
902	Omitted a required extension.

2. AddSliceAttribute

Prototype:

AddSliceAttribute (auth, slice_name, attributetype_name, attribute_value)

Description:

Add an attribute with a particular value to the named slice.

Parameters:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call.
session	string	false		The session value of the principal
slice_name	string	false		Name of slice to add the attribute to
attributetype_name	string	false		Name of the attribute type
attribute_value	struct	false		Value to set the attribute to, with name being the keys, and values being the attribute value

Successful return value:

Name	Type	Description
	integer	1 if attribute was added

Return failure value:

Fault ID	Description
100	Authentication or authorization failure.
101	Authentication structure invalid.
103	Principal not authorized.
400	One or more of the specified slices do not exist.
401	No slice attribute type matches that name.
402	The specified attribute value does not match the type.
901	Argument type incorrect.
902	Omitted a required extension.

3. GetSliceAttributes

Prototype:

GetSliceAttributes (auth, slice_name)

Description:

Return all the assigned slice attributes.

Parameters:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call.
session	string	false		The session value of the principal
slice_name	string	false		Name of slice to return attributes for

Successful return value:

Name	Type	Description
	struct	All attribute values, keys are the attribute type name

Return failure value:

Fault ID	Description
100	Authentication or authorization failure.
101	Authentication structure invalid.
103	Principal not authorized.
400	One or more of the specified slices do not exist.
902	Omitted a required extension.

4. DeleteSliceAttribute

Prototype:

DeleteSliceAttribute (auth, slice_name, attributetype_name)

Description:

Delete a slice attribute from the named slice.

Parameters:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call.
session	string	false		The session value of the principal
slice_name	string	false		Name of slice to set attribute on
attributetype_name	string	false		Name of the attribute type to delete

Successful return value:

Name	Type	Description
	integer	1 if attribute deleted.

Return failure value:

Fault ID	Description
100	Authentication or authorization failure.

Fault ID	Description
101	Authentication structure invalid.
103	Principal not authorized.
400	One or more of the specified slices do not exist.
401	No slice attribute type matches that name.
901	Argument type incorrect.
902	Omitted a required extension.

5. CreateSliceAttributeType

Prototype:

CreateSliceAttributeType (auth, name, max_per_slice, value_fields)

Description:

Create a new slice attribute type. There are no functions to update an existing slice attribute type. To accomplish this, first delete the existing type, and readd the new type.

Parameters:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call.
session	string	false		The session value of the principal
name	string	false		Name of the new slice attribute
max_per_slice	integer	false		The maximum number of instances of this attribute a slice can have. Can be zero, effectively not allowing the attribute to be added to a slice.
value_fields	array of type struct	false		Value specifications for this type
name	string	false		The name of the value
type	string	false		The type of the value; basic types should be: 'integer', 'string', or 'float'

Successful return value:

Name	Type	Description
	integer	1 if attribute type created

Return failure value:

Fault ID	Description
100	Authentication or authorization failure.
101	Authentication structure invalid.
103	Principal not authorized.

Fault ID	Description
403	An attribute type with that name already exists.
901	Argument type incorrect.
902	Omitted a required extension.

6. GetSliceAttributeType

Prototype:

GetSliceAttributeType (auth, attributetype_name)

Description:

Return slice attribute type details.

Parameters:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call.
session	string	false		The session value of the principal
attributetype_name	string	true		Attribute type name to get the specification for; if ommited, all attribute types should be returned

Successful return value:

Name	Type	Description
	array of type struct	Attribute type details, one entry per type
name	string	The name of this type
num_values	integer	The number of values this type has
max_per_slice	integer	The max number of attributes of this type a slice can have
type	array of type struct	The specification details, each element has name/type keys.

Return failure value:

Fault ID	Description
100	Authentication or authorization failure.
101	Authentication structure invalid.
103	Principal not authorized.
401	No slice attribute type matches that name.
901	Argument type incorrect.
902	Omitted a required extension.

7. DeleteSliceAttributeType

Prototype:

DeleteSliceAttributeType (auth, attributetype_name)

Description:

Delete an attribute type. All slices with attributes of this type assigned must first have those attributes removed before the function can succeed.

Parameters:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call.
session	string	false		The session value of the principal
attributetype_name	string	false		Name of the attribute type to delete

Successful return value:

Name	Type	Description
	integer	1 if attribute type was deleted

Return failure value:

Fault ID	Description
100	Authentication or authorization failure.
101	Authentication structure invalid.
103	Principal not authorized.
401	No slice attribute type matches that name.
901	Argument type incorrect.
902	Omitted a required extension.

Chapter 10. Slice Principal Assignment Interface

Manages slice principal assignment.

1. AddSlicePrincipal

Prototype:

AddSlicePrincipal (auth, slice_name, principal_list)

Description:

Add principals to the named slice.

Parameters:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call.
session	string	false		The session value of the principal
slice_name	string	false		The name of the slice to add users to
principal_list	array of type string	false		List of principals, by email address, to add

Successful return value:

Name	Type	Description
	integer	1 if principal(s) assigned to slice

Return failure value:

Fault ID	Description
100	Authentication or authorization failure.
101	Authentication structure invalid.
103	Principal not authorized.
200	One or more of the specified principals do not exist.
400	One or more of the specified slices do not exist.
901	Argument type incorrect.
902	Omitted a required extension.

2. DeleteSlicePrincipal

Prototype:

DeleteSlicePrincipal (auth, slice_name, principal_list)

Description:

Remove principals from the named slice. All principals in the list will be removed from the slice; all principals that are in the list, but not in the slice, should be ignored.

Parameters:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call.
session	string	false		The session value of the principal
slice_name	string	false		The name of the slice to remove users from
principal_list	array of type string	false		list of principals, by email address, to remove

Successful return value:

Name	Type	Description
	integer	1 if principal(s) removed from slice

Return failure value:

Fault ID	Description
100	Authentication or authorization failure.
101	Authentication structure invalid.
103	Principal not authorized.
200	One or more of the specified principals do not exist.
400	One or more of the specified slices do not exist.
901	Argument type incorrect.
902	Omitted a required extension.

3. ListSlicePrincipals

Prototype:

ListSlicePrincipals (auth, slice_name)

Description:

List all the principals assigned to the named slice.

Parameters:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call.
session	string	false		The session value of the principal
slice_name	string	false		The name of the slice to list principals

Successful return value:

Name	Type	Description
	array of type string	All the principals, by email address, assigned to the slice

Return failure value:

Fault ID	Description
100	Authentication or authorization failure.
101	Authentication structure invalid.
103	Principal not authorized.
400	One or more of the specified slices do not exist.
901	Argument type incorrect.

4. GetResponsibleSlicePrincipals

Prototype:

GetResponsibleSlicePrincipals (auth, slice_name)

Description:

List all the principals responsible for the named slice. This differs from ListSlicePrincipals in that it also should return principal investigators and any other non-slice principal that answers for the operation of this slice.

Parameters:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call.
session	string	false		The session value of the principal
slice_name	string	false		The name of the slice to list principals

Successful return value:

Name	Type	Description
	array of type string	All the principals, by email address, responsible for the slice

Return failure value:

Fault ID	Description
100	Authentication or authorization failure.
101	Authentication structure invalid.
103	Principal not authorized.
400	One or more of the specified slices do not exist.
901	Argument type incorrect.

Chapter 11. Slice Node Assignment Interface

Manages slice node assignment and the set of peered Management Authorities.

1. AddMAPeer

Prototype:

AddMAPeer (auth, ma_namespace)

Description:

Add the specified Management Authority to the list of MAs that we will allow slices to be created on.

Parameters:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call.
session	string	false		The session value of the principal
ma_namespace	string	false		The namespace of the specified MA

Successful return value:

Name	Type	Description
	integer	1 if MA was added to peered MA list.

Return failure value:

Fault ID	Description
100	Authentication or authorization failure.
101	Authentication structure invalid.
103	Principal not authorized.
901	Argument type incorrect.
902	Omitted a required extension.

2. RemoveMAPeer

Prototype:

RemoveMAPeer (auth, ma_namespace)

Description:

Remove the specified Management Authority from the list of MAs that we will allow slices to be created on. Any slices that had nodes assigned from this MA will have those nodes removed.

Parameters:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call.
session	string	false		The session value of the principal
ma_namespace	string	false		The namespace of the specified MA

Successful return value:

Name	Type	Description
	integer	1 if MA was removed from the peered MA list.

Return failure value:

Fault ID	Description
100	Authentication or authorization failure.
101	Authentication structure invalid.
103	Principal not authorized.
901	Argument type incorrect.
902	Omitted a required extension.

3. GetMAPeers

Prototype:

GetMAPeers (auth)

Description:

Return all namespace identifiers and associated details for peered Management Authorities.

Parameters:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call.
session	string	false		The session value of the principal

Successful return value:

Name	Type	Description
	array of type struct	List of peered MA namespace identifiers
ma_namespace	string	The MA namespace

Return failure value:

Fault ID	Description
100	Authentication or authorization failure.

Fault ID	Description
101	Authentication structure invalid.
103	Principal not authorized.
901	Argument type incorrect.
902	Omitted a required extension.

4. AddSliceNode

Prototype:

AddSliceNode (auth, slice_name, nodes_list, ma_namespace)

Description:

Add nodes to the named slice. If any nodes that are specified do not exist, the call should fail and not affect the set of nodes assigned to a slice. If ma_namespace is specified, all the nodes in nodes_list are assumed to belong to that namespace.

Parameters:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call.
session	string	false		The session value of the principal
slice_name	string	false		The name of the slice to add users to
nodes_list	array of type integer	false		list of node identifiers to add
ma_namespace	string	true		which Management Authority these nodes belong to. If blank, and the implementation of the the SA also includes the MA interfaces, the current MA namespace should be assumed. If just a SA (or naming authority), this must be specified.

Successful return value:

Name	Type	Description
	integer	1 if all specified nodes were added to the slice

Return failure value:

Fault ID	Description
100	Authentication or authorization failure.
101	Authentication structure invalid.
103	Principal not authorized.
301	One or more of the specified nodes do not exist.
400	One or more of the specified slices do not exist.
901	Argument type incorrect.
902	Omitted a required extension.

5. DeleteSliceNode

Prototype:

DeleteSliceNode (auth, slice_name, nodes_list, ma_namespace)

Description:

Remove nodes from the named slice. All nodes in the list will be removed from the slice. If any nodes that are specified do not exist, the call should fail and not affect the set of nodes assigned to a slice. If ma_namespace is specified, all the nodes in nodes_list are assumed to belong to that namespace.

Parameters:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call.
session	string	false		The session value of the principal
slice_name	string	false		The name of the slice to remove users from
nodes_list	array of type integer	false		List of node identifiers to remove
ma_namespace	string	true		which Management Authority these nodes belong to. If blank, and the implementation of the the SA also includes the MA interfaces, the current MA namespace should be assumed. If just a SA (or naming authority), this must be specified.

Successful return value:

Name	Type	Description
	integer	1 if all specified nodes were removed from the slice

Return failure value:

Fault ID	Description
100	Authentication or authorization failure.
101	Authentication structure invalid.
103	Principal not authorized.
301	One or more of the specified nodes do not exist.
400	One or more of the specified slices do not exist.
901	Argument type incorrect.
902	Omitted a required extension.

6. ListSliceNodes

Prototype:

ListSliceNodes (auth, slice_name)

Description:

List all the nodes assigned to the named slice, seperated into groups by their respective Management Authority.

Parameters:

Name	Type	Optional	Default	Description
auth	struct	false		Contains authentication values for this call.
session	string	false		The session value of the principal
slice_name	string	false		The name of the slice to list nodes

Successful return value:

Name	Type	Description
	struct	A list of all the nodes ids assigned to the slice, organized by the MA they are a member of.
ma_namespace	array of type integer	An array of the node_ids that are a part of the specified MA

Return failure value:

Fault ID	Description
100	Authentication or authorization failure.
101	Authentication structure invalid.
103	Principal not authorized.
400	One or more of the specified slices do not exist.
901	Argument type incorrect.
902	Omitted a required extension.

Chapter 12. Examples

1. Example Implementation

For an example of an implemented, fully compliant API, refer to the PlanetLab Central API. Documentation on that api is available at http://www.planet-lab.org/doc/plc_api [http://www.planet-lab.org/doc/plc_api/]. Source code for this can be obtained from the PlanetLab CVS system, which can be accessed using the information avaliable at http://www.planet-lab.org/CVSinfo/.

2. Example Sessions

Below are several sessions showing calls made to perform common opertions. The syntax shown below is technically Python syntax, but should be generic enough to be understood by most readers. Comments are preceeded by the hash character, #. For arguments of type struct, which are associative arrays, they will look like:

```
# initialize as a struct/associative array
struct_variable= {}
# assign a value to a key
struct_variable['key_name']= key_value
```

Exact details of how to make XML RPC calls to an API server depend on the specifics of the language being used. For the examples shown here, calls made to the API server will be prefixed with "api.". For example, a call to GetNamespace, returning the result in the variable named 'return_val', and passing in one argument, will look like:

```
return_val= api.GetNamespace( auth )
```

2.1. Principal Password Authentication

Before any other calls can be made, the caller must be authenticated, and obtain a session value. This example uses the proposed authentication extension outlined in Chapter 3, Section 4. The resultant variable auth_struct will be used in the remaining examples.

```
# Create the required authentication structure for the caller

auth_request_struct= {}
auth_request_struct['email']= 'user_email@site.com'
auth_request_struct['password']= 'user_password'


# Authenticate caller and store the session value

user_session= api.AuthenticatePrincipal( auth_request_struct )


# Prepare new authentication structure for any future calls

auth_struct= {}
auth_struct['session']= user_session

# auth_struct can now be used in calls that require an authenticated
# principal
```

2.2. Principal Certificate Authentication

An alternative to using username and password authentication for each and every session is to obtain a certificate, which can later be used to obtain sessions. This required username and password the first time, but once a certificate is obtained, that information will no longer need to be sent to the API server as long as the certificate is valid.

```
# Create the required authentication structure for the caller

auth_request_struct= {}
auth_request_struct['email']= 'user_email@site.com'
auth_request_struct['password']= 'user_password'


# Authenticate caller and get a certificate
# the public key parameter should contain a RSA public key in PEM format
# that corresponds to the users private key. The certificate will be
# bound this key pair.

user_certificate= api.GetCertificate( auth_request_struct, public_key )


# Now, the certificate contained in user_certificate should be stored for future
# use. Any time the user want to obtain a session for API calls, the certificate
# can be used as such:

# Prepare new authentication structure using certificate instead of username
# and password

auth_request_struct= {}
auth_request_struct['certificate']= user_certificate


# Authenticate a previously issued certificate

encrypted_user_session= api.AuthenticateCertificate( auth_request_struct )

# Finally, the value returned here in encrypted_user_session is a session_value
# encrypted with the public key contained in the certificate. The caller will
# now need to use their private key to decrypt the session for use in future
# API calls.
```