# Delegation Service

## Delegation Methods

### InitiateDelegation()

#### Inputs:

*None*

#### Outputs:

*Certificate Request*

#### Faults:

DelegationInternalFault

#### Description:

This method allows the user to contact the delegation service such that they may begin to delegate their credentials to the delegation service. This method does not require any inputs, however does require that the user authenticate using their grid credentials. Upon being called this method will generate a public/private key pair and corresponding certificate request. The certificate request will be based on the identity supplied in the credentials the used to authenticate. The certificate request will contain the generated public key and will be signed by the generated private key. The certificate request will be returned to the user.

### DelegateCredentials()

#### Inputs:

*Certificate[]*

#### Outputs:

*None*

#### Faults:

DelegationInternalFault, InvalidCertificateFault

#### Description:

This method is used after the certificate request obtained from the *IntiateDelegation* methodhas been signed by the user to upload the resultant certificate. The resultant certificate contains the public key and is bound to the private key generated by the *InitiateDelegation.* In order to call the DelegateCredentials method, the user must authenticate with their grid credentials. The Identity in the grid credentials must be the same as the identity in the certificate being uploaded and be the same identity used in calling the *IntiateDelegation* method. As input this method takes the certificate chain starting with the user’s long term certificate, followed any proxy certificates in the delegation chain, and ending with the resultant certificate.

### grantPolicy()

#### Inputs:

*DelegationPolicy*

#### Outputs:

*PolicyIdentifier*

#### Faults:

DelegationInternalFault, InvalidPolicyFault, NoCredentialFoundFault

#### Description:

This method allows a user to grant a delegation policy, a delegation policy specifies a list of users that may obtain delegated credentials, how long the delegated credentials may be created for, and a lifetime that the policy is valid. This method assumes that the delegation services has or has already been given a user’s credentials to delegate from. This method also requires that the user whom the policy is being granted for, authenticate using their grid credentials. This method returns a PolicyIdentifier, identifying the delegation policy.

### revokePolicy()

#### Inputs:

*PolicyIdentifier*

#### Outputs:

*None*

#### Faults:

DelegationInternalFault, InvalidPolicyFault

#### Description:

This method allows a user to revoke a delegation policy that was previously granted. This method requires that the user whose policy will be revoked, authenticate using their grid credentials.

### getPolicies()

#### Inputs:

*PolicyFilter*

#### Outputs:

*None*

#### Faults:

DelegationInternalFault

#### Description:

This method allows a user to obtain a list of delegation policies they granted. In order to call this method a user must authenticate using their grid credentials.

### getCredential()

#### Inputs:

Grid Identity

*PublicKey*

#### Outputs:

*X509Certificate[]*

#### Faults:

DelegationInternalFault, PermissionDeniedFault

#### Description:

This method allows a third party to request a user’s credential such that they may act on a user’s behalf. As input this method requires the identity of the user whose credentials are desired as well as a public key. The third party must authenticate using their grid credentials. If a valid delegation policy exists, the delegation service will locate the user’s credentials and use them to create and sign a certificate containing the public key supplied by the third party. The delegation service will then return a certificate chain ending with the newly signed certificate.