

IceGrid-Admin

IceGrid::Admin

Overview

interface Admin

The IceGrid administrative interface.

Allowing access to this interface is a security risk! Please see the IceGrid documentation for further information.

Operation Index

addApplication — Add an application to IceGrid.
syncApplication — Synchronize a deployed application with the given application descriptor.
updateApplication — Update a deployed application with the given update application descriptor.
syncApplicationWithoutRestart — Synchronize a deployed application with the given application descriptor.
updateApplicationWithoutRestart — Update a deployed application with the given update application descriptor only if no server restarts are necessary for the update of the application.
removeApplication — Remove an application from IceGrid.
instantiateServer — Instantiate a server template from an application on the given node.
patchApplication — Patch the given application data.
getApplicationInfo — Get an application descriptor.
getDefaultValueDescriptor — Get the default application descriptor.
getAllApplicationNames — Get all the IceGrid applications currently registered.
getServerInfo — Get the server information for the server with the given id.
getServerState — Get a server's state.
getServerPid — Get a server's system process id.
getServerAdminCategory — Get the category for server admin objects.
getServerAdmin — Get a proxy to the server's admin object.
enableServer — Enable or disable a server.
isServerEnabled — Check if the server is enabled or disabled.
startServer — Start a server and wait for its activation.
stopServer — Stop a server.
patchServer — Patch a server.
sendSignal — Send signal to a server.
writeMessage — Write message on server stdout or stderr. (*Deprecated*)
getAllServerIds — Get all the server ids registered with IceGrid.
getAdapterInfo — Get the adapter information for the replica group or adapter with the given id.
removeAdapter — Remove the adapter with the given id.
getAllAdapterIds — Get all the adapter ids registered with IceGrid.
addObject — Add an object to the object registry.
updateObject — Update an object in the object registry.
addObjectWithType — Add an object to the object registry and explicitly specify its type.
removeObject — Remove an object from the object registry.
getObjectInfo — Get the object info for the object with the given identity.
getObjectInfosByType — Get the object info of all the registered objects with the given type.
getAllObjectInfos — Get the object info of all the registered objects whose stringified identities match the given expression.
pingNode — Ping an IceGrid node to see if it is active.
getNodeLoad — Get the load averages of the node.
getNodeInfo — Get the node information for the node with the given name.
getNodeProcessorSocketCount — Get the number of physical processor sockets for the machine running the node with the given name.
shutdownNode — Shutdown an IceGrid node.
getNodeHostname — Get the hostname of this node.
getAllNodeNames — Get all the IceGrid nodes currently registered.
pingRegistry — Ping an IceGrid registry to see if it is active.
getRegistryInfo — Get the registry information for the registry with the given name.
shutdownRegistry — Shutdown an IceGrid registry.
getAllRegistryNames — Get all the IceGrid registries currently registered.
shutdown — Shut down the IceGrid registry.
getSliceChecksums — Returns the checksums for the IceGrid Slice definitions.

Operations

void addApplication(IceGrid::ApplicationDescriptor descriptor) throws IceGrid::AccessDeniedException, IceGrid::DeploymentException

Add an application to IceGrid.

Parameters

`descriptor` — The application descriptor.

Exceptions

`IceGrid::AccessDeniedException` — Raised if the session doesn't hold the exclusive lock or if another session is holding the lock.
`IceGrid::DeploymentException` — Raised if application deployment failed.

void syncApplication(IceGrid::ApplicationDescriptor descriptor) throws IceGrid::AccessDeniedException, IceGrid::DeploymentException, IceGrid::ApplicationNotExistException

Synchronize a deployed application with the given application descriptor. This operation will replace the current descriptor with this new descriptor.

Parameters

`descriptor` — The application descriptor.

Exceptions

`IceGrid::AccessDeniedException` — Raised if the session doesn't hold the exclusive lock or if another session is holding the lock.
`IceGrid::DeploymentException` — Raised if application deployment failed.
`IceGrid::ApplicationNotExistException` — Raised if the application doesn't exist.

void updateApplication(IceGrid::ApplicationUpdateDescriptor descriptor) throws IceGrid::AccessDeniedException, IceGrid::DeploymentException, IceGrid::ApplicationNotExistException

Update a deployed application with the given update application descriptor.

Parameters

`descriptor` — The update descriptor.

Exceptions

`IceGrid::AccessDeniedException` — Raised if the session doesn't hold the exclusive lock or if another session is holding the lock.
`IceGrid::DeploymentException` — Raised if application deployment failed.
`IceGrid::ApplicationNotExistException` — Raised if the application doesn't exist.

["ami"] void syncApplicationWithoutRestart(IceGrid::ApplicationDescriptor descriptor) throws IceGrid::AccessDeniedException, IceGrid::DeploymentException, IceGrid::ApplicationNotExistException

Synchronize a deployed application with the given application descriptor. This operation will replace the current descriptor with this new descriptor only if no server restarts are necessary for the update of the application. If some servers need to be restarted, the synchronization is rejected with a `DeploymentException`.

Parameters

`descriptor` — The application descriptor.

Exceptions

`IceGrid::AccessDeniedException` — Raised if the session doesn't hold the exclusive lock or if another session is holding the lock.
`IceGrid::DeploymentException` — Raised if application deployment failed.
`IceGrid::ApplicationNotExistException` — Raised if the application doesn't exist.

["ami"] void updateApplicationWithoutRestart(IceGrid::ApplicationUpdateDescriptor descriptor) throws IceGrid::AccessDeniedException, IceGrid::DeploymentException, IceGrid::ApplicationNotExistException

Update a deployed application with the given update application descriptor only if no server restarts are necessary for the update of the application. If some servers need to be restarted, the synchronization is rejected with a `DeploymentException`.

Parameters

`descriptor` — The update descriptor.

Exceptions

`IceGrid::AccessDeniedException` — Raised if the session doesn't hold the exclusive lock or if another session is holding the lock.
`IceGrid::DeploymentException` — Raised if application deployment failed.
`IceGrid::ApplicationNotExistException` — Raised if the application doesn't exist.

void removeApplication(string name) throws `IceGrid::AccessDeniedException`, `IceGrid::DeploymentException`, `IceGrid::ApplicationNotExistException`

Remove an application from IceGrid.

Parameters

`name` — The application name.

Exceptions

`IceGrid::AccessDeniedException` — Raised if the session doesn't hold the exclusive lock or if another session is holding the lock.
`IceGrid::ApplicationNotExistException` — Raised if the application doesn't exist.

void instantiateServer(string application, string node, `IceGrid::ServerInstanceDescriptor desc`) throws `IceGrid::AccessDeniedException`, `IceGrid::ApplicationNotExistException`, `IceGrid::DeploymentException`

Instantiate a server template from an application on the given node.

Parameters

`application` — The application name.
`node` — The name of the node where the server will be deployed.
`desc` — The descriptor of the server instance to deploy.

Exceptions

`IceGrid::AccessDeniedException` — Raised if the session doesn't hold the exclusive lock or if another session is holding the lock.
`IceGrid::DeploymentException` — Raised if server instantiation failed.
`IceGrid::ApplicationNotExistException` — Raised if the application doesn't exist.

["amd"] void patchApplication(string name, bool shutdown) throws `IceGrid::ApplicationNotExistException`, `IceGrid::PatchException`

Patch the given application data.

Parameters

`name` — The application name.
`shutdown` — If true, the servers depending on the data to patch will be shut down if necessary.

Exceptions

`IceGrid::ApplicationNotExistException` — Raised if the application doesn't exist.
`IceGrid::PatchException` — Raised if the patch failed.

["nonmutating"] `IceGrid::ApplicationInfo` getApplicationInfo(string name) throws `IceGrid::ApplicationNotExistException`

Get an application descriptor.

Parameters

`name` — The application name.

Return Value

The application descriptor.

Exceptions

[IceGrid::ApplicationNotExistException](#) — Raised if the application doesn't exist.

["nonmutating"] [IceGrid::ApplicationDescriptor](#) `getDefaultApplicationDescriptor()` throws [IceGrid::DeploymentException](#)

Get the default application descriptor.

Exceptions

[IceGrid::DeploymentException](#) — Raised if the default application descriptor can't be accessed or is invalid.

["nonmutating"] [Ice::StringSeq](#) `getAllApplicationNames()`

Get all the IceGrid applications currently registered.

Return Value

The application names.

["nonmutating"] [IceGrid::ServerInfo](#) `getServerInfo(string id)` throws [IceGrid::ServerNotExistException](#)

Get the server information for the server with the given id.

Parameters

`id` — The server id.

Return Value

The server information.

Exceptions

[IceGrid::ServerNotExistException](#) — Raised if the server doesn't exist.

["nonmutating"] [IceGrid::ServerState](#) `getServerState(string id)` throws [IceGrid::ServerNotExistException](#), [IceGrid::NodeUnreachableException](#), [IceGrid::DeploymentException](#)

Get a server's state.

Parameters

`id` — The server id.

Return Value

The server state.

Exceptions

[IceGrid::ServerNotExistException](#) — Raised if the server doesn't exist.

[IceGrid::NodeUnreachableException](#) — Raised if the node could not be reached.

[IceGrid::DeploymentException](#) — Raised if the server couldn't be deployed on the node.

["nonmutating"] `int getServerPid(string id)` throws [IceGrid::ServerNotExistException](#), [IceGrid::NodeUnreachableException](#), [IceGrid::DeploymentException](#)

Get a server's system process id. The process id is operating system dependent.

Parameters

`id` — The server id.

Return Value

The server's process id.

Exceptions

`IceGrid::ServerNotExistException` — Raised if the server doesn't exist.
`IceGrid::NodeUnreachableException` — Raised if the node could not be reached.
`IceGrid::DeploymentException` — Raised if the server couldn't be deployed on the node.

string getServerAdminCategory()

Get the category for server admin objects. You can manufacture a server admin proxy from the admin proxy by changing its identity: use the server ID as name and the returned category as category.

Return Value

The category for server admin objects.

Object* getServerAdmin(string id) throws `IceGrid::ServerNotExistException`, `IceGrid::NodeUnreachableException`, `IceGrid::DeploymentException`

Get a proxy to the server's admin object.

Parameters

`id` — The server id.

Return Value

A proxy to the server's admin object

Exceptions

`IceGrid::ServerNotExistException` — Raised if the server doesn't exist.
`IceGrid::NodeUnreachableException` — Raised if the node could not be reached.
`IceGrid::DeploymentException` — Raised if the server couldn't be deployed on the node.

void enableServer(string id, bool enabled) throws `IceGrid::ServerNotExistException`, `IceGrid::NodeUnreachableException`, `IceGrid::DeploymentException`

Enable or disable a server. A disabled server can't be started on demand or administratively. The enable state of the server is not persistent: if the node is shut down and restarted, the server will be enabled by default.

Parameters

`id` — The server id.

`enabled` — True to enable the server, false to disable it.

Exceptions

`IceGrid::ServerNotExistException` — Raised if the server doesn't exist.
`IceGrid::NodeUnreachableException` — Raised if the node could not be reached.
`IceGrid::DeploymentException` — Raised if the server couldn't be deployed on the node.

["nonmutating"] bool isServerEnabled(string id) throws `IceGrid::ServerNotExistException`, `IceGrid::NodeUnreachableException`, `IceGrid::DeploymentException`

Check if the server is enabled or disabled.

Parameters

`id` — The server id.

Exceptions

`IceGrid::ServerNotExistException` — Raised if the server doesn't exist.
`IceGrid::NodeUnreachableException` — Raised if the node could not be reached.
`IceGrid::DeploymentException` — Raised if the server couldn't be deployed on the node.

["amd"] void startServer(string id) throws `IceGrid::ServerNotExistException`, `IceGrid::ServerStartException`, `IceGrid::NodeUnreachableException`, `IceGrid::DeploymentException`

Start a server and wait for its activation.

Parameters

`id` — The server id.

Exceptions

`IceGrid::ServerNotExistException` — Raised if the server doesn't exist.
`IceGrid::ServerStartException` — Raised if the server couldn't be started.
`IceGrid::NodeUnreachableException` — Raised if the node could not be reached.
`IceGrid::DeploymentException` — Raised if the server couldn't be deployed on the node.

["amd"] void stopServer(string id) throws `IceGrid::ServerNotExistException`, `IceGrid::ServerStopException`, `IceGrid::NodeUnreachableException`, `IceGrid::DeploymentException`

Stop a server.

Parameters

`id` — The server id.

Exceptions

`IceGrid::ServerNotExistException` — Raised if the server doesn't exist.
`IceGrid::ServerStopException` — Raised if the server couldn't be stopped.
`IceGrid::NodeUnreachableException` — Raised if the node could not be reached.
`IceGrid::DeploymentException` — Raised if the server couldn't be deployed on the node.

["amd"] void patchServer(string id, bool shutdown) throws `IceGrid::ServerNotExistException`, `IceGrid::NodeUnreachableException`, `IceGrid::DeploymentException`, `IceGrid::PatchException`

Patch a server.

Parameters

`id` — The server id.
`shutdown` — If true, servers depending on the data to patch will be shut down if necessary.

Exceptions

`IceGrid::ServerNotExistException` — Raised if the server doesn't exist.
`IceGrid::NodeUnreachableException` — Raised if the node could not be reached.
`IceGrid::DeploymentException` — Raised if the server couldn't be deployed on the node.
`IceGrid::PatchException` — Raised if the patch failed.

void sendSignal(string id, string signal) throws `IceGrid::ServerNotExistException`, `IceGrid::NodeUnreachableException`, `IceGrid::DeploymentException`, `IceGrid::BadSignalException`

Send signal to a server.

Parameters

`id` — The server id.
`signal` — The signal, for example SIGTERM or 15.

Exceptions

`IceGrid::ServerNotExistException` — Raised if the server doesn't exist.
`IceGrid::NodeUnreachableException` — Raised if the node could not be reached.
`IceGrid::DeploymentException` — Raised if the server couldn't be deployed on the node.
`IceGrid::BadSignalException` — Raised if the signal is not recognized by the target server.

void writeMessage(string id, string message, int fd) throws `IceGrid::ServerNotExistException`, `IceGrid::NodeUnreachableException`, `IceGrid::DeploymentException`

Write message on server stdout or stderr.

This operation is deprecated as of version 3.3.

writeMessage is deprecated, use instead the Process facet of the server Admin object.

Parameters

`id` — The server id.
`message` — The message.
`fd` — 1 for stdout, 2 for stderr.

Exceptions

`IceGrid::ServerNotExistException` — Raised if the server doesn't exist.
`IceGrid::NodeUnreachableException` — Raised if the node could not be reached.
`IceGrid::DeploymentException` — Raised if the server couldn't be deployed on the node.

["nonmutating"] `Ice::StringSeq getAllServerIds()`

Get all the server ids registered with IceGrid.

Return Value

The server ids.

["nonmutating"] `IceGrid::AdapterInfoSeq getAdapterInfo(string id) throws IceGrid::AdapterNotExistException`

Get the adapter information for the replica group or adapter with the given id.

Parameters

`id` — The adapter id.

Return Value

A sequence of adapter information structures. If the given id refers to an adapter, this sequence will contain only one element. If the given id refers to a replica group, the sequence will contain the adapter information of each member of the replica group.

Exceptions

`IceGrid::AdapterNotExistException` — Raised if the adapter or replica group doesn't exist.

`void removeAdapter(string id) throws IceGrid::AdapterNotExistException, IceGrid::DeploymentException`

Remove the adapter with the given id.

Parameters

`id` — The adapter id.

Exceptions

`IceGrid::AdapterNotExistException` — Raised if the adapter doesn't exist.

["nonmutating"] `Ice::StringSeq getAllAdapterIds()`

Get all the adapter ids registered with IceGrid.

Return Value

The adapter ids.

`void addObject(Object* obj) throws IceGrid::ObjectExistsException, IceGrid::DeploymentException`

Add an object to the object registry. IceGrid will get the object type by calling `ice_id` on the given proxy. The object must be reachable.

Parameters

`obj` — The object to be added to the registry.

Exceptions

`IceGrid::ObjectExistsException` — Raised if the object is already registered.

`IceGrid::DeploymentException` — Raised if the object can't be added. This might be raised if the invocation on the proxy to get the object type failed.

void updateObject(Object* obj) throws [IceGrid::ObjectNotRegisteredException](#), [IceGrid::DeploymentException](#)

Update an object in the object registry. Only objects added with this interface can be updated with this operation. Objects added with deployment descriptors should be updated with the deployment mechanism.

Parameters

`obj` — The object to be updated to the registry.

Exceptions

`IceGrid::ObjectNotRegisteredException` — Raised if the object isn't registered with the registry.

`IceGrid::DeploymentException` — Raised if the object can't be updated. This might happen if the object was added with a deployment descriptor.

void addObjectWithType(Object* obj, string type) throws [IceGrid::ObjectExistsException](#), [IceGrid::DeploymentException](#)

Add an object to the object registry and explicitly specify its type.

Parameters

`obj` — The object to be added to the registry.

`type` — The object type.

Exceptions

`IceGrid::ObjectExistsException` — Raised if the object is already registered.

void removeObject(Ice::Identity id) throws [IceGrid::ObjectNotRegisteredException](#), [IceGrid::DeploymentException](#)

Remove an object from the object registry. Only objects added with this interface can be removed with this operation. Objects added with deployment descriptors should be removed with the deployment mechanism.

Parameters

`id` — The identity of the object to be removed from the registry.

Exceptions

`IceGrid::ObjectNotRegisteredException` — Raised if the object isn't registered with the registry.

`IceGrid::DeploymentException` — Raised if the object can't be removed. This might happen if the object was added with a deployment descriptor.

["nonmutating"] `IceGrid::ObjectInfo` `getObjectInfo(Ice::Identity id)` throws [IceGrid::ObjectNotRegisteredException](#)

Get the object info for the object with the given identity.

Parameters

`id` — The identity of the object.

Return Value

The object info.

Exceptions

`IceGrid::ObjectNotRegisteredException` — Raised if the object isn't registered with the registry.

["nonmutating"] `IceGrid::ObjectInfoSeq` `getObjectInfosByType(string type)`

Get the object info of all the registered objects with the given type.

Parameters

`type` — The type of the object.

Return Value

The object infos.

["nonmutating"] [IceGrid::ObjectInfoSeq](#) getAllObjectInfos(string expr)

Get the object info of all the registered objects whose stringified identities match the given expression.

Parameters

`expr` — The expression to match against the stringified identities of registered objects. The expression may contain a trailing wildcard (*) character.

Return Value

All the object infos with a stringified identity matching the given expression.

["nonmutating"] [bool](#) pingNode(string name) throws [IceGrid::NodeNotExistException](#)

Ping an IceGrid node to see if it is active.

Parameters

`name` — The node name.

Return Value

true if the node ping succeeded, false otherwise.

Exceptions

[IceGrid::NodeNotExistException](#) — Raised if the node doesn't exist.

["nonmutating"] [IceGrid::LoadInfo](#) getNodeLoad(string name) throws [IceGrid::NodeNotExistException](#), [IceGrid::NodeUnreachableException](#)

Get the load averages of the node.

Parameters

`name` — The node name.

Return Value

The node load information.

Exceptions

[IceGrid::NodeNotExistException](#) — Raised if the node doesn't exist.

[IceGrid::NodeUnreachableException](#) — Raised if the node could not be reached.

["nonmutating"] [IceGrid::NodeInfo](#) getNodeInfo(string name) throws [IceGrid::NodeNotExistException](#), [IceGrid::NodeUnreachableException](#)

Get the node information for the node with the given name.

Parameters

`name` — The node name.

Return Value

The node information.

Exceptions

[IceGrid::NodeNotExistException](#) — Raised if the node doesn't exist.

[IceGrid::NodeUnreachableException](#) — Raised if the node could not be reached.

["nonmutating"] int getNodeProcessorSocketCount(string name) throws [IceGrid::NodeNotExistException](#), [IceGrid::NodeUnreachableException](#)

Get the number of physical processor sockets for the machine running the node with the given name. Note that this method will return 1 on operating systems where this can't be automatically determined and where the IceGrid.Node.ProcessorSocketCount property for the node is not set.

Parameters

`name` — The node name.

Return Value

The number of processor sockets or 1 if the number of sockets can't be determined.

Exceptions

`IceGrid::NodeNotExistException` — Raised if the node doesn't exist.

`IceGrid::NodeUnreachableException` — Raised if the node could not be reached.

void shutdownNode(string name) throws [IceGrid::NodeNotExistException](#), [IceGrid::NodeUnreachableException](#)

Shutdown an IceGrid node.

Parameters

`name` — The node name.

Exceptions

`IceGrid::NodeNotExistException` — Raised if the node doesn't exist.

`IceGrid::NodeUnreachableException` — Raised if the node could not be reached.

["nonmutating"] string getNodeHostname(string name) throws [IceGrid::NodeNotExistException](#), [IceGrid::NodeUnreachableException](#)

Get the hostname of this node.

Parameters

`name` — The node name.

Return Value

The node hostname.

Exceptions

`IceGrid::NodeNotExistException` — Raised if the node doesn't exist.

`IceGrid::NodeUnreachableException` — Raised if the node could not be reached.

["nonmutating"] [Ice::StringSeq](#) getAllNodeNames()

Get all the IceGrid nodes currently registered.

Return Value

The node names.

bool pingRegistry(string name) throws [IceGrid::RegistryNotExistException](#)

Ping an IceGrid registry to see if it is active.

Parameters

`name` — The registry name.

Return Value

true if the registry ping succeeded, false otherwise.

Exceptions

`IceGrid::RegistryNotExistException` — Raised if the registry doesn't exist.

`IceGrid::RegistryInfo getRegistryInfo(string name) throws IceGrid::RegistryNotExistException, IceGrid::RegistryUnreachableException`

Get the registry information for the registry with the given name.

Parameters

`name` — The registry name.

Return Value

The registry information.

Exceptions

`IceGrid::RegistryNotExistException` — Raised if the registry doesn't exist.

`IceGrid::RegistryUnreachableException` — Raised if the registry could not be reached.

`void shutdownRegistry(string name) throws IceGrid::RegistryNotExistException, IceGrid::RegistryUnreachableException`

Shutdown an IceGrid registry.

Parameters

`name` — The registry name.

Exceptions

`IceGrid::RegistryNotExistException` — Raised if the registry doesn't exist.

`IceGrid::RegistryUnreachableException` — Raised if the registry could not be reached.

`Ice::StringSeq getAllRegistryNames()`

Get all the IceGrid registries currently registered.

Return Value

The registry names.

`void shutdown()`

Shut down the IceGrid registry.

`["nonmutating"] Ice::SliceChecksumDict getSliceChecksums()`

Returns the checksums for the IceGrid Slice definitions.

Return Value

A dictionary mapping Slice type ids to their checksums.
