Turaya.TrustedServer

Generated by Doxygen 1.7.6.1

Wed Aug 7 2013 14:16:30

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Class Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 turaya Namespace Reference

Namespaces

- namespace compartment
- namespace domain
- namespace organization
- namespace tcd2
- namespace user

Classes

- class NetworkManagerInvalidInterface
- class NetworkManager

  Client side NetworkManager representation.
- class NetworkManagerProxyWrapper
- class PlatformManagementException
- class PlatformManagementObtainingSerialFailed
- class PlatformManagementObtainingFirmwareVersionFailed
- class Version
- class FirmwareVersion
- class TrustedDesktop

  Represents a concrete TrustedDesktop.

Typedefs

- typedef std::string PlatformID
- typedef sirrix::utils::Singleton < TrustedDesktop > CentralTrustedDesktop

5.1.1 Typedef Documentation

5.1.1.1 typedef sirrix::utils::Singleton<TrustedDesktop> turaya::CentralTrustedDesktop

5.1.1.2 typedef std::string turaya::PlatformID

5.2 turaya::compartment Namespace Reference

Classes

- class CompartmentAdaptor

  Adaptor class to make the CompartmentSrv class accessible over DBus.
- class CompartmentSrvException
- class CompartmentSrvAlreadyExists
- class CompartmentSrvNotFound
- class CompartmentSrvHasNoDomamin
- class CompartmentSrvInvalidCompartmentData
- class CompartmentSrvWrongState
- class CompartmentSrvMissingDomain
- class CompartmentSrvVDIException
- class CompartmentSrvVDIHashMismatch
- class CompartmentManagerAdaptor

  Adaptor class to make the CompartmentManagerSrv class accessible over DBus.
- class CompartmentManagerSrv

  Server side CompartmentManager representation.
- class CompartmentSrv

  Server side compartment representation.
- class FTPException
- class FTPAccess
- class SambaException
- class SambaAccess
- class VMOptionParser

Typedefs

- typedef ::__installedCompartmentInfo InstalledCompartmentInfo

5.2.1 Typedef Documentation

5.2.1.1 typedef ::__installedCompartmentInfo turaya::compartment::InstalledCompartmentInfo

5.3 turaya::domain Namespace Reference

Classes

- class DomainAdaptor

  Adaptor class to make the DomainSrv class accessible over DBus.
- class DomainSrvException
- class DomainSrvAlreadyExists
- class DomainSrvNotFound
- class DomainSrvHasNoDomamin
- class DomainSrvInvalidDomainData
- class DomainSrvWrongState
- class DomainSrvCouldNotCreateDomainEncryptionOverlay
- class DomainSrvCouldNotRemoveDomainEncryptionOverlay
- class DomainManagerAdaptor

  Adaptor class to make the DomainManagerSrv class accessible over DBus.
- class DomainManagerSrv

  Server side DomainManager representation.
- class DomainSrv

  Server side domain representation.

Typedefs

- typedef ::__installedDomainInfo InstalledDomainInfo

5.3.1 Typedef Documentation

5.3.1.1 typedef ::__installedDomainInfo turaya::domain::InstalledDomainInfo

5.4 turaya::organization Namespace Reference

Classes

- class VDIDownloader
- class CompartmentManagerObserver
- class OrganizationAdaptor
- class OrganizationSrvException
- class OrganizationSrvAlreadyExists
- class OrganizationSrvNotFound
- class OrganizationCompartmentDataSrvNotFound
- class OrganizationSrvDomaindDataNotFound
- class OrganizationSrvInvalidOrganizationData
- class OrganizationSrvNoTOMConnection
- class OrganizationSrvTOMError

- class OrganizationSrvTOMTimeout
- class OrganizationManagerAdaptor

  Adaptor class to make the OrganizationManagerSrv class accessible over DBus.
- class OrganizationManagerSrv

  Server side OrganizationManager representation.
- class OrganizationSrv

  Server side organization representation.
- class TOM

  Server side TOM representation This Class encapsulates the CommunicationChannel (TrustedChannel) to the Trusted Object Manager server.

Typedefs

- typedef ::__ShareInfo ShareInfo

  Adaptor class to make the OrganizationSrv class accessible over DBus.
- typedef ::__installedOrganizationInfo InstalledOrganizationInfo

5.4.1 Typedef Documentation

5.4.1.1 typedef ::__installedOrganizationInfo turaya::organization::InstalledOrganizationInfo

5.4.1.2 typedef ::__ShareInfo turaya::organization::ShareInfo

Adaptor class to make the OrganizationSrv class accessible over DBus.

5.5 turaya::tcd2 Namespace Reference

Classes

- class PluginNotFoundException
- class TCDJobFactory
- class TCDPluginFactory
- class TCDPluginJob
- class TCDRootJob
- class TCReadThread
- class TCWriteThread
- class TrustedChannelDaemon2

Typedefs

- typedef std::map< string, TCDJobFactory * > ModuleList
- typedef std::map< string, TCDPluginFactory * > PluginList

5.5.1 Typedef Documentation

5.5.1.1 typedef std::map<string,TCDJobFactory*> turaya::tcd2::ModuleList

5.5.1.2 typedef std::map<string,TCDPluginFactory*> turaya::tcd2::PluginList

5.6 turaya::user Namespace Reference

Classes

- class UserAdaptor

  Adaptor class to make the UserSrv class accessible over DBus.
- class UserSrvException
- class UserSrvAlreadyExists
- class UserSrvNotFound
- class UserSrvInvalidUserData
- class UserSrvWrongState
- class UserHypervisor

  Abstraction of platform specific user functions.
- class UserManagerAdaptor

  Adaptor class to make the UserManagerSrv class accessible over DBus.
- class UserManagerSrv

  Server side UserManager representation.
- class UserSrv

  Server side user representation.

Typedefs

- typedef ::__installedUserInfo InstalledUserInfo

5.6.1 Typedef Documentation

5.6.1.1 typedef ::__installedUserInfo turaya::user::InstalledUserInfo

5.7 unittests Namespace Reference

Namespaces

- namespace CompartmentManagement_Test
- namespace OrganizationManagement_Test
- namespace PlatformManagement_Test
- namespace VMOptionParser_Tests

5.8 unittests::CompartmentManagement_Test Namespace Reference

Classes

- class CompartmentManagerObserver
- class CompartmentObserver

Functions

- TEST_CASE (CompartmentManagement_Tests)
- TEST_CASE (Compartment_Tests)

5.8.1 Function Documentation

5.8.1.1 unittests::CompartmentManagement_Test::TEST_CASE (CompartmentManagement_Tests)

create observer and connect signals

```
                                                    {
                    DEBUG_ENABLE();
                    dbuspp::dbus dbus;

                    //create CompartmentData
                    CompartmentID compartmentId = 247;
                    string name = "Private-C";
                    stringstream sstr;
                    sstr << "
0xE3B0C44298FC1C149AFBF4C8996FB92427AE41E4649B934CA495991B7852B855";
                    UInt64 imageSize = 123456789;
                    ByteVector hashHex;
                    sstr >> hashHex;
                    CompartmentVersion version = 1;
                    string date = "16.09.2009 10:24";
                    string comment = "WindowsXP";
                    string description = "A test Compartment";
                    DomainID domainID = 1;
                    OrganizationID organizationID = 42;

                    CompartmentData data(compartmentId, name, hashHex,
imageSize, version, date, comment, description, domainID, organizationID);


                    //start the compartment management service
                    const std::string COMPARTMENT_SERVICE_NAME("
session://turaya.compartment.manager");
                    const std::string COMPARTMENT_MANAGER_PATH("
/compartmentmanagement");
                    dbuspp::service_sptr spService = dbus.server().service(
COMPARTMENT_SERVICE_NAME);
                    CompartmentManagerAdaptor server(
COMPARTMENT_MANAGER_PATH, spService);
                    server.connect(spService);

                    //create the CompartmentManager dbus proxy object
```

```
                CompartmentManager compartmentManager(dbus);

                CompartmentManagerObserver observer;
                compartmentManager.signalCompartmentInstalled.Connect(&
observer, &CompartmentManagerObserver::onCompartmentInstalled);
                compartmentManager.signalCompartmentRemoved.Connect(&
observer, &CompartmentManagerObserver::onCompartmentRemoved);

                // ensure that the compartment to install in not
 already installed
                if (compartmentManager.hasCompartment(compartmentId)){
                        observer.prepareWaitingForEvent();
                        compartmentManager.getCompartment(compartmentId
).remove();
                        observer.waitForEvent();
                }
                ASSERT_THROW(compartmentManager.getCompartment(
compartmentId), turaya::CompartmentNotFound);

                //get number of installed compartments
                size_t compartmentCountBeforeInstall =
compartmentManager.getAllCompartments().size();

                //Install new compartment
                observer.prepareWaitingForEvent();
                compartmentManager.installCompartment(data);
                observer.waitForEvent();

                //ensure that the Compartment was installed
 successfully
                size_t compartmentCountAfterInstall =
compartmentManager.getAllCompartments().size();
                ASSERT(compartmentManager.hasCompartment(compartmentId)
 == true);
                ASSERT(compartmentCountBeforeInstall + 1 ==
compartmentCountAfterInstall);

                //anew installation should throw an exception
                ASSERT_THROW(compartmentManager.installCompartment(data
), CompartmentAlreadyExists);

                //remove the installed compartment
                Compartment c = compartmentManager.getCompartment(
compartmentId);
                observer.prepareWaitingForEvent();
                c.remove();
                observer.waitForEvent();
                //debugFFL << "1" << endl;
                //ensure that the Compartment was removed successfully
                ASSERT(compartmentManager.hasCompartment(compartmentId)
 == false);
                //debugFFL << "2" << endl;
                size_t compartmentCountAfterRemove = compartmentManager
.getAllCompartments().size();
                //debugFFL << "3" << endl;
                ASSERT(compartmentCountBeforeInstall ==
compartmentCountAfterRemove);
                //debugFFL << "4" << endl;
                ASSERT_THROW(compartmentManager.getCompartment(
compartmentId), turaya::CompartmentNotFound);
                //debugFFL << "5" << endl;
                //anew remove should throw an exception
```

```
                    ASSERT_THROW(c.remove(), CompartmentRemoved);
                    //debugFFL << "6" << endl;
                    //using invalid CompartmentData should throw an
 exception
                    //CompartmentData invalidData("this is not a
 json-Format"); //todo this causes a Segmentation fault
                    //
ASSERT_THROW(compartmentManager.installCompartment(invalidData), turaya::CompartmentInv
          }
```

5.8.1.2 unittests::CompartmentManagement_Test::TEST_CASE (Compartment_Tests)

create observers

```
                                    {
                    DEBUG_ENABLE();
                    dbuspp::dbus dbus;

                    //create CompartmentData
                    CompartmentID id = 247;
                    string name = "Private-C";
                    stringstream sstr;
                    sstr << "
0xE3B0C44298FC1C149AFBF4C8996FB92427AE41E4649B934CA495991B7852B855";
                    UInt64 imageSize = 123456789;
                    ByteVector hashHex;
                    sstr >> hashHex;
                    CompartmentVersion version = 1;
                    string date = "16.09.2009 10:24";
                    string comment = "WindowsXP";
                    string description = "A test Compartment";
                    DomainID domain = 1;
                    OrganizationID organizationID = 42;

                    CompartmentData data(id, name, hashHex, imageSize,
version, date, comment, description, domain, organizationID);

                    //start the compartment management service
                    const std::string COMPARTMENT_SERVICE_NAME("
session://turaya.compartment.manager");
                    const std::string COMPARTMENT_MANAGER_PATH("
/compartmentmanagement");
                    dbuspp::service_sptr spService = dbus.server().service(
COMPARTMENT_SERVICE_NAME);
                    CompartmentManagerAdaptor server(
COMPARTMENT_MANAGER_PATH, spService);
                    server.connect(spService);

                    //create the CompartmentManager dbus proxy object
                    CompartmentManager compartmentManager(dbus);

                    CompartmentManagerObserver managerObserver;
                    CompartmentObserver compartmentObserver;

                    //connect signals to CompartmentManagerObserver
                    compartmentManager.signalCompartmentInstalled.Connect(&
managerObserver, &CompartmentManagerObserver::onCompartmentInstalled);
                    compartmentManager.signalCompartmentRemoved.Connect(&
managerObserver, &CompartmentManagerObserver::onCompartmentRemoved);
```

```
                    // ensure that the compartment to install in not
 already installed
                    ASSERT(compartmentManager.hasCompartment(id) == false);

                    //Install new compartment
                    managerObserver.prepareWaitingForEvent();
                    compartmentManager.installCompartment(data);
                    managerObserver.waitForEvent();

                    Compartment compartment = compartmentManager.
getCompartment(id);

                    //connect signals to CompartmentObserver
                    compartment.signalStatusChanged.Connect(&
compartmentObserver, &CompartmentObserver::onStatusChanged);

                    //test all getter methods
                    ASSERT(compartment.getDate().compare(date) == 0);
                    ASSERT(compartment.getComment().compare(comment) == 0);
                    ASSERT(compartment.getID() == id);
                    ASSERT(compartment.getVersion() == version);
                    ASSERT(compartment.getName().compare(name) == 0);

                    ASSERT(compartment.getStatus() == Compartment::stopped)
;
                    ASSERT_THROW(compartment.getTaskID(),
CompartmentWrongState)

                    //start the Compartment
                    compartmentObserver.prepareWaitingForEvent();
                    compartment.start();
                    compartmentObserver.waitForEvent();

                    //ensure that the compartment is running
                    ASSERT(compartment.getStatus() == Compartment::running)
;

                    //anew start should throw an exception
                    ASSERT_THROW(compartment.start(), CompartmentWrongState
);

                    //removing a running compartment should throw an
 exception
                    ASSERT_THROW(compartment.remove(),
CompartmentWrongState);

                    //obtain the taskID
                    TaskID taskID = compartment.getTaskID();

                    Compartment theSameCompartment = compartmentManager.
getCompartment(taskID);

                    ASSERT(compartment.getID() == theSameCompartment.getID(
));
                    //ASSERT(compartment.getID() ==
 theSameCompartment.getID());

                    //stop the Compartment
                    compartmentObserver.prepareWaitingForEvent();
                    compartment.stop();
                    compartmentObserver.waitForEvent();
```

```
                    //ensure that the compartment is stopped
                    ASSERT(compartment.getStatus() == Compartment::stopped)
;

                    //The task id should be invalid now and
 getCompartmentByTaskID should throw an exception
                    ASSERT_THROW(compartmentManager.getCompartment(taskID),
 CompartmentNotFound);

                    //anew stopp should throw an exception
                    ASSERT_THROW(compartment.stop(), CompartmentWrongState)
;

                    //remove the Compartment
                    compartmentObserver.prepareWaitingForEvent();
                    compartment.remove();
                    compartmentObserver.waitForEvent();

                    //ensure that the compartment was removed
                    ASSERT(compartment.getStatus() == Compartment::removed)
;

                    //all method calls (except getStatus) on a removed
 compartment should throw an exception
                    ASSERT_THROW(compartment.getDate(), CompartmentRemoved)
;
                    ASSERT_THROW(compartment.getComment(),
CompartmentRemoved);
                    ASSERT_THROW(compartment.getID(), CompartmentRemoved);
                    ASSERT_THROW(compartment.getName(), CompartmentRemoved)
;
                    ASSERT_THROW(compartment.getVersion(),
CompartmentRemoved);
                    //ASSERT_THROW(compartment->getDomain(),
 CompartmentRemoved); //todo not jet implemented
                    ASSERT_THROW(compartment.getTaskID(),
CompartmentRemoved);
                    ASSERT_THROW(compartment.start(), CompartmentRemoved);
                    ASSERT_THROW(compartment.stop(), CompartmentRemoved);
                    ASSERT_THROW(compartment.remove(), CompartmentRemoved);

          }
```

5.9 unittests::OrganizationManagement_Test Namespace Reference

Classes

- class OrganizationManagerObserver
- class OrganizationObserver

5.10 unittests::PlatformManagement_Test Namespace Reference

Functions

- TEST_CASE (TrustedDesktop_getPlatformID)
- TEST_CASE (TrustedDesktop_getFirmwareVersion)

5.10.1 Function Documentation

5.10.1.1 unittests::PlatformManagement_Test::TEST_CASE (TrustedDesktop_getPlatformID)

```
                                                {
                    TrustedServer& td = CentralTrustedServer::getInstance
    ();
                    PlatformID pltID = td.getPlatformID();

                    ASSERT( pltID.size() > 0 );
            }
```

5.10.1.2 unittests::PlatformManagement_Test::TEST_CASE (TrustedDesktop_getFirmwareVersion)

```
                                                    {
                    TrustedServer& td = CentralTrustedServer::getInstance
    ();
                    FirmwareVersion version = td.getFirmwareVersion();

                    ASSERT( version > FirmwareVersion( 0, 9, 0, 0 ) );
            }
```

5.11 unittests::VMOptionParser_Tests Namespace Reference

Functions

- TEST_CASE (EmptyOption_Test)
- TEST_CASE (SomeOptions_Test)
- TEST_CASE (AllOptions_TEST)
- TEST_CASE (wrongFormat_TEST)
- TEST_CASE (outOfRange_TEST)

5.11.1 Function Documentation

5.11.1.1 unittests::VMOptionParser_Tests::TEST_CASE (EmptyOption_Test)

```
                                {
                VMOptionParser p("");
                UInt32 mem = 0;
                UInt32 vmem = 0;
                string audioIF;
```

```
                    string audioHW;
                    string nicHW;
                    string smbBackup;
                    bool pae = false;
                    bool smartCard = true;
                    bool usbPlainAccess = true;
                    bool cdPassthru = true;
                    bool amd64 = true;

                    p.parseVMOptions(mem, vmem, audioIF, audioHW, nicHW,
    pae, smartCard, usbPlainAccess, cdPassthru, smbBackup, amd64);
                    ASSERT(mem == 512);
                    ASSERT(vmem == 32);
                    ASSERT(audioIF == "pulse");
                    ASSERT(audioHW == "ac97");
                    ASSERT(nicHW == "Am79C973");
                    ASSERT(pae == true);
                    ASSERT(smartCard == false);
                    ASSERT(usbPlainAccess == false);
                    ASSERT(cdPassthru == false);
                    ASSERT(smbBackup == "none");
                    ASSERT(amd64 == false);
            }
```

5.11.1.2 unittests::VMOptionParser_Tests::TEST_CASE (SomeOptions_Test)

```
                                    {
                    VMOptionParser p("--ram 1024 --audioHW hda --smartCard
     on");
                    UInt32 mem = 0;
                    UInt32 vmem = 0;
                    string audioIF;
                    string audioHW;
                    string nicHW;
                    string smbBackup;
                    bool pae = false;
                    bool smartCard = true;
                    bool usbPlainAccess = true;
                    bool cdPassthru = true;
                    bool amd64 = true;

                    p.parseVMOptions(mem, vmem, audioIF, audioHW, nicHW,
    pae, smartCard, usbPlainAccess, cdPassthru, smbBackup, amd64);
                    ASSERT(mem == 1024);
                    ASSERT(vmem == 32);
                    ASSERT(audioIF == "pulse");
                    ASSERT(audioHW == "hda");
                    ASSERT(nicHW == "Am79C973");
                    ASSERT(pae == true);
                    ASSERT(smartCard == true);
                    ASSERT(usbPlainAccess == false);
                    ASSERT(cdPassthru == false);
                    ASSERT(smbBackup == "none");
                    ASSERT(amd64 == false);
            }
```

5.11.1.3 unittests::VMOptionParser_Tests::TEST_CASE (AllOptions_TEST)

```
                                    {
                    VMOptionParser p("--ram 1024 --vram 64 --audioIF alsa
 --audioHW hda --nicHW 82540EM --pae off --smartCard on --usbPlainAccess on
 --cdPassthru on -m server\\share -d on");
                    UInt32 mem = 0;
                    UInt32 vmem = 0;
                    string audioIF;
                    string audioHW;
                    string nicHW;
                    string smbBackup;
                    bool pae = false;
                    bool smartCard = true;
                    bool usbPlainAccess = false;
                    bool cdPassthru = false;
                    bool amd64 = false;

                    p.parseVMOptions(mem, vmem, audioIF, audioHW, nicHW,
pae, smartCard, usbPlainAccess, cdPassthru, smbBackup, amd64);
                    ASSERT(mem == 1024);
                    ASSERT(vmem == 64);
                    ASSERT(audioIF == "alsa");
                    ASSERT(audioHW == "hda");
                    ASSERT(nicHW == "82540EM");
                    ASSERT(pae == false);
                    ASSERT(smartCard == true);
                    ASSERT(usbPlainAccess == true);
                    ASSERT(cdPassthru == true);
                    cout << " SMB:" << smbBackup << endl;
                    ASSERT(smbBackup == "server\\share");
                    ASSERT(amd64 == true);
            }
```

5.11.1.4 unittests::VMOptionParser_Tests::TEST_CASE (wrongFormat_TEST)

```
                                    {
                    VMOptionParser p("-foo bar --lirum 5 larum  234 ipsum")
;
                    UInt32 mem = 0;
                    UInt32 vmem = 0;
                    string audioIF;
                    string audioHW;
                    string nicHW;
                    string smbBackup;
                    bool pae = false;
                    bool smartCard = true;
                    bool usbPlainAccess = true;
                    bool cdPassthru = true;
                    bool amd64 = true;

                    p.parseVMOptions(mem, vmem, audioIF, audioHW, nicHW,
pae, smartCard, usbPlainAccess, cdPassthru, smbBackup, amd64);
                    ASSERT(mem == 512);
                    ASSERT(vmem == 32);
                    ASSERT(audioIF == "pulse");
                    ASSERT(audioHW == "ac97");
                    ASSERT(nicHW == "Am79C973");
                    ASSERT(pae == true);
                    ASSERT(smartCard == false);
```

```
                    ASSERT(usbPlainAccess == false);
                    ASSERT(cdPassthru == false);
                    ASSERT(smbBackup == "none");
                    ASSERT(amd64 == false);
                }
```

5.11.1.5 unittests::VMOptionParser_Tests::TEST_CASE (outOfRange_TEST)

```
                                        {
                    VMOptionParser p("--ram 42 --vram 17 --audioIF ear
 --audioHW mouth --nicHW xylophone --pae onoff --smartCard maybe --usbPlainAccess 1
 --cdPassthru yes --adm64 tak");
                    UInt32 mem = 0;
                    UInt32 vmem = 0;
                    string audioIF;
                    string audioHW;
                    string nicHW;
                    string smbBackup;
                    bool pae = false;
                    bool smartCard = true;
                    bool usbPlainAccess = true;
                    bool cdPassthru = true;
                    bool amd64;

                    p.parseVMOptions(mem, vmem, audioIF, audioHW, nicHW,
 pae, smartCard, usbPlainAccess, cdPassthru, smbBackup, amd64);
                    ASSERT(mem == 512);
                    ASSERT(vmem == 32);
                    ASSERT(audioIF == "pulse");
                    ASSERT(audioHW == "ac97");
                    ASSERT(nicHW == "Am79C973");
                    ASSERT(pae == true);
                    ASSERT(smartCard == false);
                    ASSERT(usbPlainAccess == false);
                    ASSERT(cdPassthru == false);
                    ASSERT(smbBackup == "none");
                    ASSERT(amd64 == false);
                }
```

Chapter 6

Class Documentation

6.1 __installedCompartmentInfo Class Reference

`#include <CompartmentManagerAdaptor.hxx>`

Public Member Functions

- __installedCompartmentInfo ()
- virtual ~__installedCompartmentInfo () throw ()

Public Attributes

- UInt32 myCompartmentID
- std::string myCompartmentPath

6.1.1 Constructor & Destructor Documentation

6.1.1.1 __installedCompartmentInfo::__installedCompartmentInfo ( ) [inline]

```
                                    :
                myCompartmentID(0),
                myCompartmentPath(){};
```

6.1.1.2 virtual __installedCompartmentInfo::~__installedCompartmentInfo ( ) throw () [inline, virtual]

```
{};
```

6.1.2 Member Data Documentation

6.1.2.1 UInt32 __installedCompartmentInfo::myCompartmentID

6.1.2.2 std::string __installedCompartmentInfo::myCompartmentPath

6.2 __installedDomainInfo Class Reference

```
#include <DomainManagerAdaptor.hxx>
```

Public Member Functions

- __installedDomainInfo ()
- virtual ~__installedDomainInfo () throw ()

Public Attributes

- UInt32 myDomainID
- std::string myDomainPath

6.2.1 Constructor & Destructor Documentation

6.2.1.1 __installedDomainInfo::__installedDomainInfo ( ) [inline]

```
                                :
myDomainID(0),
myDomainPath(){};
```

6.2.1.2 virtual __installedDomainInfo::~__installedDomainInfo ( ) throw () [inline, virtual]

```
{};
```

6.2.2 Member Data Documentation

6.2.2.1 UInt32 __installedDomainInfo::myDomainID

6.2.2.2 std::string __installedDomainInfo::myDomainPath

6.3 __installedOrganizationInfo Class Reference

```
#include <OrganizationManagerAdaptor.hxx>
```

Public Member Functions

- __installedOrganizationInfo ()
- virtual ~__installedOrganizationInfo () throw ()

Public Attributes

- UInt32 myOrganizationID
- std::string myOrganizationPath

6.3.1 Constructor & Destructor Documentation

6.3.1.1 __installedOrganizationInfo::__installedOrganizationInfo ( ) [inline]

```
                                    :
myOrganizationID(0),
myOrganizationPath(){};
```

6.3.1.2 virtual __installedOrganizationInfo::~__installedOrganizationInfo ( ) throw () [inline, virtual]

```
{};
```

6.3.2 Member Data Documentation

6.3.2.1 UInt32 __installedOrganizationInfo::myOrganizationID

6.3.2.2 std::string __installedOrganizationInfo::myOrganizationPath

6.4 __installedUserInfo Class Reference

```
#include <UserManagerAdaptor.hxx>
```

Public Member Functions

- __installedUserInfo ()
- virtual ~__installedUserInfo () throw ()

Public Attributes

- UInt32 myUserID
- std::string myUserPath

6.4.1 Constructor & Destructor Documentation

6.4.1.1 __installedUserInfo::__installedUserInfo ( ) [inline]

```
                                    :
                                myUserID( 0 ), myUserPath() {
            }
```

6.4.1.2 virtual __installedUserInfo::~__installedUserInfo ( ) throw () [inline, virtual]

```
                                                          {
            }
```

6.4.2 Member Data Documentation

6.4.2.1 UInt32 __installedUserInfo::myUserID

6.4.2.2 std::string __installedUserInfo::myUserPath

6.5 __ShareInfo Class Reference

```
#include <OrganizationAdaptor.hxx>
```

Public Member Functions

- __ShareInfo ()
- virtual ~__ShareInfo () throw ()

Public Attributes

- std::string myName
- std::string myURI
- UInt32 myType
- UInt32 myTypeID

6.5.1 Constructor & Destructor Documentation

6.5.1.1 __ShareInfo::__ShareInfo ( ) [inline]

```
                        :
                    myName(),
                    myURI(),
                    myType(0),
                    myTypeID(){};
```

6.5.1.2 virtual __ShareInfo::~__ShareInfo ( ) throw () [inline, virtual]

```
{};
```

6.5.2 Member Data Documentation

6.5.2.1 std::string __ShareInfo::myName

6.5.2.2 UInt32 __ShareInfo::myType

6.5.2.3 UInt32 __ShareInfo::myTypeID

6.5.2.4 std::string __ShareInfo::myURI

6.6 turaya::tcd2::TCDRootJob::ClientStateThread Class Reference

```
#include <TCDRootJob.hxx>
```

Public Member Functions

- ClientStateThread (TCDRootJob &tcdRootJob)
- void * run ()

6.6.1 Constructor & Destructor Documentation

6.6.1.1 TCDRootJob::ClientStateThread::ClientStateThread (TCDRootJob & *tcdRootJob*)

```
                                                                        :
      myTCDRootJob(tcdRootJob)
{
}
```

6.6.2 Member Function Documentation

6.6.2.1 void * TCDRootJob::ClientStateThread::run ( )

```
{
        try {
                while (1) {
                        Object clientState = myTCDRootJob.prepareClientState();
                        myTCDRootJob.prepareOutgoingMessage( AtomicValue(
      clientState ).getByteVector() );
                        os::System::msleep(2000);
                }
        } catch ( exception &ex ) {
                debugFFL << "Exception in ClientStateThread has been thrown: "
      << ex.what() << endl;
```

```
                myTCDRootJob.fail();
        }

        return 0;
}
```

6.7 turaya::compartment::CompartmentAdaptor Class Reference

Adaptor class to make the CompartmentSrv class accessible over DBus.

`#include <CompartmentAdaptor.hxx>`

Public Types

- typedef sirrix::utils::SharedPointer < CompartmentAdaptor > Pointer

Public Member Functions

- CompartmentAdaptor (CompartmentSrv::Pointer compartment, std::string dBusPath)
- virtual ~CompartmentAdaptor () throw ()
- void update (dbuspp::fixed_array< unsigned char > dBusData)
- void setVirtualDiskImage (std::string filePath)
- std::string getDate () const
- UInt32 getDomainID () const
- std::string getComment () const
- CompartmentID getID () const
- std::string getName () const
- UInt32 getStatus () const
- Int32 getTaskID () const
- UInt32 getVersion () const
- dbuspp::fixed_array< unsigned char > getVDIDigest () const
- void remove ()
- void start ()
- void stop ()
- void exportImage (std::string partition)
- void importImageSMB (std::string smburl)
- void exportImageSMB (std::string smburl)
- void setDownloadProgress (UInt32 downloadProgress)
- void discardState ()
- virtual void slotStatusChanged (Compartment::Status status)
- virtual void slotProgressChanged (UInt32 progress)
- CompartmentSrv::Pointer getCompartment ()

6.7.1 Detailed Description

Adaptor class to make the CompartmentSrv class accessible over DBus.

6.7.2 Member Typedef Documentation

6.7.2.1 typedef sirrix::utils::SharedPointer< CompartmentAdaptor > turaya::compartment::CompartmentAdaptor::Pointer

6.7.3 Constructor & Destructor Documentation

6.7.3.1 DBUSPP_INTERFACE_IMPLEMENT_END_MAP DBUSPP_OBJECT_IMPLEMENT_END_MAP CompartmentAdaptor::CompartmentAdaptor (CompartmentSrv::Pointer *compartment,* std::string *dBusPath*)

```
                    :
        dbuspp::object(dBusPath),
        myCompartment(compartment),
        statusChanged(),
        progressChanged(){
//      debugFFL << "enter" << endl;
        myCompartment->signalStatusChanged.Connect(this, &
      CompartmentAdaptor::slotStatusChanged);
        myCompartment->signalProgressChanged.Connect(this, &
      CompartmentAdaptor::slotProgressChanged);
//      debugFFL << "leave" << endl;
}
```

6.7.3.2 CompartmentAdaptor::~CompartmentAdaptor ( ) throw () [virtual]

```
                                                {
//      debugFFL << "enter" << endl;
        myCompartment->signalStatusChanged.Disconnect(this,&
      CompartmentAdaptor::slotStatusChanged);
        myCompartment->signalProgressChanged.Disconnect(this, &
      CompartmentAdaptor::slotProgressChanged);
//      debugFFL << "leave" << endl;
}
```

6.7.4 Member Function Documentation

6.7.4.1 void CompartmentAdaptor::discardState ( )

```
                                      {
        try{
                myCompartment->discardState();
        }catch (CompartmentSrvWrongState &e){
                DBUSPP_THROW_ERROR_EXCEPTION(ERROR_COMPARTMENT_WRONG_STATE.
      c_str(), e.what());
        }
}
```

6.7.4.2 void CompartmentAdaptor::exportImage (std::string *partition*)

```
                                                        {
        try{
                myCompartment->exportImage(partition);
        } catch (CompartmentSrvWrongState &e){
                DBUSPP_THROW_ERROR_EXCEPTION(ERROR_COMPARTMENT_WRONG_STATE.
      c_str(), e.what());
        } catch( CompartmentSrvException &e ){
                debugFFL << "exporting failed: " << e.what() << endl;
                DBUSPP_THROW_ERROR_EXCEPTION(ERROR_VDI.c_str(), e.what());
        }
}
```

6.7.4.3 void CompartmentAdaptor::exportImageSMB (std::string *smburl*)

```
                                                        {
        try{
                myCompartment->exportImageSMB(smburl);
        } catch (CompartmentSrvWrongState &e){
                DBUSPP_THROW_ERROR_EXCEPTION(ERROR_COMPARTMENT_WRONG_STATE.
      c_str(), e.what());
        } catch( CompartmentSrvException &e ){
                debugFFL << "exporting SMB failed: " << e.what() << endl;
                DBUSPP_THROW_ERROR_EXCEPTION(ERROR_VDI.c_str(), e.what());
        }
}
```

6.7.4.4 string CompartmentAdaptor::getComment ( ) const

```
                                          {
        return myCompartment->getComment();
}
```

6.7.4.5 CompartmentSrv::Pointer turaya::compartment::CompartmentAdaptor-::getCompartment ( ) [inline]

```
{return myCompartment;};
```

6.7.4.6 string CompartmentAdaptor::getDate ( ) const

```
                                         {
        return myCompartment->getDate();
}
```

6.7.4.7 UInt32 CompartmentAdaptor::getDomainID ( ) const

```
                                          {
        try {
                return myCompartment->getDomainID();
        } catch (CompartmentSrvHasNoDomamin &e){
                debugFFL << "getDomain failed: " << e.what() << endl;
                DBUSPP_THROW_ERROR_EXCEPTION(ERROR_COMPARTMENT_HAS_NO_DOMAIN.
      c_str(), e.what());
        }
        return -1; //prevents compiler warning
}
```

6.7.4.8 CompartmentID CompartmentAdaptor::getID ( ) const

```
                                            {
        return myCompartment->getID();
}
```

6.7.4.9 string CompartmentAdaptor::getName ( ) const

```
                                        {
        return myCompartment->getName();
}
```

6.7.4.10 UInt32 CompartmentAdaptor::getStatus ( ) const

```
                                         {
        return myCompartment->getStatus();
}
```

6.7.4.11 Int32 CompartmentAdaptor::getTaskID ( ) const

```
                                        {
        try {
                return myCompartment->getTaskID();
        }catch (CompartmentSrvWrongState &e){
                DBUSPP_THROW_ERROR_EXCEPTION(ERROR_COMPARTMENT_WRONG_STATE.
      c_str(), e.what());
        }
        return 0; //avoids warning: control reaches end of non-void function
}
```

6.7.4.12 dbuspp::fixed_array< unsigned char > CompartmentAdaptor::getVDIDigest ( ) const

```
                                                                        {
        ByteVector digest = myCompartment->getVDIDigest();
```

```
        dbuspp::fixed_array<unsigned char> retValue(digest.toCArray(), digest.
      size());
        return retValue;
}
```

6.7.4.13 UInt32 CompartmentAdaptor::getVersion ( ) const

```
                                              {
        return myCompartment->getVersion();
}
```

6.7.4.14 void CompartmentAdaptor::importImageSMB (std::string *smburl*)

```
                                                        {
        try{
                myCompartment->importImageSMB(smburl);
        } catch (CompartmentSrvWrongState &e){
                DBUSPP_THROW_ERROR_EXCEPTION(ERROR_COMPARTMENT_WRONG_STATE.
      c_str(), e.what());
        } catch( CompartmentSrvException &e ){
                debugFFL << "importing SMB failed: " << e.what() << endl;
                DBUSPP_THROW_ERROR_EXCEPTION(ERROR_VDI.c_str(), e.what());
        }
}
```

6.7.4.15 void CompartmentAdaptor::remove ( )

```
                                {
        try {
                myCompartment->remove();
        }catch (CompartmentSrvWrongState &e){
                DBUSPP_THROW_ERROR_EXCEPTION(ERROR_COMPARTMENT_WRONG_STATE.
      c_str(), e.what());
        }
        //onCompartmentAdaptorRemoved(myCompartment->getID());
}
```

6.7.4.16 void CompartmentAdaptor::setDownloadProgress (UInt32 *downloadProgress*)

```
                                                                        {
        try{
                myCompartment->setDownloadProgress(downloadProgress);
        }catch (CompartmentSrvWrongState &e){
                DBUSPP_THROW_ERROR_EXCEPTION(ERROR_COMPARTMENT_WRONG_STATE.
      c_str(), e.what());
        }
}
```

6.7.4.17 void CompartmentAdaptor::setVirtualDiskImage (std::string *filePath*)

```
                                                                {
        try{
                myCompartment->setVirtualDiskImage(filePath);
        } catch( CompartmentSrvVDIException &e ){
                debugFFL << "setVirtualDiskImage failed: " << e.what() << endl;
                DBUSPP_THROW_ERROR_EXCEPTION(ERROR_VDI.c_str(), e.what());
        } catch ( CompartmentSrvVDIHashMismatch  &e ){
                debugFFL << "setVirtualDiskImage failed: " << e.what() << endl;
                DBUSPP_THROW_ERROR_EXCEPTION(ERROR_VDI_HASH_MISMATCH.c_str(), e
      .what());
        }
}
```

6.7.4.18 void CompartmentAdaptor::slotProgressChanged (UInt32 *progress*) [virtual]

```
                                                              {
        progressChanged(progress);
}
```

6.7.4.19 void CompartmentAdaptor::slotStatusChanged (Compartment::Status *status*) [virtual]

```
                                                                    {
        statusChanged(status);
}
```

6.7.4.20 void CompartmentAdaptor::start ( )

```
                              {
        try{
                myCompartment->start();
        }catch (CompartmentSrvWrongState &e){
                DBUSPP_THROW_ERROR_EXCEPTION(ERROR_COMPARTMENT_WRONG_STATE.
      c_str(), e.what());
        }
}
```

6.7.4.21 void CompartmentAdaptor::stop ( )

```
                              {
        try {
                myCompartment->stop();
        }catch (CompartmentSrvWrongState &e){
                DBUSPP_THROW_ERROR_EXCEPTION(ERROR_COMPARTMENT_WRONG_STATE.
      c_str(), e.what());
        }
}
```

6.7.4.22 void CompartmentAdaptor::update (dbuspp::fixed_array< unsigned char > *dBusData*)

```
                                                                        {
        try {
                ByteVector bv (dBusData.get(), dBusData.size());
                CompartmentData data(bv);
                myCompartment->update(data);
        }catch (CompartmentSrvInvalidCompartmentData &e){
                debugFFL << "updateDomain failed: " << e.what() << endl;
                DBUSPP_THROW_ERROR_EXCEPTION(ERROR_INVALID_COMPARTMENT_DATA.
      c_str(), e.what());
        }
}
```

6.8 turaya::compartment::CompartmentManagerAdaptor Class - Reference

Adaptor class to make the CompartmentManagerSrv class accessible over DBus.

`#include <CompartmentManagerAdaptor.hxx>`

Public Types

- typedef std::map < CompartmentID, CompartmentAdaptor::Pointer > - CompartmentAdaptorPtrs

Public Member Functions

- CompartmentManagerAdaptor (const std::string &objectPath, dbuspp::service_- sptr spService)
- virtual ~CompartmentManagerAdaptor () throw ()
- std::vector< std::string > getAllCompartments ()
- void installCompartment (dbuspp::fixed_array< unsigned char > dBusData)
- void createCompartmentAdaptors ()

6.8.1 Detailed Description

Adaptor class to make the CompartmentManagerSrv class accessible over DBus.

6.8.2 Member Typedef Documentation

6.8.2.1 typedef std::map< CompartmentID, CompartmentAdaptor::Pointer > turaya::compartment::CompartmentManagerAdaptor::Compartment- AdaptorPtrs

6.8.3 Constructor & Destructor Documentation

6.8.3.1 DBUSPP_INTERFACE_IMPLEMENT_END_MAP DBUSPP_OBJECT_IMPLEMENT_END_MAP CompartmentManagerAdaptor::CompartmentManagerAdaptor (const std::string & *objectPath*, dbuspp::service_sptr *spService*)

```
                                    :
        dbuspp::object(objectPath),
        myCompartmentAdaptors(),
        myCompartmentManagerSrv(),
        myService(spService),
        compartmentInstalled(),
        compartmentRemoved(){
//      debugFFL << "enter" << endl;
        myCompartmentManagerSrv.signalCompartmentRemoving.Connect(this, &
      CompartmentManagerAdaptor::slotCompartmentRemoving);
        myCompartmentManagerSrv.signalCompartmentInstalled.Connect(this, &
      CompartmentManagerAdaptor::slotCompartmentInstalled);
        createCompartmentAdaptors();
//      debugFFL << "leave" << endl;
}
```

6.8.3.2 CompartmentManagerAdaptor::~CompartmentManagerAdaptor ( ) throw () [virtual]

```
                                                                    {
//      debugFFL << "enter" << endl;
        myCompartmentManagerSrv.signalCompartmentRemoving.Disconnect(this, &
      CompartmentManagerAdaptor::slotCompartmentRemoving);
        myCompartmentManagerSrv.signalCompartmentInstalled.Disconnect(this, &
      CompartmentManagerAdaptor::slotCompartmentInstalled);
        myCompartmentAdaptors.erase(myCompartmentAdaptors.begin(),
      myCompartmentAdaptors.end());
//      debugFFL << "leave" << endl;
}
```

6.8.4 Member Function Documentation

6.8.4.1 void CompartmentManagerAdaptor::createCompartmentAdaptors ( )

```
                                                          {
        CompartmentManagerSrv::CompartmentSrvPtrs compartments =
      myCompartmentManagerSrv.getAllCompartments();
        CompartmentManagerSrv::CompartmentSrvPtrs::iterator i;
        for(i = compartments.begin(); i != compartments.end(); i++){
                string path = createCompartmentPath(i->second->getID());
                debugFFL << "Create CompartmentAdaptor for Compartment with ID"
                                << i->second->getID() << "at path " << path <<
      endl;

                CompartmentAdaptor::Pointer compAdapt(new CompartmentAdaptor(i
      ->second, path));
                compAdapt->connect(myService);
                myCompartmentAdaptors.insert(pair<CompartmentID,
       CompartmentAdaptor::Pointer>(i->second->getID(), compAdapt));
```

```
        }
}
```

6.8.4.2 std::vector< std::string > CompartmentManagerAdaptor::getAll-Compartments ( )

```
                                                                        {

  CompartmentAdaptorPtrs::iterator i;
  std::vector< std::string > paths;
  for(i = myCompartmentAdaptors.begin(); i != myCompartmentAdaptors.end()
; i++){
          paths.push_back((*i).second->path());
  }
  return paths;
}
```

6.8.4.3 void CompartmentManagerAdaptor::installCompartment (dbuspp::fixed_array< unsigned char > *dBusData*)

```
            {

  try {
          ByteVector bv(dBusData.get(),dBusData.size());
          CompartmentData data = CompartmentData(bv);
          myCompartmentManagerSrv.installCompartment(data);
  } catch (CompartmentSrvAlreadyExists &e){
          debugFFL << "installCompartment failed: " << e.what() << endl;
          DBUSPP_THROW_ERROR_EXCEPTION(ERROR_COMPARTMENT_ALREADY_EXISTS.
c_str(), e.what());
  } catch (CompartmentSrvInvalidCompartmentData &e){
          debugFFL << "installCompartment failed: " << e.what() << endl;
          DBUSPP_THROW_ERROR_EXCEPTION(ERROR_INVALID_COMPARTMENT_DATA.
c_str(), e.what());
  } catch (CompartmentSrvMissingDomain &e){
          debugFFL << "installCompartment failed: " << e.what() << endl;
          DBUSPP_THROW_ERROR_EXCEPTION(ERROR_COMPARTMENT_MISSING_DOMAIN.
c_str(), e.what());
  } catch (CompartmentSrvException &e){
          debugFFL << "installCompartment failed: " << e.what() << endl;
          DBUSPP_THROW_ERROR_EXCEPTION(COMPARTMENT_GENERAL_ERROR.c_str(),
 e.what());
  }
}
```

6.9 turaya::organization::CompartmentManagerObserver Class - Reference

`#include <CompartmentManagerObserver.hxx>`

Public Member Functions

- CompartmentManagerObserver (OrganizationSrv &org)
- virtual ~CompartmentManagerObserver ()
- void * run ()

6.9.1 Constructor & Destructor Documentation

6.9.1.1 CompartmentManagerObserver::CompartmentManagerObserver (OrganizationSrv & *org*)

```
                                                                        :
  myCompartmentManager(0),
  myCompartments(),
  myDownloads(),
  myLockingMutex(),
  myOrganization(org){
  start();

}
```

6.9.1.2 CompartmentManagerObserver::~CompartmentManagerObserver ( ) [virtual]

```
                                                    {
  ScopedMutex scopeLock(myLockingMutex);
  while( myDownloads.begin() != myDownloads.end() ){
          VDIDownloader* downloader = myDownloads.begin()->second;
          myDownloads.erase( myDownloads.begin());
          downloader->stop();
          debugFFL << "Waiting for download thread to terminate..." <<
endl;
          downloader->join();
          debugFFL << "Download thread terminated..." << endl;
          delete downloader;
  }
}
```

6.9.2 Member Function Documentation

6.9.2.1 void * CompartmentManagerObserver::run ( )

```
                                  {
  //first we go to sleep for ten seconds
  sleep(10);
  try{
          myCompartmentManager = &CompartmentManager::getInstance(
CentralTrustedServer::getInstance());
          myCompartments = myCompartmentManager->getAllCompartments();

          for (CompartmentManager::Compartments::iterator comp_it =
myCompartments.begin();
                          comp_it != myCompartments.end(); comp_it++){
```

```
                    debugFFL << "Connecting status-changed-signal of
 Compartment" << comp_it->first << endl;
                    comp_it->second.signalStatusChanged.Connect(this, &
CompartmentManagerObserver::onCompartmentStatusChanged);
                    if (comp_it->second.getStatus() ==
Compartment::downloadingImage) {
                            CompartmentID id = comp_it->second.getID();
                            VDIDownloader* downloader = new VDIDownloader(
myOrganization, id);
                            if (myDownloads.insert(pair<CompartmentID,
 VDIDownloader*>(id, downloader)).second){
                                    downloader->start();
                            }else{
                                    debugFFL << "Downloader for compartment
 " << id << "already exists!!!" << endl;
                                    delete downloader;
                            }
                    }
            }
            myCompartmentManager->signalCompartmentInstalled.Connect(this,
&CompartmentManagerObserver::onNewCompartment);
            myCompartmentManager->signalCompartmentRemoved.Connect(this, &
CompartmentManagerObserver::onCompartmentRemoved);
          }catch (CompartmentException &e){
            debugFFL << "Exception during initialization of
 CompartmentManagerObserver: " << e.what() << endl;

          }
          return 0;
}
```

6.10 unittests::CompartmentManagement_Test::CompartmentManagerObserver Class Reference

Public Member Functions

- void onCompartmentRemoved (CompartmentID iD)
- void onCompartmentInstalled (CompartmentID iD)
- void prepareWaitingForEvent ()
- void waitForEvent ()

6.10.1 Member Function Documentation

6.10.1.1 void unittests::CompartmentManagement_Test::CompartmentManagerObserver::onCompartmentInstalled (CompartmentID *iD*) [inline]

```
                                                                    {
                            debugFFL << "enter" << endl;
                            myEvent = true;
                    }
```

6.10.1.2 void unittests::CompartmentManagement_Test::CompartmentManagerObserver::onCompartmentRemoved (CompartmentID *iD*) [inline]

```
                                                                {
                        debugFFL << "enter" << endl;
                        myEvent = true;
                }
```

6.10.1.3 void unittests::CompartmentManagement_Test::CompartmentManagerObserver::prepareWaitingForEvent ( ) [inline]

```
                                                {
                        debugFFL << "enter" << endl;
                        myEvent = false;
                }
```

6.10.1.4 void unittests::CompartmentManagement_Test::CompartmentManagerObserver::waitForEvent ( ) [inline]

```
                                  {
                        debugFFL << "enter" << endl;
                        while (!myEvent){
                                //sleep(1);
                        }
                        debugFFL << "leave" << endl;
                }
```

6.11 turaya::compartment::CompartmentManagerSrv Class Reference

Server side CompartmentManager representation.

`#include <CompartmentManagerSrv.hxx>`

Public Types

- typedef std::map < CompartmentID, CompartmentSrv::Pointer > CompartmentSrvPtrs

Public Member Functions

- CompartmentManagerSrv ()
- virtual ~CompartmentManagerSrv () throw ()

- CompartmentSrvPtrs getAllCompartments ()

  Returns all installed Compartments.

- void installCompartment (CompartmentData compartmentData)

  Install a new Compartment.

Public Attributes

- sirrix::utils::Signal1 < CompartmentSrv::Pointer > signalCompartmentInstalled

  Emitted when a new Compartment is installed.

- sirrix::utils::Signal1 < turaya::CompartmentID > signalCompartmentRemoving

  Emitted on removement of a Compartment.

6.11.1 Detailed Description

Server side CompartmentManager representation.

6.11.2 Member Typedef Documentation

6.11.2.1 typedef std::map<CompartmentID, CompartmentSrv::Pointer > turaya::compartment::CompartmentManagerSrv::CompartmentSrvPtrs

6.11.3 Constructor & Destructor Documentation

6.11.3.1 CompartmentManagerSrv::CompartmentManagerSrv ( )

```
                                                :
signalCompartmentInstalled(),
signalCompartmentRemoving(),
myCompartments(){
//      debugFFL << "enter" << endl;
        init(); //adopted from old libCompartment
        loadCompartments();
//      debugFFL << "leave" << endl;
}
```

6.11.3.2 CompartmentManagerSrv::~CompartmentManagerSrv ( ) throw () [virtual]

```
                                                    {
//      debugFFL << "enter" << endl;
        myCompartments.erase(myCompartments.begin(), myCompartments.end());
//      debugFFL << "leave" << endl;
}
```

6.11.4 Member Function Documentation

6.11.4.1 CompartmentManagerSrv::CompartmentSrvPtrs CompartmentManagerSrv::getAllCompartments ( )

Returns all installed Compartments.

Returns

All Compartments.

```
{
  return myCompartments;
}
```

6.11.4.2 void CompartmentManagerSrv::installCompartment (CompartmentData *compartmentData*)

Install a new Compartment.

Parameters

compartmentData	CompartmentData describing the Compartment to install

Exceptions

CompartmentSrvAlreadyExists	if the Compartment already exists
CompartmentSrvInvalidCompartmentData	if the given CompartmentData is invalid
CompartmentSrvMissingDomain	if the Compartment requires a not installed Domain.

```
{
  debugFFL << "Installing new Compartment with ID " << compartmentData.
getID() << endl;
  CompartmentSrv::Pointer newCompartment = internal_install(
compartmentData);
  signalCompartmentInstalled(newCompartment);
  newCompartment->fireStatusChanged();
}
```

6.11.5 Member Data Documentation

6.11.5.1 sirrix::utils::Signal1<CompartmentSrv::Pointer> turaya::compartment::-CompartmentManagerSrv::signalCompartmentInstalled

Emitted when a new Compartment is installed.

6.11.5.2 sirrix::utils::Signal1<turaya::CompartmentID> turaya::compartment::-CompartmentManagerSrv::signalCompartmentRemoving

Emitted on removement of a Compartment.

6.12 unittests::CompartmentManagement_Test::Compartment-Observer Class Reference

Public Member Functions

- void onStatusChanged (Compartment &compartment, Compartment::Status status)
- void prepareWaitingForEvent ()
- void waitForEvent ()

6.12.1 Member Function Documentation

6.12.1.1 void unittests::CompartmentManagement_Test::CompartmentObserver-::onStatusChanged (Compartment & *compartment,* Compartment::Status *status*) [inline]

```
{
        debugFFL << "enter" << endl;
        myEvent = true;
}
```

6.12.1.2 void unittests::CompartmentManagement_Test::-CompartmentObserver::prepareWaitingForEvent ( ) [inline]

```
{
        debugFFL << "enter" << endl;
        myEvent = false;
}
```

6.12.1.3 void unittests::CompartmentManagement_Test::CompartmentObserver-::waitForEvent ( ) [inline]

```
{
```

```
        debugFFL << "enter" << endl;
        while (!myEvent){
                //sleep(1);
        }
        debugFFL << "leave" << endl;
}
```

6.13 turaya::compartment::CompartmentSrv Class Reference

Server side compartment representation.

`#include <CompartmentSrv.hxx>`

Classes

- class **ExportThread**
- class **ProgressThread**
- class **RunningCompartmentWatchDog**
- class **SMBAccessThread**
- class **StartThread**
- class **StopThread**

Public Types

- typedef sirrix::utils::SharedPointer < CompartmentSrv > Pointer

Public Member Functions

- CompartmentSrv (const CompartmentID &compartmentID)
- virtual ~CompartmentSrv ()
- void update (const CompartmentData &compartmentData)
- void setVirtualDiskImage (sirrix::os::Path filePath)
- DomainID getDomainID () const
- Date getDate () const
- std::string getComment () const
- CompartmentID getID () const
- std::string getName () const
- turaya::Compartment::Status getStatus () const
- CompartmentVersion getVersion () const
- TaskID getTaskID () const

  returns the process id of the process currently executing this compartment
- sirrix::utils::ByteVector getVDIDigest () const

  returns the digest of the associated Compartment image
- void remove ()

  Uninstalls this Compartment.

- void start ()

  Starts this Compartment.
- void stop ()

  Stops this Compartment.
- void discardState ()

  Discards the VM state, will fail if compartment is not currently stopped.
- void exportImage (std::string partition)

  Exports the VDI, will fail if compartment is not in status stopped.
- void exportImageSMB (std::string smburl)

  Exports the VDI to a SMB share, will fail if compartment is not in status stopped.
- void importImageSMB (std::string smburl)

  Imports the VDI from a SMB share, will fail if compartment is not in status stopped.
- void fireStatusChanged ()
- void setDownloadProgress (UInt32 progress)

  Sets the download Progress.

Public Attributes

- sirrix::utils::Signal1 < turaya::Compartment::Status > signalStatusChanged

  Emitted when status of this Compartment has changed.
- sirrix::utils::Signal1< UInt32 > signalProgressChanged

  Emitted when the progress has changed.
- sirrix::utils::Signal1 < turaya::CompartmentID > signalRemoved

  Emitted on removement of this Compartment.

6.13.1 Detailed Description

Server side compartment representation.

6.13.2 Member Typedef Documentation

6.13.2.1 typedef sirrix::utils::SharedPointer< CompartmentSrv > turaya::compartment::CompartmentSrv::Pointer

6.13.3 Constructor & Destructor Documentation

6.13.3.1 CompartmentSrv::CompartmentSrv (const CompartmentID & *compartmentID*)

:

```
signalStatusChanged(),
signalProgressChanged(),
signalRemoved(),
myRunningCompartmentWatchDogThread(*this),
myProgressThread(*this),
myStartThread(*this),
```

```
        myStopThread(*this),
        myExportThread(*this),
        mySMBAccessThread(*this),
        myID(compartmentID),
        myDate(DataBaseUtils::getString(COMPARTMENTTABLE, "date", myID)),
        myDomainID(DataBaseUtils::getInt(COMPARTMENTTABLE, "domainID", myID)),
        myComment(DataBaseUtils::getString(COMPARTMENTTABLE, "comment", myID)),
        myDescription(DataBaseUtils::getString(COMPARTMENTTABLE, "description",
       myID)),
        myName(DataBaseUtils::getString(COMPARTMENTTABLE, "name", myID)),
        myTaskID(obtainTaskID()),
        myStatus((myTaskID!=0)?Compartment::running:Compartment::stopped),
        myVersion(DataBaseUtils::getInt(COMPARTMENTTABLE, "version", myID)),
        myVDIName(DataBaseUtils::getString(COMPARTMENTTABLE, "vdi_name", myID))
      ,
        myVDIHash(),
        mySMBBackupShare(DataBaseUtils::getString(COMPARTMENTTABLE, "
      smb_backup_share", myID)),
        myTempVDIFilePath(),
        myProgress(0),
        myCopyFileSize(0){

        stringstream sstr;
        sstr << DataBaseUtils::getString(COMPARTMENTTABLE, "downloadHash", myID
      );

        sstr >> myVDIHash;

        if (myVDIName.size() == 0){
                tryImportImage();
                if (myStatus != Compartment::importing){
                        debugFFL << "Created Compartment in \"downloadingImage
      \" status" << endl;
                        myStatus = Compartment::downloadingImage;
                }
        }
        myProgressThread.start();
        myRunningCompartmentWatchDogThread.start();

}
```

6.13.3.2 CompartmentSrv::~CompartmentSrv ( ) [virtual]

```
                                {
        debugFFL << "enter" << endl;
        myProgressThread.stop();
        myRunningCompartmentWatchDogThread.stop();
        try {
                myProgressThread.join();
                myRunningCompartmentWatchDogThread.join();
        }catch (ThreadError &e){
                debugFFL << "Exception: " << e.what() << endl;
        }
        debugFFL << "leave" << endl;
}
```

6.13.4 Member Function Documentation

6.13.4.1 void CompartmentSrv::discardState ( )

Discards the VM state, will fail if compartment is not currently stopped.

```
                                {
        int result = 0;
        stringstream sstr;

        //hack until CompartmentManager supports multiple users
        fixPermissions();

        sstr << "sudo su -c \"VBoxManage discardstate " << myName << "\" -s
       /bin/bash " << getActiveUser() << " &";
        //sstr << "VBoxManage discardstate " << myName;
        result = system(sstr.str().c_str());
}
```

6.13.4.2 void CompartmentSrv::exportImage (std::string *partition*)

Exports the VDI, will fail if compartment is not in status stopped.

Parameters

partition	Name of the partition on the external media to be used

```
                                                {
        debugFFL << "enter" << endl;
        if (myStatus != Compartment::stopped) {
                throw CompartmentSrvWrongState("Compartment must be in state
       stopped for export");
        }

        if (partition.empty()) {
                throw CompartmentSrvException("Partition name not given.");
        }

        Path plainPartition(EXT_MEDIA_PLAIN_MOUNTPOINT);
        plainPartition.addSubDir(partition);
        Path ecryptOverlayPath(EXT_MEDIA_ECRYPT_MOUTPOINT_PREFIX);
        Path plainPath(plainPartition);
        Path vdiSource(IMAGE_DIRECTORY);
        vdiSource.addSubDir(myVDIName);
        stringstream sstr;
        sstr << myDomainID;
        string TVDid(sstr.str());
        sstr.clear();
        sstr.str("");
        sstr << myID;
        string compId(sstr.str());

        ecryptOverlayPath.addSubDir(TVDid).addSubDir(EXT_MEDIA_SUB_DIRECTORY).
      addSubDir(partition);

        plainPath.addSubDir(EXPORT_DIRECTORY).addSubDir(TVDid).addSubDir(compId
      );
        Path vdiTarget(ecryptOverlayPath);
        vdiTarget.addSubDir(EXPORT_DIRECTORY).addSubDir(TVDid).addSubDir(compId
```

```
      ).addSubDir(myVDIName);

        debugFFL << "Plain partition: " << plainPartition.getPath() << endl;
        debugFFL << "Plain export path: " << plainPath.getPath() << endl;
        debugFFL << "Overlay path: " << ecryptOverlayPath.getPath() << endl;
        debugFFL << "VDI Source: " << vdiSource.getPath() << endl;
        debugFFL << "VDI Target: " << vdiTarget.getPath() << endl;

        if (!FileManager::isDir(plainPartition)){
                debugFFL << "Path: " << plainPartition.getPath() << " does not
       exist." << endl;
                throw CompartmentSrvException("Partition not Found!");
        }
        if (!FileManager::isDir(ecryptOverlayPath)){
                debugFFL << "Path: " << ecryptOverlayPath.getPath() << " does
       not exist." << endl;
                throw CompartmentSrvException("Encryption overlay not Found!");
        }

        if ( !FileManager::fileExists(vdiSource)){
                debugFFL << "VDI: " << vdiSource.getPath() << "does not exists.
      " << endl;
                throw CompartmentSrvException("Image to export not Found!");
        }

        UInt64 vdiSize = FileManager::getFileSize(vdiSource);
        UInt64 freeSpace = FileManager::getFreeSpace(plainPartition);

        if (vdiSize > freeSpace) {
                debugFFL << "Not enough disk space, needed: " << vdiSize << "
       available: " << freeSpace << endl;
                throw CompartmentSrvException("Not enough disk space!");
        }

        try {
                FileManager::createDirWithParents(plainPath);
        }catch (FileManagerException &e){
                debugFFL << "createDirWithParents: " << plainPath.getPath() <<
      " - " << e.what() << endl;
                throw CompartmentSrvException("Failed to create export
       directory!");
        }

        myStatus = Compartment::exporting;
        signalStatusChanged(myStatus);
        myExportThread.setSource(vdiSource);
        myExportThread.setTarget(vdiTarget);
        myExportThread.start();
}
```

6.13.4.3 void CompartmentSrv::exportImageSMB (std::string *smburl*)

Exports the VDI to a SMB share, will fail if compartment is not in status stopped.

Parameters

the	smb share url witch user credentials e.g.: smb://user-:password/sharename

```
                                                    {
        if (myStatus != Compartment::stopped) {
                throw CompartmentSrvWrongState("Compartment must be in state
       stopped for export");
        }

        if ( mySMBBackupShare == "none" ) {
                throw CompartmentSrvException("SMB URL not given.");
        }

        if (smburl.empty()){
                throw CompartmentSrvException("SMB credentials not given.");
        }

        UserID userID = 0;
        try{
                UserManager& usrMgr = UserManager::getInstance(
      CentralTrustedServer::getInstance());
                userID = usrMgr.getUser(usrMgr.getCurrentUser()).getUserID();
        }catch (UserException &e) {
                debugFFL << "exception thrown: " << e.what() << endl;
                throw CompartmentSrvException("Failed to obtain curren user ID.
      ");
        }

        stringstream sstr;
        sstr << "ftp://" << smburl << "@" << mySMBBackupShare << "/U" << userID
       << "_D" << myDomainID << "_C" << myID << ".vdi";
        string smbPath = sstr.str();

        signalStatusChanged(myStatus = Compartment::exporting);
        Path vdiSource(IMAGE_DIRECTORY);
        vdiSource.addSubDir(myVDIName);
        debugFFL << "Exporting file: " << vdiSource.getPath() << " to " <<
      smbPath << endl;
        mySMBAccessThread.setFile(smbPath, vdiSource);
}
```

6.13.4.4 void turaya::compartment::CompartmentSrv::fireStatusChanged ( ) [inline]

```
{signalStatusChanged (myStatus);}
```

6.13.4.5 string CompartmentSrv::getComment ( ) const

```
                                        {
        return myComment;
}
```

6.13.4.6 Date CompartmentSrv::getDate ( ) const

returns the installation date of this compartment

```
                                    {
        return myDate;
}
```

6.13.4.7 DomainID CompartmentSrv::getDomainID ( ) const

returns the ID of Domains this compartment belongs to

Returns

DomainID

```
                                                {
        return myDomainID;
}
```

6.13.4.8 CompartmentID CompartmentSrv::getID ( ) const

```
                                                {
        return myID;
}
```

6.13.4.9 string CompartmentSrv::getName ( ) const

```
                                        {
        return myName;
}
```

6.13.4.10 turaya::Compartment::Status CompartmentSrv::getStatus ( ) const

```
                                                                {
        return myStatus;
}
```

6.13.4.11 TaskID CompartmentSrv::getTaskID ( ) const

returns the process id of the process currently executing this compartment

Exceptions

CompartmentSrv-WrongState	if this compartment is not currently running

```
                                        {
        if (myStatus == Compartment::running){
```

```
                return myTaskID;
        }
        throw CompartmentSrvWrongState("Compartment must be in state running
       for getTaskID()");
}
```

6.13.4.12 sirrix::utils::ByteVector CompartmentSrv::getVDIDigest ( ) const

returns the digest of the associated Compartment image

```
                                                        {
        return myVDIHash;
}
```

6.13.4.13 CompartmentVersion CompartmentSrv::getVersion ( ) const

```
                                                    {
        return myVersion;
}
```

6.13.4.14 void CompartmentSrv::importImageSMB (std::string *smburl*)

Imports the VDI from a SMB share, will fail if compartment is not in status stopped.

Parameters

the	smb share url witch user credentials e.g.: smb://user-:password/sharename

```
                                                {
        if (myStatus != Compartment::stopped) {
                throw CompartmentSrvWrongState("Compartment must be in state
       stopped for import");
        }

        if ( mySMBBackupShare == "none" ) {
                throw CompartmentSrvException("SMB URL not given.");
        }

        if (smburl.empty()){
                throw CompartmentSrvException("SMB credentials not given.");
        }
        UserID userID = 0;
        try{
                UserManager& usrMgr = UserManager::getInstance(
      CentralTrustedServer::getInstance());
                userID = usrMgr.getUser(usrMgr.getCurrentUser()).getUserID();
        }catch (UserException &e) {
                debugFFL << "exception thrown: " << e.what() << endl;
                throw CompartmentSrvException("Failed to obtain curren user ID.
      ");
        }
```

```
        stringstream sstr;
        sstr << "ftp://" << smburl << "@" << mySMBBackupShare << "/U" << userID
       << "_D" << myDomainID << "_C" << myID << ".vdi";
        string smbPath = sstr.str();

        signalStatusChanged(myStatus = Compartment::importing);
        Path vdiTarget(IMAGE_DIRECTORY);
        vdiTarget.addSubDir(myVDIName);
        debugFFL << "Importing file: " << smbPath << " to: " << vdiTarget.
      getPath() << endl;
        mySMBAccessThread.getFile(smbPath, vdiTarget);
        discardState();
}
```

6.13.4.15 void CompartmentSrv::remove ( )

Uninstalls this Compartment.

Exceptions

CompartmentSrv-WrongState	if this compartment in not currently stopped

```
                          {
        debugFFL << "enter" << endl;
        if (myStatus != Compartment::stopped && myStatus !=
      Compartment::downloadingImage){
                throw CompartmentSrvWrongState("Compartment must be in state
       stopped or downloadingImage for remove()");
        }

        if (myStatus != Compartment::downloadingImage){
                unregisterVirtualMashine();
        }
        myStatus = Compartment::removed;

        //signalStatusChanged(myStatus);
        signalRemoved(myID);
        fixPermissions();
        debugFFL << "leave" << endl;
}
```

6.13.4.16 void CompartmentSrv::setDownloadProgress (UInt32 *progress*)

Sets the download Progress.

```
                                                    {
        if (myStatus == Compartment::downloadingImage){
                myProgress = progress;
                debugFFL << "Signaling Progress: " << myProgress << endl;
                myProgressThread.wakeUp();
        }
}
```

6.13.4.17 void CompartmentSrv::setVirtualDiskImage (sirrix::os::Path *filePath*)

```
                                                        {
        debugFFL << "enter" << endl;
        if (myStatus != Compartment::downloadingImage) {
                throw CompartmentSrvWrongState("Compartment must be in state
      downloadingImage for setVirtualDiskImage");
        }

        myTempVDIFilePath = filePath;
        Thread::start();
        debugFFL << "Installation thread started." << endl;
}
```

6.13.4.18 void CompartmentSrv::start ( )

Starts this Compartment.

Exceptions

CompartmentSrv-WrongState	if this compartment is not currently stopped

```
                        {
        if (myStatus != Compartment::stopped){
                throw CompartmentSrvWrongState("Compartment must be in state
      stopped for start()");
        }
        fixPermissions();

        myStatus = Compartment::starting;
        signalStatusChanged(myStatus);
        myStartThread.start();
}
```

6.13.4.19 void CompartmentSrv::stop ( )

Stops this Compartment.

Exceptions

CompartmentSrv-WrongState	if this compartment is not currently running

```
                        {
        if (myStatus != Compartment::running){
                throw CompartmentSrvWrongState("Compartment must be in state
      running for stop()");
        }
        myStatus = Compartment::stopping;
        signalStatusChanged(myStatus);
        myStopThread.start();
}
```

6.13.4.20 void CompartmentSrv::update (const CompartmentData & *compartmentData*)

```
                                                                    {
 bool statusHasChanged = false;
 if (myID != compartmentData.getID()){
         stringstream sstr;
         sstr << "CompartmentID mismatch. Compartment with ID " << myID
                         << " can not be updated thru CompartmentData
with ID "<< compartmentData.getID();
         debugFFL << sstr.str() << endl;
         throw CompartmentSrvInvalidCompartmentData(sstr.str());
 }

 if (myDate == compartmentData.getLastChange() &&
         myComment == compartmentData.getComment() &&
         myDescription == compartmentData.getDescription() &&
         myName == compartmentData.getName() &&
         myVersion == compartmentData.getVersion() &&
         myVDIHash == compartmentData.getImageHash()&&
         myDomainID == compartmentData.getDomainID()) {
         debugFFL << "Compartment " << myID << " already up-to-date." <<
endl;
         return;
 }

 if (myVDIHash != compartmentData.getImageHash()){
         debugFFL << "Update virtual disk image." << endl;
         unregisterVirtualMashine();
         myStatus = Compartment::downloadingImage;
         statusHasChanged = true;
 }

 if (myDescription != compartmentData.getDescription()){
         myDescription = compartmentData.getDescription();
         configureVMSettings();
 }

 myVDIHash = compartmentData.getImageHash();
 myDate = compartmentData.getLastChange();
 myComment = compartmentData.getComment();
 //myName = compartmentData.getName();
 myVersion = compartmentData.getVersion();
 myDomainID = compartmentData.getDomainID();

 stringstream sstr;
 sstr << "UPDATE " << COMPARTMENTTABLE << " SET name = \""       <<
myName

 << "\" , comment = \"" << myComment

 << "\" , description = \"" << myDescription

 << "\" , downloadHash = \"" << myVDIHash

 << "\" , vdi_name = \"" << myVDIName

 << "\" , date = \""     << myDate

 << "\" , version = "    << myVersion

 << ", domainID = " << myDomainID
```

```
        << " WHERE id = " << myID;

         debugFFL << "Execute: " << sstr.str() << endl;
         DataBaseUtils::sqlExecute(sstr.str()); //TODO comment in
         debugFFL << "leave" << endl;

         if (statusHasChanged) {
                 signalStatusChanged(myStatus);
         }
}
```

6.13.5 Member Data Documentation

6.13.5.1 sirrix::utils::Signal1<UInt32> turaya::compartment::CompartmentSrv::signalProgressChanged

Emitted when the progress has changed.

6.13.5.2 sirrix::utils::Signal1<turaya::CompartmentID> turaya::compartment::CompartmentSrv::signalRemoved

Emitted on removement of this Compartment.

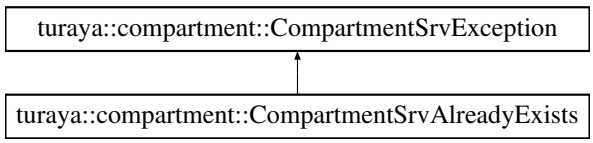
6.13.5.3 sirrix::utils::Signal1<turaya::Compartment::Status> turaya::compartment::CompartmentSrv::signalStatusChanged

Emitted when status of this Compartment has changed.

6.14 turaya::compartment::CompartmentSrvAlreadyExists Class - Reference

`#include <CompartmentExceptionsSrv.hxx>`

Inheritance diagram for turaya::compartment::CompartmentSrvAlreadyExists:



Public Member Functions

- CompartmentSrvAlreadyExists (const std::string &context="CompartmentSrvAlreadyExists")
- virtual ~CompartmentSrvAlreadyExists () throw ()

6.14.1 Constructor & Destructor Documentation

6.14.1.1 turaya::compartment::CompartmentSrvAlreadyExists::-CompartmentSrvAlreadyExists (const std::string & *context* = "CompartmentSrvAlreadyExists") [inline]

```
:
CompartmentSrvException(context){};
```

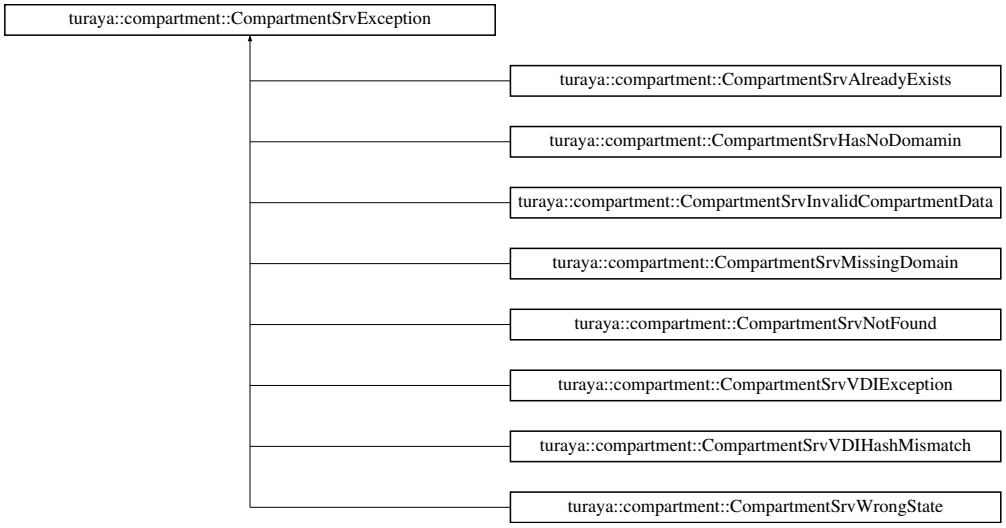
6.14.1.2 virtual turaya::compartment::CompartmentSrvAlreadyExists-::~CompartmentSrvAlreadyExists ( ) throw () [inline, virtual]

```
{};
```

6.15 turaya::compartment::CompartmentSrvException Class - Reference

`#include <CompartmentExceptionsSrv.hxx>`

Inheritance diagram for turaya::compartment::CompartmentSrvException:



Public Member Functions

- CompartmentSrvException (const std::string &context)
- virtual ~CompartmentSrvException () throw ()

6.15.1 Constructor & Destructor Documentation

6.15.1.1 turaya::compartment::CompartmentSrvException::-CompartmentSrvException (const std::string & *context*) [inline]

```
                                                                :
                                    runtime_error(context) {};
```

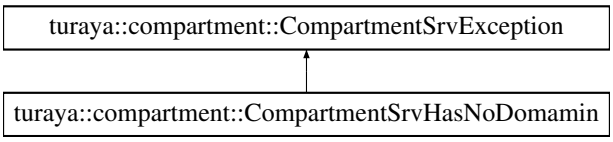
6.15.1.2 virtual turaya::compartment::CompartmentSrvException-::~CompartmentSrvException ( ) throw () [inline, virtual]

```
{};
```

6.16 turaya::compartment::CompartmentSrvHasNoDomamin - Class Reference

```
#include <CompartmentExceptionsSrv.hxx>
```

Inheritance diagram for turaya::compartment::CompartmentSrvHasNoDomamin:



Public Member Functions

- CompartmentSrvHasNoDomamin (const std::string &context="CompartmentSrv-HasNoDomamin")
- virtual ~CompartmentSrvHasNoDomamin () throw ()

6.16.1 Constructor & Destructor Documentation

6.16.1.1 turaya::compartment::CompartmentSrvHasNoDomamin::-CompartmentSrvHasNoDomamin (const std::string & *context* = "CompartmentSrvHasNoDomamin") [inline]

```
                                :
                    CompartmentSrvException(context){};
```

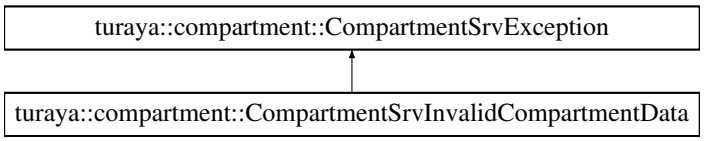
6.16.1.2 virtual turaya::compartment::CompartmentSrvHasNoDomamin-::~CompartmentSrvHasNoDomamin ( ) throw () [inline, virtual]

```
{};
```

6.17 turaya::compartment::CompartmentSrvInvalidCompartment-Data Class Reference

```
#include <CompartmentExceptionsSrv.hxx>
```

Inheritance diagram for turaya::compartment::CompartmentSrvInvalidCompartment-Data:



Public Member Functions

- CompartmentSrvInvalidCompartmentData (const std::string &context="-CompartmentSrvInvalidCompartmentData")
- virtual ~CompartmentSrvInvalidCompartmentData () throw ()

6.17.1 Constructor & Destructor Documentation

6.17.1.1 turaya::compartment::CompartmentSrvInvalidCompartmentData::-CompartmentSrvInvalidCompartmentData (const std::string & *context* = "CompartmentSrvInvalidCompartmentData") [inline]

```
                                        :
CompartmentSrvException(context){};
```

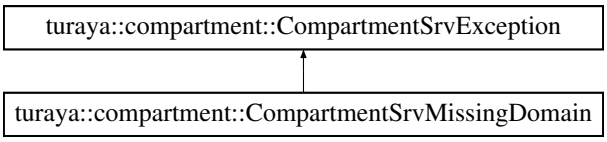
6.17.1.2 virtual turaya::compartment::CompartmentSrvInvalidCompartmentData-::~CompartmentSrvInvalidCompartmentData ( ) throw () [inline, virtual]

```
{};
```

6.18 turaya::compartment::CompartmentSrvMissingDomain Class Reference

`#include <CompartmentExceptionsSrv.hxx>`

Inheritance diagram for turaya::compartment::CompartmentSrvMissingDomain:



Public Member Functions

- CompartmentSrvMissingDomain (const std::string &context="CompartmentSrvMissingDomain")
- virtual ~CompartmentSrvMissingDomain () throw ()

6.18.1 Constructor & Destructor Documentation

6.18.1.1 turaya::compartment::CompartmentSrvMissingDomain::CompartmentSrvMissingDomain (const std::string & *context* = "CompartmentSrvMissingDomain") [inline]

```
:
CompartmentSrvException(context){};
```

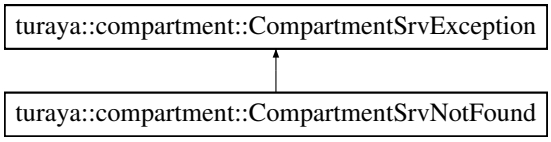
6.18.1.2 virtual turaya::compartment::CompartmentSrvMissingDomain::~CompartmentSrvMissingDomain ( ) throw () [inline, virtual]

```
{};
```

6.19 turaya::compartment::CompartmentSrvNotFound Class Reference

`#include <CompartmentExceptionsSrv.hxx>`

Inheritance diagram for turaya::compartment::CompartmentSrvNotFound:



Public Member Functions

- CompartmentSrvNotFound (const std::string &context="CompartmentSrvNotFound")
- virtual ~CompartmentSrvNotFound () throw ()

6.19.1 Constructor & Destructor Documentation

6.19.1.1 turaya::compartment::CompartmentSrvNotFound::CompartmentSrvNotFound (const std::string & *context* = "CompartmentSrvNotFound") [inline]

```
:
    CompartmentSrvException(context){};
```

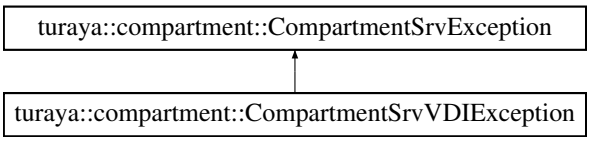
6.19.1.2 virtual turaya::compartment::CompartmentSrvNotFound::~CompartmentSrvNotFound ( ) throw () [inline, virtual]

```
{};
```

6.20 turaya::compartment::CompartmentSrvVDIException Class Reference

```
#include <CompartmentExceptionsSrv.hxx>
```

Inheritance diagram for turaya::compartment::CompartmentSrvVDIException:



Public Member Functions

- CompartmentSrvVDIException (const std::string &context="CompartmentSrvVDIException")

- virtual ~CompartmentSrvVDIException () throw ()

6.20.1 Constructor & Destructor Documentation

6.20.1.1 turaya::compartment::CompartmentSrvVDIException::-CompartmentSrvVDIException (const std::string & *context* = "CompartmentSrvVDIException") [inline]

```
:
CompartmentSrvException(context){};
```

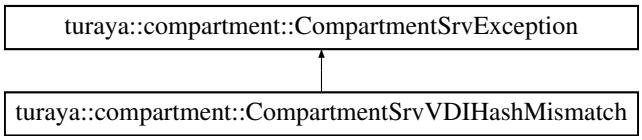
6.20.1.2 virtual turaya::compartment::CompartmentSrvVDIException-::~CompartmentSrvVDIException ( ) throw () [inline, virtual]

```
{};
```

6.21 turaya::compartment::CompartmentSrvVDIHashMismatch - Class Reference

```
#include <CompartmentExceptionsSrv.hxx>
```

Inheritance diagram for turaya::compartment::CompartmentSrvVDIHashMismatch:



Public Member Functions

- CompartmentSrvVDIHashMismatch (const std::string &context="Compartment-SrvVDIHashMismatch")
- virtual ~CompartmentSrvVDIHashMismatch () throw ()

6.21.1 Constructor & Destructor Documentation

6.21.1.1 turaya::compartment::CompartmentSrvVDIHashMismatch::-CompartmentSrvVDIHashMismatch (const std::string & *context* = "CompartmentSrvVDIHashMismatch") [inline]

```
:
CompartmentSrvException(context){};
```

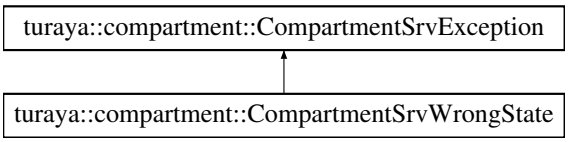
6.21.1.2 virtual turaya::compartment::CompartmentSrvVDIHashMismatch-::~CompartmentSrvVDIHashMismatch ( ) throw () [inline, virtual]

```
{};
```

6.22 turaya::compartment::CompartmentSrvWrongState Class Reference

```
#include <CompartmentExceptionsSrv.hxx>
```

Inheritance diagram for turaya::compartment::CompartmentSrvWrongState:



Public Member Functions

- CompartmentSrvWrongState (const std::string &context="CompartmentSrv-WrongState")
- virtual ~CompartmentSrvWrongState () throw ()

6.22.1 Constructor & Destructor Documentation

6.22.1.1 turaya::compartment::CompartmentSrvWrongState::-CompartmentSrvWrongState (const std::string & *context* = "CompartmentSrvWrongState") [inline]

```
:
 CompartmentSrvException(context){};
```

6.22.1.2 virtual turaya::compartment::CompartmentSrvWrongState-::~CompartmentSrvWrongState ( ) throw () [inline, virtual]

```
{};
```

6.23 turaya::domain::DomainAdaptor Class Reference

Adaptor class to make the DomainSrv class accessible over DBus.

`#include <DomainAdaptor.hxx>`

Public Types

- typedef sirrix::utils::SharedPointer < DomainAdaptor > Pointer

Public Member Functions

- DomainAdaptor (DomainSrv::Pointer domain, std::string dBusPath)
- virtual ~DomainAdaptor () throw ()
- void update (dbuspp::fixed_array< unsigned char > dBusDomainData)
- DomainID getID () const
- std::string getName () const
- UInt32 getColor () const
- void remove ()
- UInt32 getStatus () const
- virtual void onStatusChanged (Domain::Status status)
- dbuspp::fixed_array< unsigned char > encrypt (const dbuspp::fixed_array< unsigned char > &plainData) const
- dbuspp::fixed_array< unsigned char > decrypt (const dbuspp::fixed_array< unsigned char > &encodedData) const

Public Attributes

- sirrix::utils::Signal1 < turaya::DomainID > onDomainAdaptorRemoved

6.23.1 Detailed Description

Adaptor class to make the DomainSrv class accessible over DBus.

6.23.2 Member Typedef Documentation

6.23.2.1 typedef sirrix::utils::SharedPointer< DomainAdaptor > turaya::domain::DomainAdaptor::Pointer

6.23.3 Constructor & Destructor Documentation

6.23.3.1 DBUSPP_INTERFACE_IMPLEMENT_END_MAP DBUSPP_OBJECT_IMPLEMENT_END_MAP DomainAdaptor::DomainAdaptor (DomainSrv::Pointer *domain,* std::string *dBusPath*)

:

```
dbuspp::object(dBusPath),
onDomainAdaptorRemoved(),
myDomain(domain),
```

```
        statusChanged(),
        removed() {
//      debugFFL << "enter" << endl;
        myDomain->onStatusChanged.Connect(this, &DomainAdaptor::onStatusChanged
      );
//      debugFFL << "leave" << endl;
}
```

6.23.3.2 DomainAdaptor::~DomainAdaptor ( ) throw () [virtual]

```
                                    {
//      debugFFL << "enter" << endl;
        myDomain->onStatusChanged.Disconnect(this,&
      DomainAdaptor::onStatusChanged);
//      debugFFL << "leave" << endl;
}
```

6.23.4 Member Function Documentation

6.23.4.1 dbuspp::fixed_array< unsigned char > DomainAdaptor::decrypt (const dbuspp::fixed_array< unsigned char > & *encodedData*) const

```
                                      {
        try {
                ByteVector bv(encodedData.get(), encodedData.size());
                ByteVector decryptedData = myDomain->decrypt(bv);
                return dbuspp::fixed_array<unsigned char>(decryptedData.
      toCArray(), decryptedData.size());
        }catch (DomainSrvInvalidDomainData &e){
                debugFFL << "updateDomain failed: " << e.what() << endl;
                DBUSPP_THROW_ERROR_EXCEPTION(ERROR_INVALID_DOMAIN_DATA.c_str(),
       e.what());
        }
        return dbuspp::fixed_array<unsigned char>();
}
```

6.23.4.2 dbuspp::fixed_array< unsigned char > DomainAdaptor::encrypt (const dbuspp::fixed_array< unsigned char > & *plainData*) const

```
                                    {
        try {
                ByteVector bv(plainData.get(),plainData.size());
                ByteVector encryptedData = myDomain->encrypt(bv);
                return dbuspp::fixed_array<unsigned char>(encryptedData.
      toCArray(), encryptedData.size());
        }catch (DomainSrvInvalidDomainData &e){
                debugFFL << "updateDomain failed: " << e.what() << endl;
                DBUSPP_THROW_ERROR_EXCEPTION(ERROR_INVALID_DOMAIN_DATA.c_str(),
       e.what());
        }
        return dbuspp::fixed_array<unsigned char>();
}
```

6.23.4.3 UInt32 DomainAdaptor::getColor ( ) const

```
                                    {
        return myDomain->getColor().getColor();
}
```

6.23.4.4 DomainID DomainAdaptor::getID ( ) const

```
                                   {
        return myDomain->getID();
}
```

6.23.4.5 string DomainAdaptor::getName ( ) const

```
                                   {
        return myDomain->getName();
}
```

6.23.4.6 UInt32 DomainAdaptor::getStatus ( ) const

```
                                     {
        return myDomain->getStatus();
}
```

6.23.4.7 void DomainAdaptor::onStatusChanged (Domain::Status *status*) [virtual]

```
                                                          {
        statusChanged(status);
}
```

6.23.4.8 void DomainAdaptor::remove ( )

```
                        {
        myDomain->remove();
}
```

6.23.4.9 void DomainAdaptor::update (dbuspp::fixed_array< unsigned char > *dBusDomainData*)

```
                                                                                {
        try {
                ByteVector bv(dBusDomainData.get(),dBusDomainData.size());
                DomainData domainData(bv);
```

```
                myDomain->update(domainData);
         }catch (DomainSrvInvalidDomainData &e){
                debugFFL << "updateDomain failed: " << e.what() << endl;
                DBUSPP_THROW_ERROR_EXCEPTION(ERROR_INVALID_DOMAIN_DATA.c_str(),
        e.what());
         }
}
```

6.23.5 Member Data Documentation

6.23.5.1 sirrix::utils::Signal1<turaya::DomainID> turaya::domain::DomainAdaptor::onDomainAdaptorRemoved

6.24 turaya::domain::DomainManagerAdaptor Class Reference

Adaptor class to make the DomainManagerSrv class accessible over DBus.

`#include <DomainManagerAdaptor.hxx>`

Public Types

- typedef std::map< DomainID, DomainAdaptor::Pointer > DomainAdaptorPtrs

Public Member Functions

- DomainManagerAdaptor (const std::string &objectPath, dbuspp::service_sptr spService)
- virtual ~DomainManagerAdaptor () throw ()
- std::vector< std::string > getAllDomains ()
- void installDomain (dbuspp::fixed_array< unsigned char > dBusDomainData)
- void createDomainAdaptors ()

6.24.1 Detailed Description

Adaptor class to make the DomainManagerSrv class accessible over DBus.

6.24.2 Member Typedef Documentation

6.24.2.1 typedef std::map< DomainID, DomainAdaptor::Pointer > turaya::domain::DomainManagerAdaptor::DomainAdaptorPtrs

6.24.3 Constructor & Destructor Documentation

6.24.3.1 DBUSPP_INTERFACE_IMPLEMENT_END_MAP DBUSPP_OBJECT_IMPLEMENT_END_MAP DomainManagerAdaptor::DomainManagerAdaptor (const std::string & *objectPath,* dbuspp::service_sptr *spService*)

```
                        :
        dbuspp::object(objectPath),
        myDomainAdaptors(),
        myDomainManagerSrv(),
        myService(spService),
        domainInstalled(),
        domainRemoved(){
//      debugFFL << "enter" << endl;

        myDomainManagerSrv.signalDomainInstalled.Connect(this, &
      DomainManagerAdaptor::slotDomainInstalled);
        myDomainManagerSrv.signalDomainRemoved.Connect(this, &
      DomainManagerAdaptor::slotDomainRemoving);
        createDomainAdaptors();
//      debugFFL << "leave" << endl;
}
```

6.24.3.2 DomainManagerAdaptor::~DomainManagerAdaptor ( ) throw () [virtual]

```
                                                    {
//      debugFFL << "enter" << endl;
        myDomainAdaptors.erase(myDomainAdaptors.begin(), myDomainAdaptors.end()
      );
//      debugFFL << "leave" << endl;
}
```

6.24.4 Member Function Documentation

6.24.4.1 void DomainManagerAdaptor::createDomainAdaptors ( )

```
                                            {
        DomainManagerSrv::DomainSrvPtrs domains = myDomainManagerSrv.
      getAllDomains();

        DomainManagerSrv::DomainSrvPtrs::iterator i;
        for(i = domains.begin(); i != domains.end(); i++){
                string path = createDomainPath(i->second->getID());
                debugFFL << "Create DomainAdaptor for Domain with ID "
                                << i->second->getID() << " at path " << path <<
       endl;

                DomainAdaptor::Pointer domAdapt(new DomainAdaptor(i->second,
      path));
                domAdapt->connect(myService);
                domAdapt->onDomainAdaptorRemoved.Connect(this, &
      DomainManagerAdaptor::slotDomainRemoving);
                myDomainAdaptors.insert(pair<DomainID, DomainAdaptor::Pointer>(
      i->second->getID(), domAdapt));
        }
}
```

6.24.4.2 std::vector< std::string > DomainManagerAdaptor::getAllDomains ( )

```
                                                                {

        DomainAdaptorPtrs::iterator i;
        std::vector< std::string > paths;
        for(i = myDomainAdaptors.begin(); i != myDomainAdaptors.end(); i++){
                paths.push_back((*i).second->path());
        }
        return paths;
}
```

6.24.4.3 void DomainManagerAdaptor::installDomain (dbuspp::fixed_array< unsigned char > *dBusDomainData*)

```
              {

        try {
                ByteVector bv(dBusDomainData.get(),dBusDomainData.size());
                DomainData domainData = DomainData(bv);
                myDomainManagerSrv.installDomain(domainData);
        }catch (DomainSrvAlreadyExists &e){
                debugFFL << "installDomain failed: " << e.what() << endl;
                DBUSPP_THROW_ERROR_EXCEPTION(ERROR_DOMAIN_ALREADY_EXISTS.c_str(
      ), e.what());
        }
        catch (DomainSrvInvalidDomainData &e){
                debugFFL << "installDomain failed: " << e.what() << endl;
                DBUSPP_THROW_ERROR_EXCEPTION(ERROR_INVALID_DOMAIN_DATA.c_str(),
       e.what());

        }
}
```

6.25 turaya::domain::DomainManagerSrv Class Reference

Server side DomainManager representation.

`#include <DomainManagerSrv.hxx>`

Public Types

- typedef std::map< DomainID, DomainSrv::Pointer > DomainSrvPtrs

Public Member Functions

- DomainManagerSrv ()
- virtual ~DomainManagerSrv () throw ()
- DomainSrvPtrs getAllDomains ()

  Returns all installed Domains.

- bool hasDomain (DomainID domainID)

  checks if a domain is already installed.
- void installDomain (DomainData domainData)

  Install a new Domain.
- void removeDomain (DomainID domainID)

  Remove a new Domain.

Public Attributes

- sirrix::utils::Signal1 < DomainSrv::Pointer > signalDomainInstalled
- sirrix::utils::Signal1 < turaya::DomainID > signalDomainRemoved

  Emitted on removement of a Compartment.

6.25.1 Detailed Description

Server side DomainManager representation.

6.25.2 Member Typedef Documentation

6.25.2.1 typedef std::map<DomainID, DomainSrv::Pointer> turaya::domain::DomainManagerSrv::DomainSrvPtrs

6.25.3 Constructor & Destructor Documentation

6.25.3.1 DomainManagerSrv::DomainManagerSrv ( )

```
                                    :
signalDomainInstalled(),
signalDomainRemoved(),
myDomains(){
//      debugFFL << "enter" << endl;
        init(); //adopted from old libCompartment
        loadDomains();
//      debugFFL << "leave" << endl;
}
```

6.25.3.2 DomainManagerSrv::~DomainManagerSrv ( ) throw () [virtual]

```
                                           {
        debugFFL << "~DomainManagerSrv()" << endl;
        myDomains.erase(myDomains.begin(), myDomains.end());
//      debugFFL << "leave" << endl;
}
```

6.25.4 Member Function Documentation

6.25.4.1 DomainManagerSrv::DomainSrvPtrs DomainManagerSrv::getAllDomains ( )

Returns all installed Domains.

Returns

All Domains.

```
                                                                    {
        return myDomains;
}
```

6.25.4.2 bool DomainManagerSrv::hasDomain (DomainID *domainID*)

checks if a domain is already installed.

Returns

true if domain exists else false.

```
                                                  {
        stringstream sstr;
        sstr << "SELECT COUNT(*) FROM " << DOMAINTABLE << " WHERE id = " <<
      domainID;
        sqlite3 * db;
        sqlite3_stmt *statement;
        if (sqlite3_open(DataBaseUtils::getDatabasePath().c_str(), &db) !=
      SQLITE_OK || sqlite3_prepare_v2(db, sstr.str().c_str(), -1, &statement, NULL) !=
      SQLITE_OK) {
                sqlite3_finalize(statement);
                sqlite3_close(db);
                throw DataBaseError();
        }
        sqlite3_step(statement);
        bool result(sqlite3_column_int(statement, 0));
        sqlite3_finalize(statement);
        sqlite3_close(db);
        return result;
}
```

6.25.4.3 void DomainManagerSrv::installDomain (DomainData *domainData*)

Install a new Domain.

Parameters

domainData	DomainData describing the Domain to install

Exceptions

DomainSrvAlready-Exists	if the Domain already exists
DomainSrvInvalid-CompartmentData	if the given DomainData is invalid

```
                                                                {

        debugFFL << "Installing new Domain with ID " << domainData.getID() <<
      endl;
        DomainSrv::Pointer newDomain = internal_install(domainData);
        signalDomainInstalled(newDomain);
}
```

6.25.4.4 void DomainManagerSrv::removeDomain (DomainID *domainID*)

Remove a new Domain.

Parameters

domainID	describing the Domain to remove

Exceptions

DomainSrvNotFound	if Domain is allready removed

```
                                                          {
        DomainSrvPtrs::iterator i = myDomains.find(domainID);
        if (i == myDomains.end()){
                debugFFL << "Domain with ID " << domainID << " not found" <<
      endl;
                throw DomainSrvNotFound();
        }
        internal_remove(i->second);
        myDomains.erase(i);
}
```

6.25.5 Member Data Documentation

6.25.5.1 sirrix::utils::Signal1<DomainSrv::Pointer> turaya::domain::Domain-ManagerSrv::signalDomainInstalled

6.25.5.2 sirrix::utils::Signal1<turaya::DomainID> turaya::domain::DomainManagerSrv-::signalDomainRemoved

Emitted on removement of a Compartment.

6.26 turaya::domain::DomainSrv Class Reference

Server side domain representation.

`#include <DomainSrv.hxx>`

Public Types

- typedef sirrix::utils::SharedPointer < DomainSrv > Pointer

Public Member Functions

- DomainSrv (const DomainID &domainID)
- virtual ~DomainSrv ()
- DomainID getID () const

  Returns the domain identifier of this domain.
- std::string getName () const
- turaya::Color getColor () const
- turaya::Domain::Status getStatus () const
- void update (const DomainData &domainData)
- sirrix::utils::ByteVector encrypt (const sirrix::utils::ByteVector &plainData) const

  Encrypts data using the domain specific key.
- sirrix::utils::ByteVector decrypt (const sirrix::utils::ByteVector &nonceAndEncData) const

  Decrypts data using the domain specific key.
- void remove ()

  Uninstalls this Domain.

Public Attributes

- sirrix::utils::Signal1 < turaya::Domain::Status > onStatusChanged

  Emitted when status of this Domain has changed.
- sirrix::utils::Signal1 < turaya::DomainID > signalDomainRemoved

  Emitted on removement of this Domain.

6.26.1 Detailed Description

Server side domain representation.

6.26.2 Member Typedef Documentation

6.26.2.1 typedef sirrix::utils::SharedPointer< DomainSrv > turaya::domain::DomainSrv::Pointer

6.26.3 Constructor & Destructor Documentation

6.26.3.1 DomainSrv::DomainSrv (const DomainID & *domainID*)

```
                                                :
        onStatusChanged(),
        signalDomainRemoved(),
        myID( domainID ),
        myName( DataBaseUtils::getString( DOMAINTABLE, "name", myID ) ),
   myColor( DataBaseUtils::getInt( DOMAINTABLE, "color", myID ) ),
        myStatus(turaya::Domain::undefined)

{
}
```

6.26.3.2 DomainSrv::~DomainSrv ( ) [virtual]

```
                      {
        debugFFL << "~DomainSrv()" << endl;
}
```

6.26.4 Member Function Documentation

6.26.4.1 sirrix::utils::ByteVector DomainSrv::decrypt (const sirrix::utils::ByteVector & *nonceAndEncData*) const

Decrypts data using the domain specific key.

```
                {
        ScopedPointer<Decrypter> pDecrypter(getDomainDecrypter());
        unsigned int nonceLen =  pDecrypter->getNonceLength(); //todo fix it
        ByteVector nonce = nonceAndEncData.left(nonceLen);
        ByteVector encData = nonceAndEncData.right(nonceAndEncData.size() -
      nonceLen);
        return pDecrypter->decrypt(encData, nonce);
}
```

6.26.4.2 sirrix::utils::ByteVector DomainSrv::encrypt (const sirrix::utils::ByteVector & *plainData*) const

Encrypts data using the domain specific key.

```
            {
        ScopedPointer<Decrypter> pDecrypter(getDomainDecrypter());
```

```
        ScopedPointer<Encrypter> pEncrypter(pDecrypter->getEncrypter());
        if (!pEncrypter){
                throw DomainSrvException("Failed to obtain encrypter.");
        }
        ByteVector nonce(pEncrypter->getNonceLength()); //TODO fix it
        rnd >> nonce;
        ByteVector encData =  pEncrypter->encrypt(plainData, nonce);
        return nonce+encData;
}
```

6.26.4.3 turaya::Color DomainSrv::getColor ( ) const

```
                                        {
        return myColor;
}
```

6.26.4.4 DomainID DomainSrv::getID ( ) const

Returns the domain identifier of this domain.

```
                                {
        return myID;
}
```

6.26.4.5 string DomainSrv::getName ( ) const

```
                                {
        return myName;
}
```

6.26.4.6 turaya::Domain::Status DomainSrv::getStatus ( ) const

```
                                                {
        return myStatus;
}
```

6.26.4.7 void DomainSrv::remove ( )

Uninstalls this Domain.

Exceptions

DomainSrvWrong-State	if this domain in not currently stopped

```
                        {
```

```
//      if (myStatus != Domain::stopped){
//              throw DomainSrvWrongState("Compartment must be in state stopped
       for remove()");
//      }
//      myStatus = Domain::removed;
//      onStatusChanged(myStatus);
        //onDomainRemoved(myID);
        signalDomainRemoved(myID);
//      debugFFL << "leave" << endl;
}
```

6.26.4.8 void DomainSrv::update (const DomainData & *domainData*)

```
                                                    {
        debugFFL << "enter" << endl;
        if (myID != domainData.getID()){
                stringstream sstr;
                sstr << "DomainID mismatch. Domain with ID " << myID
                                << " can not be updated thru DomainData with
       ID "<< domainData.getID();
                debugFFL << sstr.str() << endl;
                throw DomainSrvInvalidDomainData(sstr.str());
        }
        string name = domainData.getName();
        Color color = domainData.getColor();
        if (myName.compare(name) == 0 && myColor == color){
                debugFFL << "Domain " << myID << " already up-to-date." << endl
     ;
                return;
        }
        myName = name;
        myColor = color;

        stringstream sstr;
        sstr << "UPDATE " << DOMAINTABLE << " SET name = \"" << myName << "\" ,
       color = " << myColor.getColor() << " " <<
                        "WHERE id = " << myID;
        debugFFL << "Execute: " << sstr.str() << endl;
        try {
                DataBaseUtils::sqlExecute(sstr.str());
        }catch (std::exception &e){
                debugFFL << "Exception occurred: " << e.what() << endl;
                throw DomainSrvException(e.what());
        }
        debugFFL << "leave" << endl;
}
```

6.26.5 Member Data Documentation

6.26.5.1 sirrix::utils::Signal1<turaya::Domain::Status> turaya::domain::DomainSrv::onStatusChanged

Emitted when status of this Domain has changed.

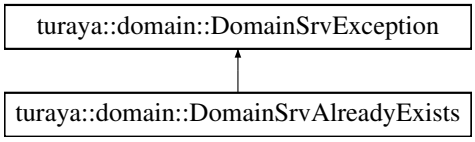
6.26.5.2 sirrix::utils::Signal1<turaya::DomainID> turaya::domain::DomainSrv::signalDomainRemoved

Emitted on removement of this Domain.

6.27 turaya::domain::DomainSrvAlreadyExists Class Reference

`#include <DomainExceptionsSrv.hxx>`

Inheritance diagram for turaya::domain::DomainSrvAlreadyExists:



Public Member Functions

- DomainSrvAlreadyExists (const string &context="DomainSrvAlreadyExists")
- virtual ~DomainSrvAlreadyExists () throw ()

6.27.1 Constructor & Destructor Documentation

6.27.1.1 turaya::domain::DomainSrvAlreadyExists::DomainSrvAlreadyExists (const string & *context* = "DomainSrvAlreadyExists") [inline]

```
:
        DomainSrvException(context){};
```

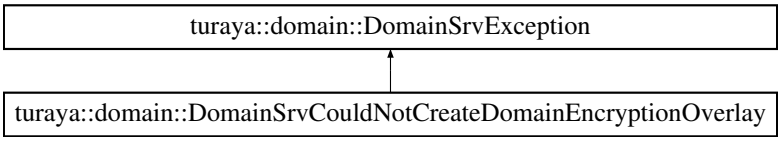
6.27.1.2 virtual turaya::domain::DomainSrvAlreadyExists::~DomainSrvAlreadyExists ( ) throw () [inline, virtual]

```
{};
```

6.28 turaya::domain::DomainSrvCouldNotCreateDomainEncryptionOverlay Class Reference

`#include <DomainExceptionsSrv.hxx>`

Inheritance diagram for turaya::domain::DomainSrvCouldNotCreateDomainEncryptionOverlay:



Public Member Functions

- DomainSrvCouldNotCreateDomainEncryptionOverlay (const string &context="-DomainSrvCouldNotCreateDomainEncryptionOverlay")
- virtual ~DomainSrvCouldNotCreateDomainEncryptionOverlay () throw ()

6.28.1 Constructor & Destructor Documentation

6.28.1.1 turaya::domain::DomainSrvCouldNotCreateDomainEncryptionOverlay-::DomainSrvCouldNotCreateDomainEncryptionOverlay (const string & *context* = "DomainSrvCouldNotCreateDomainEncryption-Overlay") [inline]

:

```
DomainSrvException(context){};
```

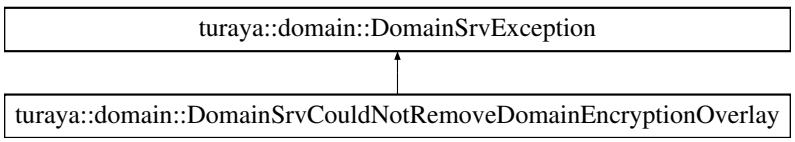
6.28.1.2 virtual turaya::domain::DomainSrvCouldNotCreateDomainEncryption-Overlay::~DomainSrvCouldNotCreateDomainEncryptionOverlay ( ) throw () [inline, virtual]

```
{};
```

6.29 turaya::domain::DomainSrvCouldNotRemoveDomainEncryption-Overlay Class Reference

```
#include <DomainExceptionsSrv.hxx>
```

Inheritance diagram for turaya::domain::DomainSrvCouldNotRemoveDomain-EncryptionOverlay:



Public Member Functions

- DomainSrvCouldNotRemoveDomainEncryptionOverlay (const string &context="-DomainSrvCouldNotRemoveDomainEncryptionOverlay")
- virtual ~DomainSrvCouldNotRemoveDomainEncryptionOverlay () throw ()

6.29.1 Constructor & Destructor Documentation

6.29.1.1 turaya::domain::DomainSrvCouldNotRemoveDomainEncryptionOverlay::DomainSrvCouldNotRemoveDomainEncryptionOverlay (const string & *context* = "DomainSrvCouldNotRemoveDomainEncryptionOverlay") [inline]

:

```
DomainSrvException(context){};
```

6.29.1.2 virtual turaya::domain::DomainSrvCouldNotRemoveDomainEncryptionOverlay::~DomainSrvCouldNotRemoveDomainEncryptionOverlay ( ) throw () [inline, virtual]

```
{};
```

6.30 turaya::domain::DomainSrvException Class Reference

```
#include <DomainExceptionsSrv.hxx>
```

Inheritance diagram for turaya::domain::DomainSrvException:

Public Member Functions

- DomainSrvException (const string &context)
- virtual ~DomainSrvException () throw ()

6.30.1 Constructor & Destructor Documentation

6.30.1.1 turaya::domain::DomainSrvException::DomainSrvException (const string & *context*) [inline]

:

```
runtime_error(context) {};
```

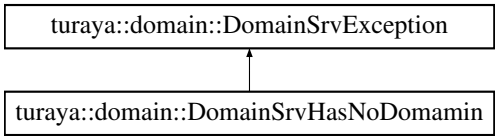
6.30.1.2 virtual turaya::domain::DomainSrvException::~DomainSrvException ( ) throw () [inline, virtual]

```
{};
```

6.31 turaya::domain::DomainSrvHasNoDomamin Class Reference

`#include <DomainExceptionsSrv.hxx>`

Inheritance diagram for turaya::domain::DomainSrvHasNoDomamin:



Public Member Functions

- DomainSrvHasNoDomamin (const string &context="DomainSrvHasNoDomamin")
- virtual ~DomainSrvHasNoDomamin () throw ()

6.31.1 Constructor & Destructor Documentation

6.31.1.1 turaya::domain::DomainSrvHasNoDomamin::DomainSrvHasNoDomamin (const string & *context* = "DomainSrvHasNoDomamin") [inline]

```
:
        DomainSrvException(context){};
```

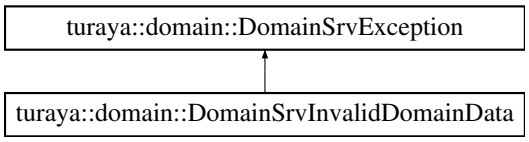
6.31.1.2 virtual turaya::domain::DomainSrvHasNoDomamin::~DomainSrvHasNoDomamin ( ) throw () [inline, virtual]

```
{};
```

6.32 turaya::domain::DomainSrvInvalidDomainData Class Reference

`#include <DomainExceptionsSrv.hxx>`

Inheritance diagram for turaya::domain::DomainSrvInvalidDomainData:



Public Member Functions

- DomainSrvInvalidDomainData (const string &context="DomainSrvInvalid-DomainData")
- virtual ~DomainSrvInvalidDomainData () throw ()

6.32.1 Constructor & Destructor Documentation

6.32.1.1 turaya::domain::DomainSrvInvalidDomainData::-DomainSrvInvalidDomainData (const string & *context* = "DomainSrvInvalidDomainData") [inline]

```
:
DomainSrvException(context){};
```

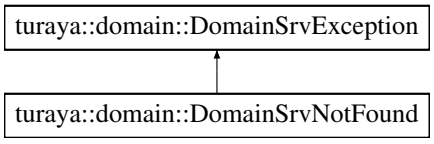
6.32.1.2 virtual turaya::domain::DomainSrvInvalidDomainData::~-DomainSrvInvalidDomainData ( ) throw () [inline, virtual]

```
{};
```

6.33 turaya::domain::DomainSrvNotFound Class Reference

```
#include <DomainExceptionsSrv.hxx>
```

Inheritance diagram for turaya::domain::DomainSrvNotFound:



Public Member Functions

- DomainSrvNotFound (const string &context="DomainSrvNotFound")
- virtual ~DomainSrvNotFound () throw ()

6.33.1 Constructor & Destructor Documentation

6.33.1.1 turaya::domain::DomainSrvNotFound::DomainSrvNotFound (const string & *context* = "DomainSrvNotFound") [inline]

```
:
                                DomainSrvException(context){};
```

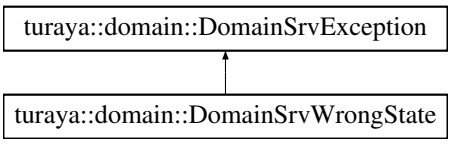
6.33.1.2 virtual turaya::domain::DomainSrvNotFound::~DomainSrvNotFound ( ) throw () [inline, virtual]

```
{};
```

6.34 turaya::domain::DomainSrvWrongState Class Reference

```
#include <DomainExceptionsSrv.hxx>
```

Inheritance diagram for turaya::domain::DomainSrvWrongState:



Public Member Functions

- DomainSrvWrongState (const string &context="DomainSrvWrongState")
- virtual ~DomainSrvWrongState () throw ()

6.34.1 Constructor & Destructor Documentation

6.34.1.1 turaya::domain::DomainSrvWrongState::DomainSrvWrongState (const string & *context* = "DomainSrvWrongState") [inline]

```
:
                        DomainSrvException(context){};
```

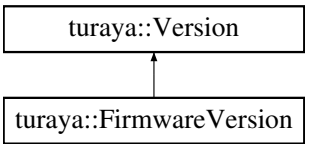
6.34.1.2 virtual turaya::domain::DomainSrvWrongState::~DomainSrvWrongState ( ) throw () [inline, virtual]

```
{};
```

6.35 turaya::FirmwareVersion Class Reference

`#include <TrustedDesktop.hxx>`

Inheritance diagram for turaya::FirmwareVersion:



Public Member Functions

- FirmwareVersion (UInt32 major, UInt32 minor, UInt32 patchlevel, UInt32 build)
- UInt32 getBuild () const

  Get build identifier.
- std::string getString () const

  Get version as string: "major.minor.patchlevel (build)".
- bool operator== (const FirmwareVersion &other)

  compare equal operator
- bool operator!= (const FirmwareVersion &other)

  compare not equal operator
- bool operator>= (const FirmwareVersion &other)

  compare greater or equal operator
- bool operator> (const FirmwareVersion &other)

  compare greater operator
- bool operator<= (const FirmwareVersion &other)

  compare smaller or equal operator
- bool operator< (const FirmwareVersion &other)

  compare smaller operator

6.35.1 Constructor & Destructor Documentation

6.35.1.1 turaya::FirmwareVersion::FirmwareVersion (UInt32 *major*, UInt32 *minor*, UInt32 *patchlevel*, UInt32 *build*) [inline]

Constructor

Parameters

major	Major version number
minor	Minor version number
patchlevel	Patch level
build	Build identifier

```
                        :
                                Version( major, minor, patchlevel ),
    myBuild( build ) {
                    }
```

6.35.2 Member Function Documentation

6.35.2.1 UInt32 turaya::FirmwareVersion::getBuild ( ) const [inline]

Get build identifier.

```
                                    {
                            return myBuild;
                    }
```

6.35.2.2 std::string turaya::FirmwareVersion::getString ( ) const [inline]

Get version as string: "major.minor.patchlevel (build)".

Reimplemented from turaya::Version.

```
                                    {
                            std::stringstream ss;
                            ss << Version::getString() << " (" << myBuild <
    < ")";
                            return ss.str();
                    }
```

6.35.2.3 bool turaya::FirmwareVersion::operator!= (const FirmwareVersion & *other*) [inline]

compare not equal operator

```
                                                                    {
                            return !( *this == other );
                    }
```

6.35.2.4 bool turaya::FirmwareVersion::operator< (const FirmwareVersion & *other*) [inline]

compare smaller operator

```
                                                                {
                            return ( ( myBuild < other.myBuild &&
    Version::operator==( other ) ) /* 0.11.3 build 51 < 0.11.3 build 52 */
                            || ( Version::operator<( other ) ) ); /* 0.11.2
     build 12 < 0.11.3 build 4 */
                    }
```

6.35.2.5 bool turaya::FirmwareVersion::operator<= (const FirmwareVersion & *other*) [inline]

compare smaller or equal operator

```
                                                                    {
                        return ( myBuild <= other.myBuild &&
    Version::operator<=( other ) );
                    }
```

6.35.2.6 bool turaya::FirmwareVersion::operator== (const FirmwareVersion & *other*) [inline]

compare equal operator

```
                                                                    {
                        return ( Version::operator==( other ) &&
    myBuild == other.myBuild );
                    }
```

6.35.2.7 bool turaya::FirmwareVersion::operator> (const FirmwareVersion & *other*) [inline]

compare greater operator

```
                                                                   {
                        return ( ( myBuild > other.myBuild &&
    Version::operator==( other ) ) /* 0.11.3 build 52 > 0.11.3 build 51 */
                        || ( Version::operator>( other ) ) ); /* 0.11.3
     build 12 > 0.11.2 build 52 */
                    }
```

6.35.2.8 bool turaya::FirmwareVersion::operator>= (const FirmwareVersion & *other*) [inline]

compare greater or equal operator

```
                                                                    {
                        return ( myBuild >= other.myBuild &&
    Version::operator>=( other ) );
                    }
```

6.36 turaya::compartment::FTPAccess Class Reference

```
#include <FTPAccess.hxx>
```

Public Member Functions

- FTPAccess (std::string interface)
- virtual ~FTPAccess ()
- void getFile (const std::string &url, const sirrix::os::Path &targetPath, sirrix::utils::Delegate1< UInt64 > delegate=sirrix::utils::Delegate1< UInt64 >())
- void setFile (const sirrix::os::Path &sourcePath, const std::string &url, sirrix::utils::Delegate1< UInt64 > delegate=sirrix::utils::Delegate1< UInt64 >())

Static Public Member Functions

- static int getFileprogressFunction (void *clientp, double dltotal, double dlnow, double ultotal, double ulnow)
- static int setFileprogressFunction (void *clientp, double dltotal, double dlnow, double ultotal, double ulnow)

6.36.1 Constructor & Destructor Documentation

6.36.1.1 turaya::compartment::FTPAccess::FTPAccess (std::string *interface*)

6.36.1.2 FTPAccess::~FTPAccess ( ) [virtual]

```
                    {
        if ( myCurl != 0) {
                curl_easy_cleanup( myCurl );
                myCurl = 0;
        }
}
```

6.36.2 Member Function Documentation

6.36.2.1 void FTPAccess::getFile (const std::string & *url,* const sirrix::os::Path & *targetPath,* sirrix::utils::Delegate1< UInt64 > *delegate* = sirrix::utils::Delegate1<UInt64>())

```
                                        {

        FILE* file = openfile( targetPath, "w");

        CURLcode result = curl_easy_setopt(myCurl, CURLOPT_WRITEDATA, file);
        if ( CURLE_OK != result ) {
                throw FTPException ( string( "setting option CURLOPT_WRITEDATA
failed with " ) + string_cast(result) );
        }

        result = curl_easy_setopt( myCurl, CURLOPT_NOPROGRESS, 0L);
        if ( CURLE_OK != result ) {
                throw FTPException ( string( "setting option CURLOPT_NOPROGRESS
failed with " ) + string_cast(result) );
        }
```

```
        result = curl_easy_setopt( myCurl, CURLOPT_PROGRESSFUNCTION, &
      FTPAccess::getFileprogressFunction );
        if ( CURLE_OK != result ) {
                throw FTPException ( string( "setting option
       CURLOPT_PROGRESSFUNCTION failed with " ) + string_cast(result) );
        }


        result = curl_easy_setopt( myCurl, CURLOPT_PROGRESSDATA, this );
        if ( CURLE_OK != result ) {
                throw FTPException ( string( "setting option
       CURLOPT_PROGRESSDATA failed with " ) + string_cast(result) );
        }

        result = curl_easy_setopt( myCurl, CURLOPT_URL, url.c_str() );
        if ( CURLE_OK != result ) {
                throw FTPException ( string( "setting option CURLOPT_URL failed
       with " ) + string_cast(result) );
        }

        string interface("if!");
        interface += myInterface;
        result = curl_easy_setopt( myCurl, CURLOPT_INTERFACE, interface.c_str()
       );
        if ( CURLE_OK != result ) {
                throw FTPException ( string( "setting option CURLOPT_INTERFACE
       failed with " ) + string_cast(result) );
        }

        result = curl_easy_setopt( myCurl, CURLOPT_NETRC, CURL_NETRC_IGNORED );
        if ( CURLE_OK != result ) {
                throw FTPException ( string( "setting option CURLOPT_NETRC
       failed with " ) + string_cast(result) );
        }

        // register slot for signalling progress
        if (delegate != sirrix::utils::Delegate1<UInt64>()){
                debugFFL << "Connection signal" << endl;
                myCopyProgress.Connect( delegate );
        }

        result = curl_easy_perform( myCurl );
        if ( CURLE_OK != result ) {
                throw FTPException ( string( "call to curl_easy_perform failed
       with " ) + string_cast(result) );
        }
}
```

6.36.2.2 int FTPAccess::getFileprogressFunction (void * *clientp*, double *dltotal*, double *dlnow*, double *ultotal*, double *ulnow*) [static]

```
                                        {
        debugFFL << dltotal << ", " << dlnow << ", " << ultotal << ", " <<
      ulnow << endl;
        UInt32 percent = 0;
        if ( dltotal ) {
                        percent= dlnow*100/dltotal;
        }

        FTPAccess* fake_this = static_cast<FTPAccess*>(clientp);
```

```
        if (percent != fake_this->myPercent) {
                fake_this->myPercent = percent;
                fake_this->myCopyProgress(static_cast<UInt64>(fake_this->
      myPercent));
        }

        return 0;
}
```

6.36.2.3 void FTPAccess::setFile (const sirrix::os::Path & *sourcePath,* const std::string & *url,* sirrix::utils::Delegate1< UInt64 > *delegate =* sirrix::utils::Delegate1<UInt64>())

```
                                        {

        FILE* file = openfile( sourcePath, "r");

        CURLcode result = curl_easy_setopt(myCurl, CURLOPT_READDATA, file);
        if ( CURLE_OK != result ) {
                throw FTPException ( string( "setting option CURLOPT_READDATA
       failed with " ) + string_cast(result) );
        }

        result = curl_easy_setopt( myCurl, CURLOPT_NOPROGRESS, 0L);
        if ( CURLE_OK != result ) {
                throw FTPException ( string( "setting option CURLOPT_NOPROGRESS
       failed with " ) + string_cast(result) );
        }

        result = curl_easy_setopt( myCurl, CURLOPT_PROGRESSFUNCTION, &
      FTPAccess::setFileprogressFunction );
        if ( CURLE_OK != result ) {
                throw FTPException ( string( "setting option
       CURLOPT_PROGRESSFUNCTION failed with " ) + string_cast(result) );
        }

        result = curl_easy_setopt( myCurl, CURLOPT_PROGRESSDATA, this );
        if ( CURLE_OK != result ) {
                throw FTPException ( string( "setting option
       CURLOPT_PROGRESSDATA failed with " ) + string_cast(result) );
        }

        result = curl_easy_setopt( myCurl, CURLOPT_INFILESIZE_LARGE, (
      curl_off_t)FileManager::getFileSize( sourcePath ) );
        if ( CURLE_OK != result ) {
                throw FTPException ( string( "setting option
       CURLOPT_INFILESIZE_LARGE failed with " ) + string_cast(result) );
        }

        result = curl_easy_setopt( myCurl, CURLOPT_URL, url.c_str() );
        if ( CURLE_OK != result ) {
                throw FTPException ( string( "setting option CURLOPT_URL failed
       with " ) + string_cast(result) );
        }

        string interface("if!");
        interface += myInterface;
        result = curl_easy_setopt( myCurl, CURLOPT_INTERFACE, interface.c_str()
       );
```

```
        if ( CURLE_OK != result ) {
                throw FTPException ( string( "setting option CURLOPT_INTERFACE
       failed with " ) + string_cast(result) );
        }

        result = curl_easy_setopt( myCurl, CURLOPT_NETRC, CURL_NETRC_IGNORED );
        if ( CURLE_OK != result ) {
                throw FTPException ( string( "setting option CURLOPT_NETRC
       failed with " ) + string_cast(result) );
        }

        result = curl_easy_setopt( myCurl, CURLOPT_FTP_CREATE_MISSING_DIRS, 1L
      );
        if ( CURLE_OK != result ) {
                throw FTPException ( string( "setting option
       CURLOPT_FTP_CREATE_MISSING_DIRS failed with " ) + string_cast(result) );
        }

        result = curl_easy_setopt( myCurl, CURLOPT_UPLOAD, 1L );
        if ( CURLE_OK != result ) {
                throw FTPException ( string( "setting option CURLOPT_UPLOAD
       failed with " ) + string_cast(result) );
        }
        // register slot for signalling progress
        if (delegate != sirrix::utils::Delegate1<UInt64>()){
                debugFFL << "Connection signal" << endl;
                myCopyProgress.Connect( delegate );
        }

        result = curl_easy_perform( myCurl );
        if ( CURLE_OK != result ) {
                throw FTPException ( string( "call to curl_easy_perform failed
       with " ) + string_cast(result) );
        }
}
```

6.36.2.4 int FTPAccess::setFileprogressFunction (void * *clientp*, double *dltotal*, double *dlnow*, double *ultotal*, double *ulnow*) [static]

```
                                        {
        debugFFL << dltotal << ", " << dlnow << ", " << ultotal << ", " <<
      ulnow << endl;
        UInt32 percent = 0;
        if ( ultotal ) {
                percent= ulnow*100/ultotal;
        }
        FTPAccess* fake_this = static_cast<FTPAccess*>(clientp);
        if (percent != fake_this->myPercent) {
                fake_this->myPercent = percent;
                fake_this->myCopyProgress(static_cast<UInt64>(fake_this->
      myPercent));
        }
        return 0;
}
```

6.37 turaya::compartment::FTPException Class Reference

`#include <FTPAccess.hxx>`

Public Member Functions

- FTPException (const std::string &context)
- virtual ~FTPException () throw ()

6.37.1 Constructor & Destructor Documentation

6.37.1.1 turaya::compartment::FTPException::FTPException (const std::string & *context*) [inline]

```
                                                                    :
                                    runtime_error(context) {};
```

6.37.1.2 virtual turaya::compartment::FTPException::~FTPException ( ) throw () [inline, virtual]

```
{};
```

6.38 turaya::NetworkManager Class Reference

Client side NetworkManager representation.

`#include <NetworkManager.hxx>`

Public Types

- enum State { NM_STATE_UNKNOWN, NM_STATE_ASLEEP, NM_STATE_CONNECTING, NM_STATE_CONNECTED, NM_STATE_DISCONNECTED }

Public Member Functions

- NetworkManager (const NetworkManager &c)
- NetworkManager & operator= (const NetworkManager &c)
- ~NetworkManager () throw ()

Static Public Member Functions

- static NetworkManager & getInstance (TrustedDesktop &td)

Public Attributes

- sirrix::utils::Signal1< UInt32 > signalStateChanged
- sirrix::utils::Signal0< void > connectionLost

6.38.1 Detailed Description

Client side NetworkManager representation.

6.38.2 Member Enumeration Documentation

6.38.2.1 enum turaya::NetworkManager::State

Enumerator:

NM_STATE_UNKNOWN

NM_STATE_ASLEEP

NM_STATE_CONNECTING

NM_STATE_CONNECTED

NM_STATE_DISCONNECTED

```
{NM_STATE_UNKNOWN, NM_STATE_ASLEEP, NM_STATE_CONNECTING, NM_STATE_CONNECTED,
      NM_STATE_DISCONNECTED} State;
```

6.38.3 Constructor & Destructor Documentation

6.38.3.1 NetworkManager::NetworkManager (const NetworkManager & *c*)

```
                                                              :
signalStateChanged(),
connectionLost(),
myNetworkManagerProxy(o.myNetworkManagerProxy){
        debugFFL << "****enter***" << endl;
        connectSignals();
        debugFFL << "****leave ***" << endl;
}
```

6.38.3.2 NetworkManager::~NetworkManager ( ) throw ()

```
                                        {
        debugFFL << "enter" << endl;
        disconnectSignals();
        debugFFL << "leave" << endl;
}
```

6.38.4 Member Function Documentation

6.38.4.1 NetworkManager & NetworkManager::getInstance (TrustedDesktop & *td*) [static]

```
                                                                {
        dbuspp::dbus& dbus = td.getDBus();
        dbuspp::service_proxy_sptr service = dbus.client().service(
      NETWORK_MANAGER_SERVICE_NAME);;
        static NetworkManager instance(service);
        return instance;
}
```

6.38.4.2 NetworkManager & NetworkManager::operator= (const NetworkManager & *c*)

```
                                                                    {
        //debugFFL << "****enter***" << endl;
        if (this == &rhs){
                return *this;
        }
        disconnectSignals();
        myNetworkManagerProxy = rhs.myNetworkManagerProxy;
        connectSignals();
        return *this;
}
```

6.38.5 Member Data Documentation

6.38.5.1 sirrix::utils::Signal0<void> turaya::NetworkManager::connectionLost

6.38.5.2 sirrix::utils::Signal1< UInt32> turaya::NetworkManager::signalStateChanged

6.39 turaya::NetworkManagerInvalidInterface Class Reference

```
#include <NetworkManager.hxx>
```

Public Member Functions

- NetworkManagerInvalidInterface (const std::string &context="validate interfaces failed")
- virtual ~NetworkManagerInvalidInterface () throw ()

6.39.1 Constructor & Destructor Documentation

6.39.1.1 turaya::NetworkManagerInvalidInterface::NetworkManagerInvalidInterface (const std::string & *context* = "validate interfaces failed") [inline]

```
                :
        std::runtime_error(context){};
```

6.39.1.2 virtual turaya::NetworkManagerInvalidInterface::~NetworkManagerInvalidInterface ( ) throw () [inline, virtual]

```
{};
```

6.40 turaya::NetworkManagerProxyWrapper Class Reference

```
#include <NetworkManagerProxyWrapper.hxx>
```

Public Member Functions

- NetworkManagerProxyWrapper (const std::string &path, dbuspp::service_proxy_sptr &service)

  Creates a new CompartmentProxyWrapper object.
- virtual ~NetworkManagerProxyWrapper () throw ()

Public Attributes

- sirrix::utils::Signal1< UInt32 > signalStateChanged

6.40.1 Detailed Description

This class wraps the auto-generated NetworkManagerProxy class to offer signals that supports multiple slot-connections at the same time.

6.40.2 Constructor & Destructor Documentation

6.40.2.1 NetworkManagerProxyWrapper::NetworkManagerProxyWrapper (const std::string & *path*, dbuspp::service_proxy_sptr & *service*)

Creates a new CompartmentProxyWrapper object.

Parameters

path	Dbus path of the server-side object to connect to
Dbus	service where the appropriate server-side object exists

```
                                        :
NetworkManagerProxy(path),
signalStateChanged(){
        debugFFL << "enter" << endl;
        connect(service);
        (this->operator->())->StateChanged.connect(boost::bind(&
      NetworkManagerProxyWrapper::slotStateChanged, this, _1));
        debugFFL << "leave" << endl;
}
```

6.40.2.2 NetworkManagerProxyWrapper::~NetworkManagerProxyWrapper ( ) throw () [virtual]

```
                                                                    {
        debugFFL << "enter" << endl;
        (this->operator->())->StateChanged.disconnect();
        debugFFL << "leave" << endl;
}
```

6.40.3 Member Data Documentation

6.40.3.1 sirrix::utils::Signal1<UInt32> turaya::NetworkManagerProxyWrapper::signalStateChanged

6.41 turaya::organization::OrganizationAdaptor Class Reference

`#include <OrganizationAdaptor.hxx>`

Public Types

- typedef sirrix::utils::SharedPointer < OrganizationAdaptor > Pointer

Public Member Functions

- OrganizationAdaptor (OrganizationSrv::Pointer organization, std::string dBusPath)
- virtual ~OrganizationAdaptor () throw ()
- OrganizationID getID () const
- std::string getName () const
- std::vector < dbuspp::fixed_array < unsigned char > > getAllCompartmentData ()
- dbuspp::fixed_array< unsigned char > getDomainData (DomainID domainID)
- UInt32 getConnectionStatus () const
- void installCompartment (UInt32 compartmentID)
- void removeCompartment (UInt32 compartmentID)
- void installShare (const ShareInfo &shareInfo)
- void removeShare (UInt32 shareID)

- void remove ()
- dbuspp::fixed_array< unsigned char > authenticateUser (std::string username, std::string password)

6.41.1 Member Typedef Documentation

6.41.1.1 typedef sirrix::utils::SharedPointer< OrganizationAdaptor > turaya::organization::OrganizationAdaptor::Pointer

6.41.2 Constructor & Destructor Documentation

6.41.2.1 DBUSPP_INTERFACE_IMPLEMENT_END_MAP DBUSPP_OBJECT_IMPLEMENT_END_MAP OrganizationAdaptor::OrganizationAdaptor (OrganizationSrv::Pointer *organization,* std::string *dBusPath*)

```
                    :
    dbuspp::object(dBusPath),
    myOrganization(organization),
    removed(){
    debugFFL << "###enter###" << endl;
    myOrganization->signalRemoved.Connect(this, &
OrganizationAdaptor::slotOrganizationRemoved);
    myOrganization->signalConnectionStatusChanged.Connect(this, &
OrganizationAdaptor::slotConnectionStatusChanged);
    debugFFL << "###leave###" << endl;
}
```

6.41.2.2 OrganizationAdaptor::~OrganizationAdaptor ( ) throw () [virtual]

```
                                                {
    debugFFL << "###enter###" << endl;
    myOrganization->signalRemoved.Disconnect(this, &
OrganizationAdaptor::slotOrganizationRemoved);
    myOrganization->signalConnectionStatusChanged.Disconnect(this, &
OrganizationAdaptor::slotConnectionStatusChanged);
    debugFFL << "###leave###" << endl;
}
```

6.41.3 Member Function Documentation

6.41.3.1 dbuspp::fixed_array< unsigned char > OrganizationAdaptor::authenticateUser (std::string *username,* std::string *password*)

```
                          {
    debugFFL << "###enter###" << endl;
    ByteVector bv;
    try {
            bv = (myOrganization->authenticateUser(username, password)).
getEncodedData();
    }catch (OrganizationSrvException &e){
            debugFFL << "authenticateUser failed: " << e.what() << endl;
```

```
                DBUSPP_THROW_ERROR_EXCEPTION(ERROR_TOM_NOT_CONNECTED.c_str(), e
      .what());
          }
          dbuspp::fixed_array<unsigned char> fa(bv.toCArray(), bv.size());
          debugFFL << "###leave###" << endl;
          return fa;
}
```

6.41.3.2 vector< dbuspp::fixed_array< unsigned char > > OrganizationAdaptor::getAllCompartmentData ( )

```
              {
          debugFFL << "###enter###" << endl;
          CompartmentDataMap compData;
          try {
                  compData = myOrganization->getAllCompartmentData();
          } catch (OrganizationSrvException &e){
                  debugFFL << "getAllCompartmentData failed: " << e.what() <<
      endl;
                  DBUSPP_THROW_ERROR_EXCEPTION(ERROR_TOM_NOT_CONNECTED.c_str(), e
      .what());
          }
          CompartmentDataMap::iterator i;
          vector<dbuspp::fixed_array<unsigned char> > encCompData;
          for (i = compData.begin(); i != compData.end(); i++){
                  ByteVector bv = i->second.getEncodedData();
                  dbuspp::fixed_array<unsigned char> fa(bv.toCArray(), bv.size())
      ;
                  encCompData.push_back(fa);
          }
          debugFFL << "###leave###" << endl;
          return encCompData;
}
```

6.41.3.3 UInt32 OrganizationAdaptor::getConnectionStatus ( ) const

```
                                                    {
          debugFFL << "###enter###" << endl;
          UInt32 status = myOrganization->getConnectionStatus();
          debugFFL << "###leave###" << endl;
          return status;
}
```

6.41.3.4 dbuspp::fixed_array< unsigned char > OrganizationAdaptor::getDomainData (DomainID *domainID*)

```
              {
          debugFFL << "###enter###" << endl;
          ByteVector bv;
          try {
                  DomainData domData = myOrganization->getDomainData(domainID);
                  bv = domData.getEncodedData();
          }catch (OrganizationSrvException &e){
                  debugFFL << "getDomainData failed: " << e.what() << endl;
```

```
                DBUSPP_THROW_ERROR_EXCEPTION(ERROR_DOMAINDATA_NOT_FOUND.c_str()
      , e.what());
        }
        dbuspp::fixed_array<unsigned char> fa(bv.toCArray(), bv.size());
        debugFFL << "###leave###" << endl;
        return fa;
}
```

6.41.3.5 OrganizationID OrganizationAdaptor::getID ( ) const

```
                                                {
        debugFFL << "###enter###" << endl;
        OrganizationID id = myOrganization->getID();
        debugFFL << "###leave###" << endl;
        return id;
}
```

6.41.3.6 string OrganizationAdaptor::getName ( ) const

```
                                              {
        debugFFL << "###enter###" << endl;
        string name = myOrganization->getName();
        debugFFL << "###leave###" << endl;
        return name;
}
```

6.41.3.7 void OrganizationAdaptor::installCompartment (UInt32 *compartmentID*)

```
                                                                    {
        debugFFL << "###enter###" << endl;
        try {
                return myOrganization->installCompartment(compartmentID);
        }catch (OrganizationSrvException &e){
                debugFFL << "installCompartment failed: " << e.what() << endl;
                DBUSPP_THROW_ERROR_EXCEPTION(ERROR_COMPARTMENTDATA_NOT_FOUND.
      c_str(), e.what());
        }
        debugFFL << "###leave###" << endl;
}
```

6.41.3.8 void OrganizationAdaptor::installShare (const ShareInfo & *shareInfo*)

```
                                                                    {
        debugFFL << "###enter###" << endl;
        try {
                return myOrganization->installShareRequest(shareInfo.myName,
                                shareInfo.myURI,
                                static_cast<Share::Type>(shareInfo.myType),
                                shareInfo.myTypeID);
        }catch (OrganizationSrvException &e){
                debugFFL << "removeCompartment failed: " << e.what() << endl;
```

```
                DBUSPP_THROW_ERROR_EXCEPTION(ERROR_COMPARTMENTDATA_NOT_FOUND.
      c_str(), e.what());
        }
        debugFFL << "###leave###" << endl;
}
```

6.41.3.9 void OrganizationAdaptor::remove ( )

```
                                {
        debugFFL << "###enter###" << endl;
        myOrganization->remove();
        debugFFL << "###leave###" << endl;
}
```

6.41.3.10 void OrganizationAdaptor::removeCompartment (UInt32 *compartmentID*)

```
                                                                {
        debugFFL << "###enter###" << endl;
        try {
                return myOrganization->removeCompartment(compartmentID);
        }catch (OrganizationSrvException &e){
                debugFFL << "removeCompartment failed: " << e.what() << endl;
                DBUSPP_THROW_ERROR_EXCEPTION(ERROR_COMPARTMENTDATA_NOT_FOUND.
      c_str(), e.what());
        }
        debugFFL << "###leave###" << endl;
}
```

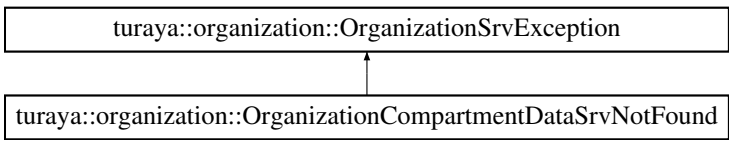
6.41.3.11 void OrganizationAdaptor::removeShare (UInt32 *shareID*)

```
                                                        {
        debugFFL << "###enter###" << endl;
        try {
                return myOrganization->removeShareRequest(shareID);
        }catch (OrganizationSrvException &e){
                debugFFL << "removeCompartment failed: " << e.what() << endl;
                DBUSPP_THROW_ERROR_EXCEPTION(ERROR_COMPARTMENTDATA_NOT_FOUND.
      c_str(), e.what());
        }
        debugFFL << "###leave###" << endl;
}
```

6.42 turaya::organization::OrganizationCompartmentDataSrvNotFound Class Reference

`#include <OrganizationExceptionsSrv.hxx>`

Inheritance diagram for turaya::organization::OrganizationCompartmentDataSrvNotFound:



Public Member Functions

- OrganizationCompartmentDataSrvNotFound (const std::string &context="-OrganizationCompartmentDataSrvNotFound")
- virtual ~OrganizationCompartmentDataSrvNotFound () throw ()

6.42.1 Constructor & Destructor Documentation

6.42.1.1 turaya::organization::OrganizationCompartmentDataSrvNotFound::-OrganizationCompartmentDataSrvNotFound (const std::string & *context* = "OrganizationCompartmentDataSrvNotFound") [inline]

```
:
OrganizationSrvException(context){};
```

6.42.1.2 virtual turaya::organization::OrganizationCompartmentDataSrvNotFound-::~OrganizationCompartmentDataSrvNotFound ( ) throw () [inline, virtual]

```
{};
```

6.43 turaya::organization::OrganizationManagerAdaptor Class - Reference

Adaptor class to make the OrganizationManagerSrv class accessible over DBus.

`#include <OrganizationManagerAdaptor.hxx>`

Public Types

- typedef std::map < OrganizationID, OrganizationAdaptor::Pointer > - OrganizationAdaptors

Public Member Functions

- OrganizationManagerAdaptor (const std::string &objectPath, dbuspp::service_-sptr spService)

- virtual ~OrganizationManagerAdaptor () throw ()
- std::vector< std::string > getAllOrganizations ()
- std::string getOrganization (const uint32_t &id)
- void installOrganization (dbuspp::fixed_array< unsigned char > dBusOrganizationData)
- void updateOrganization (dbuspp::fixed_array< unsigned char > dBusOrganizationData)
- void createOrganizationAdaptors ()

6.43.1 Detailed Description

Adaptor class to make the OrganizationManagerSrv class accessible over DBus.

6.43.2 Member Typedef Documentation

6.43.2.1 typedef std::map< OrganizationID, OrganizationAdaptor::Pointer > turaya::organization::OrganizationManagerAdaptor::OrganizationAdaptors

6.43.3 Constructor & Destructor Documentation

6.43.3.1 DBUSPP_INTERFACE_IMPLEMENT_END_MAP DBUSPP_OBJECT_IMPLEMENT_END_MAP OrganizationManagerAdaptor::OrganizationManagerAdaptor (const std::string & *objectPath*, dbuspp::service_sptr *spService*)

```
                                    :
        dbuspp::object(objectPath),
        myOrganizationAdaptors(),
        myOrganizationManagerSrv(),
        myService(spService),
        organizationInstalled(),
        organizationRemoved(){
        debugFFL << "###enter###" << endl;
        myOrganizationManagerSrv.signalOrganizationRemoving.Connect(this, &
      OrganizationManagerAdaptor::slotOrganizationRemoving);
        myOrganizationManagerSrv.signalOrganizationInstalled.Connect(this, &
      OrganizationManagerAdaptor::slotOrganizationInstalled);
        createOrganizationAdaptors();
        debugFFL << "###leave###" << endl;
}
```

6.43.3.2 OrganizationManagerAdaptor::~OrganizationManagerAdaptor ( ) throw () [virtual]

```
                                                                    {
        debugFFL << "###enter###" << endl;
        myOrganizationManagerSrv.signalOrganizationRemoving.Disconnect(this, &
      OrganizationManagerAdaptor::slotOrganizationRemoving);
        myOrganizationManagerSrv.signalOrganizationInstalled.Disconnect(this, &
      OrganizationManagerAdaptor::slotOrganizationInstalled);
        myOrganizationAdaptors.erase(myOrganizationAdaptors.begin(),
```

```
      myOrganizationAdaptors.end());
        debugFFL << "###leave###" << endl;
}
```

6.43.4 Member Function Documentation

6.43.4.1 void OrganizationManagerAdaptor::createOrganizationAdaptors ( )

```
                                                              {
        debugFFL << "###enter###" << endl;
        OrganizationManagerSrv::Organizations organizations =
      myOrganizationManagerSrv.getAllOrganizations();
        OrganizationManagerSrv::Organizations::iterator i;
        for(i = organizations.begin(); i != organizations.end(); i++){
                string path = createOrganizationPath(i->second->getID());
                debugFFL << "Create organizationsAdaptor for organizations with
       ID"
                                << i->second->getID() << "at path " << path <<
      endl;

                OrganizationAdaptor::Pointer orgAdapt(new OrganizationAdaptor(i
      ->second, path));
                orgAdapt->connect(myService);
                myOrganizationAdaptors.insert(pair<OrganizationID,
       OrganizationAdaptor::Pointer>(i->second->getID(), orgAdapt));
        }
        debugFFL << "###leave###" << endl;
}
```

6.43.4.2 std::vector< std::string > OrganizationManagerAdaptor::getAllOrganizations ( )

```
                                                                        {
        debugFFL << "###enter###" << endl;
        OrganizationAdaptors::iterator i;
        std::vector< std::string > paths;
        for(i = myOrganizationAdaptors.begin(); i != myOrganizationAdaptors.end
      (); i++){
                paths.push_back((*i).second->path());
        }
        debugFFL << "###leave###" << endl;
        return paths;

}
```

6.43.4.3 std::string OrganizationManagerAdaptor::getOrganization (const uint32_t & *id*)

```
                                                                          {
        debugFFL << "###enter###" << endl;

        OrganizationAdaptors::iterator i = myOrganizationAdaptors.find(id);
        if (i == myOrganizationAdaptors.end()){
                debugFFL << "OrganizationAdaptor with id " << id << " not found
```

```
      " << endl;
                DBUSPP_THROW_ERROR_INVALID_ARGS("wrong organization ID");
        }
        return i->second->path();
        debugFFL << "###leave###" << endl;
}
```

6.43.4.4 void OrganizationManagerAdaptor::installOrganization (dbuspp::fixed_array< unsigned char > *dBusOrganizationData*)

```
                                {
        debugFFL << "###enter###" << endl;
        ByteVector bv(dBusOrganizationData.get(),dBusOrganizationData.size());
        OrganizationData organizationData = OrganizationData(bv);

        try {
                myOrganizationManagerSrv.installOrganization(organizationData);
        }catch (OrganizationSrvAlreadyExists &e){
                debugFFL << "installOrganization failed: " << e.what() << endl;
                DBUSPP_THROW_ERROR_EXCEPTION(ERROR_ORGANIZATION_ALREADY_EXISTS.
      c_str(), e.what());
        }
        catch (OrganizationSrvInvalidOrganizationData &e){
                debugFFL << "installOrganization failed: " << e.what() << endl;
                DBUSPP_THROW_ERROR_EXCEPTION(ERROR_INVALID_ORGANIZATION_DATA.
      c_str(), e.what());
        }
        debugFFL << "###leave###" << endl;
}
```

6.43.4.5 void OrganizationManagerAdaptor::updateOrganization (dbuspp::fixed_array< unsigned char > *dBusOrganizationData*)

```
                                {
        debugFFL << "###enter###" << endl;
        ByteVector bv(dBusOrganizationData.get(),dBusOrganizationData.size());
        OrganizationData organizationData = OrganizationData(bv);

        try {
                myOrganizationManagerSrv.updateOrganization(organizationData);
        }catch (OrganizationSrvNotFound &e){
                debugFFL << "updateOrganization failed: " << e.what() << endl;
                DBUSPP_THROW_ERROR_EXCEPTION(ERROR_ORGANIZATION_NOT_FOUND.c_str
      (), e.what());
        }
        catch (OrganizationSrvInvalidOrganizationData &e){
                debugFFL << "installOrganization failed: " << e.what() << endl;
                DBUSPP_THROW_ERROR_EXCEPTION(ERROR_INVALID_ORGANIZATION_DATA.
      c_str(), e.what());
        }
        debugFFL << "###leave###" << endl;
}
```

6.44 unittests::OrganizationManagement_Test::OrganizationManagerObserver Class Reference

Public Member Functions

- void onOrganizationRemoved (OrganizationID iD)
- void onOrganizationInstalled (OrganizationID iD)
- void prepareWaitingForEvent ()
- void waitForEvent ()

6.44.1 Member Function Documentation

6.44.1.1 void unittests::OrganizationManagement_Test::OrganizationManagerObserver::onOrganizationInstalled (OrganizationID *iD*) [inline]

```
{
                                debugFFL << "enter" << endl;
                                myEvent = true;
                            }
```

6.44.1.2 void unittests::OrganizationManagement_Test::OrganizationManagerObserver::onOrganizationRemoved (OrganizationID *iD*) [inline]

```
                                                                    {
                                debugFFL << "enter" << endl;
                                myEvent = true;
                            }
```

6.44.1.3 void unittests::OrganizationManagement_Test::OrganizationManagerObserver::prepareWaitingForEvent ( ) [inline]

```
                                                    {
                                debugFFL << "enter" << endl;
                                myEvent = false;
                            }
```

6.44.1.4 void unittests::OrganizationManagement_Test::OrganizationManagerObserver::waitForEvent ( ) [inline]

```
                                          {
                                debugFFL << "enter" << endl;
                                while (!myEvent){
```

```
            //sleep(1);
        }
        debugFFL << "leave" << endl;
    }
```

6.45 turaya::organization::OrganizationManagerSrv Class Reference

Server side OrganizationManager representation.

`#include <OrganizationManagerSrv.hxx>`

Classes

- class **SetupShareManagerThread**
- class **TearDownShareManagerThread**

Public Types

- typedef std::map < OrganizationID, OrganizationSrv::Pointer > Organizations

Public Member Functions

- OrganizationManagerSrv ()
- virtual ~OrganizationManagerSrv () throw ()
- Organizations getAllOrganizations ()

  Returns all available Organizations.
- void installOrganization (OrganizationData organizationData)

  Install a new Organization.
- void updateOrganization (OrganizationData organizationData)

  Updates an existing Organization.

Public Attributes

- sirrix::utils::Signal1 < OrganizationSrv::Pointer > signalOrganizationInstalled

  Emitted when a new Organization is installed.
- sirrix::utils::Signal1 < turaya::OrganizationID > signalOrganizationRemoving

  Emitted on removement of a Organisation.

6.45.1 Detailed Description

Server side OrganizationManager representation.

6.45.2 Member Typedef Documentation

6.45.2.1 typedef std::map<OrganizationID, OrganizationSrv::Pointer > turaya::organization::OrganizationManagerSrv::Organizations

6.45.3 Constructor & Destructor Documentation

6.45.3.1 OrganizationManagerSrv::OrganizationManagerSrv ( )

```
                                                :
signalOrganizationInstalled(),
signalOrganizationRemoving(),
mySetupShareManagerThread(*this),
myTearDownShareManagerThread(*this),
myOrganizations(){
//      debugFFL << "enter" << endl;
        init(); //adopted from old libCompartment
        // enter the "poll user manager if a user is logged in and if, start
       the user manager" thread
        mySetupShareManagerThread.start();
        loadOrganizations();
//      debugFFL << "leave" << endl;
}
```

6.45.3.2 OrganizationManagerSrv::~OrganizationManagerSrv ( ) throw () [virtual]

```
                                                    {
//      debugFFL << "enter" << endl;
        debugFFL << "####Close Shares####" << endl;
        ShareManager& shareManager = CentralShareManager::getInstance();
        shareManager.unmountAllShares();
        debugFFL << "####Close Shares finished####" << endl;
        myOrganizations.erase(myOrganizations.begin(), myOrganizations.end());
//      debugFFL << "leave" << endl;
}
```

6.45.4 Member Function Documentation

6.45.4.1 OrganizationManagerSrv::Organizations OrganizationManagerSrv::getAllOrganizations ( )

Returns all available Organizations.

Returns

All Organizations.

```
       {
        return myOrganizations;
}
```

6.45.4.2 void OrganizationManagerSrv::installOrganization (OrganizationData *organizationData*)

Install a new Organization.

Parameters

organization-Data	OrganizationData describing the Organization to install

Exceptions

OrganizationSrv-AlreadyExists	if the Organization already exists
OrganizationSrv-InvalidOrganization-Data	if the given OrganizationData is invalid

```
        {

          debugFFL << "Installing new Organization with ID " << organizationData.
      getID() << endl;
          OrganizationSrv::Pointer newOrganization = internal_install(
      organizationData);
          signalOrganizationInstalled(newOrganization);
}
```

6.45.4.3 void OrganizationManagerSrv::updateOrganization (OrganizationData *organizationData*)

Updates an existing Organization.

Parameters

organization-Data	OrganizationData describing the Organization to update

Exceptions

OrganizationSrvNot-Found	if the Organization already exists
OrganizationSrv-InvalidOrganization-Data	if the given OrganizationData is invalid

```
        {
          Organizations::iterator i = myOrganizations.find(organizationData.getID
      ());
```

```
        if (i == myOrganizations.end()){
                throw OrganizationSrvNotFound();
        }else{
                debugFFL << "Updating Organization: " << i->first << endl;
                i->second->update(organizationData.getName(), organizationData.
      getTomData() );
                sirrix::encoding::Sequence signingChains = organizationData.
      getSigningChains();
                installSigningCertificateChains(signingChains);
        }
}
```

6.45.5 Member Data Documentation

6.45.5.1 sirrix::utils::Signal1<OrganizationSrv::Pointer> turaya::organization::-OrganizationManagerSrv::signalOrganizationInstalled

Emitted when a new Organization is installed.

6.45.5.2 sirrix::utils::Signal1<turaya::OrganizationID> turaya::organization::-OrganizationManagerSrv::signalOrganizationRemoving

Emitted on removement of a Organisation.

6.46 unittests::OrganizationManagement_Test::Organization-Observer Class Reference

Public Member Functions

- void prepareWaitingForEvent ()
- void waitForEvent ()

6.46.1 Member Function Documentation

6.46.1.1 void unittests::OrganizationManagement_Test::-OrganizationObserver::prepareWaitingForEvent ( ) [inline]

```
                                                {
                                        debugFFL << "enter" << endl;
                                        myEvent = false;
                                }
```

6.46.1.2 void unittests::OrganizationManagement_Test::OrganizationObserver-::waitForEvent ( ) [inline]

```
                                                {
```

```
        debugFFL << "enter" << endl;
        while (!myEvent){
                //sleep(1);
        }
        debugFFL << "leave" << endl;
}
```

6.47 turaya::organization::OrganizationSrv Class Reference

Server side organization representation.

`#include <OrganizationSrv.hxx>`

Public Types

- typedef sirrix::utils::SharedPointer < OrganizationSrv > Pointer
- typedef std::map< TomID, TOM::Pointer > Toms

Public Member Functions

- OrganizationSrv (const OrganizationID &organizationID)
- virtual ~OrganizationSrv ()
- CompartmentDataMap getAllCompartmentData ()

  returns all CompartmentData this organization provides
- Organization::ConnectionStatus getConnectionStatus () const

  returns the current status of the connection to a TOM
- void installCompartment (CompartmentID compartmentID)

  triggers the TOM to initiate a compartment installation
- void removeCompartment (CompartmentID compartmentID)

  triggers the TOM to initiate a compartment removal
- void installShareRequest (std::string name, std::string uri, share::Share::Type type, UInt32 typeID=0)

  sends a share installation request to the TOM
- void removeShareRequest (share::ShareID shareID)

  sends a share removal request to the TOM
- DomainData getDomainData (DomainID domainID)

  returns the DomainData for the given DomainID
- OrganizationID getID () const

  returns the OrganizationID of this Organization
- UserData authenticateUser (std::string username, std::string password)

  forwards user credentials to TOM to perform authentication
- std::string getName () const

  returns the name of this Organization
- void remove ()

Uninstalls this Organization.

- void update (std::string name, std::vector< TOMData > toms)
- void downloadAndInstallVDIFile (CompartmentID id)

Public Attributes

- sirrix::utils::Signal1 < turaya::OrganizationID > signalRemoved

  Emitted on removement of this Organization.
- sirrix::utils::Signal1 < Organization::ConnectionStatus > signalConnectionStatusChanged

  Emitted when the status of the TOM connection changed.

6.47.1 Detailed Description

Server side organization representation.

6.47.2 Member Typedef Documentation

6.47.2.1 typedef sirrix::utils::SharedPointer< OrganizationSrv > turaya::organization::OrganizationSrv::Pointer

6.47.2.2 typedef std::map<TomID, TOM::Pointer> turaya::organization::OrganizationSrv::Toms

6.47.3 Constructor & Destructor Documentation

6.47.3.1 OrganizationSrv::OrganizationSrv (const OrganizationID & *organizationID*)

```
                                                                    :
        signalRemoved(),
        signalConnectionStatusChanged(),
        myID(organizationID),
        myName(DataBaseUtils::getString(ORGANIZATIONTABLE, "name", myID)),
        myToms(),
        myConnectionWatchDogTrigger(1),
        myCompMgrObserver(*this){
        loadToms();
        start();
}
```

6.47.3.2 OrganizationSrv::~OrganizationSrv ( ) [virtual]

```
                                {
        debugFFL << "------enter" << endl;
        debugFFL << "------leave" << endl;
}
```

6.47.4 Member Function Documentation

6.47.4.1 UserData OrganizationSrv::authenticateUser (std::string *username*, std::string *password*)

forwards user credentials to TOM to perform authentication

Parameters

username	login name of user
password	password the user has entered

Returns

UserData object

```
                                                                        {
        for (Toms::iterator i = myToms.begin() ; i != myToms.end(); i++) {
                if (i->second->getStatus() == TOM::connected) {
                        return i->second->authenticateUser(username, password);
                }
        }
        throw OrganizationSrvNoTOMConnection();
}
```

6.47.4.2 void OrganizationSrv::downloadAndInstallVDIFile (CompartmentID *id*)

```
                                                                {

        TOM::Pointer tom;
        for (Toms::iterator i = myToms.begin(); i != myToms.end(); i++){
                if (i->second->getStatus() == TOM::connected){
                        tom = i->second;
                }
        }

        if ( tom.isNull() ){
                throw OrganizationSrvNoTOMConnection();
        }

        Path vDI_FilePath = tom->downloadVDIFile(id);
        try {
                CompartmentManager& comMgr = CompartmentManager::getInstance(
      CentralTrustedServer::getInstance());
                Compartment comp = comMgr.getCompartment(id);
                comp.setVirtualDiskImage(vDI_FilePath);
        } catch (CompartmentException &e){
                debugFFL << "Exception while setting virtual disk image
       occurred :" << e.what() << endl;
                throw OrganizationSrvException(e.what());
        }
}
```

6.47.4.3 CompartmentDataMap OrganizationSrv::getAllCompartmentData ( )

returns all CompartmentData this organization provides

Returns

map of CompartmentData

```
                                                        {
        for (Toms::iterator i = myToms.begin(); i != myToms.end(); i++){
                if (i->second->getStatus() == TOM::connected){
                        return i->second->getAllCompartmentData();
                }
        }
        throw OrganizationSrvNoTOMConnection();
}
```

6.47.4.4 Organization::ConnectionStatus OrganizationSrv::getConnectionStatus ( ) const

returns the current status of the connection to a TOM

Returns

ConnectionStatus

```
                                                                {
        for (Toms::const_iterator i = myToms.begin(); i != myToms.end(); i++){
                if (i->second->getStatus() == TOM::connected){
                        return Organization::Connected;
                }
        }
        return Organization::Disconnected;
}
```

6.47.4.5 DomainData OrganizationSrv::getDomainData (DomainID *domainID*)

returns the DomainData for the given DomainID

Parameters

domainID	DomainID identifying the DomainData to return

Exceptions

OrganizationSrv-DomaindDataNot-Found	if the given DomainID is invalid

```
                                                        {
```

```
        debugFFL << "enter" << endl;
        for (Toms::iterator i = myToms.begin(); i != myToms.end(); i++){
                if (i->second->getStatus() == TOM::connected){
                        return i->second->getDomainData(domainID);
                }
        }
        throw OrganizationSrvNoTOMConnection();
}
```

6.47.4.6 OrganizationID OrganizationSrv::getID ( ) const

returns the OrganizationID of this Organization

Returns

OrganizationID

```
                                                {
        return myID;
}
```

6.47.4.7 string OrganizationSrv::getName ( ) const

returns the name of this Organization

Returns

string

```
                                              {
        return myName;
}
```

6.47.4.8 void OrganizationSrv::installCompartment (CompartmentID *compartmentID*)

triggers the TOM to initiate a compartment installation

Parameters

compartmentID	ID of the CompartmentData describing the Compartment to install

```
                                                                    {
        for (Toms::iterator i = myToms.begin(); i != myToms.end(); i++){
                if (i->second->getStatus() == TOM::connected){
                        i->second->installCompartmentRequest(compartmentID);
                        return;
```

```
                }
        }
        throw OrganizationSrvNoTOMConnection();
}
```

6.47.4.9 void OrganizationSrv::installShareRequest (std::string *name*, std::string *uri*, share::Share::Type *type*, UInt32 *typeID* = 0)

sends a share installation request to the TOM

Parameters

name	the name of the new share
uri	the Uniform Resource Identifier of the new share
type	type of the new share (Appliance, Compartment or Domain)
typeID	if type is Compartment or Domain the ID of the appropriate Domain or Compartment object

```
                                                        {
        for (Toms::iterator i = myToms.begin(); i != myToms.end(); i++){
                if (i->second->getStatus() == TOM::connected){
                        i->second->installShareRequest(name, uri, type, typeID)
    ;
                        return;
                }
        }
        throw OrganizationSrvNoTOMConnection();
}
```

6.47.4.10 void OrganizationSrv::remove ( )

Uninstalls this Organization.

```
                            {
        debugFFL << "------enter" << endl;
        signalRemoved(myID);
        debugFFL << "------leave" << endl;
}
```

6.47.4.11 void OrganizationSrv::removeCompartment (CompartmentID *compartmentID*)

triggers the TOM to initiate a compartment removal

Parameters

compartment-ID	ID of the CompartmentData describing the Compartment to remove

```
                                                                        {
        for (Toms::iterator i = myToms.begin(); i != myToms.end(); i++){
                if (i->second->getStatus() == TOM::connected){
                        i->second->removeCompartmentRequest(compartmentID);
                        return;
                }
        }
        throw OrganizationSrvNoTOMConnection();
}
```

6.47.4.12 void OrganizationSrv::removeShareRequest (share::ShareID *shareID*)

sends a share removal request to the TOM

Parameters

shareID	ID of share to be removed

```
                                                                {
        for (Toms::iterator i = myToms.begin(); i != myToms.end(); i++){
                if (i->second->getStatus() == TOM::connected){
                        i->second->removeShareRequest(shareID);
                        return;
                }
        }
        throw OrganizationSrvNoTOMConnection();
}
```

6.47.4.13 void OrganizationSrv::update (std::string *name,* std::vector< TOMData > *toms*)

```
                                                                        {
        if (name == myName) {
                debugFFL << "Organization name" << myName << " remains
       unchanged." << endl;
        } else {
                debugFFL << "Updating Organization name from " << myName << "
       to " << name << endl;
                DataBaseUtils::setString(ORGANIZATIONTABLE, "name", myID, name)
      ;
                myName = name;
        }
        //update toms
        //first remove installed
        for (Toms::iterator installedTom_it = myToms.begin(); installedTom_it !
      = myToms.end();){
                bool found = false;
                for (std::vector<TOMData>::iterator newTOM_it = toms.begin();
      newTOM_it != toms.end(); newTOM_it++){
                        if ( installedTom_it->first == newTOM_it->getID() ){
                                found = true;
                        }
                }
                if (!found){
                        debugFFL << "Deleting TOM: " << installedTom_it->first
```

```
      << endl;
                        myToms.erase(installedTom_it++);
                }else{
                        ++installedTom_it;
                }
        }

        //update installed
        for (std::vector<TOMData>::iterator newTOM_it = toms.begin(); newTOM_it
       != toms.end(); newTOM_it++){
                Toms::iterator installedTom_it = myToms.find(newTOM_it->getID()
      );
                if (installedTom_it != myToms.end()){
                        debugFFL << "Updating TOM: " << installedTom_it->first
      << endl;
                        installedTom_it->second->update(*newTOM_it);
                }
        }
}
```

6.47.5 Member Data Documentation

6.47.5.1 sirrix::utils::Signal1<Organization::ConnectionStatus > turaya::organization::OrganizationSrv::signalConnectionStatusChanged

Emitted when the status of the TOM connection changed.

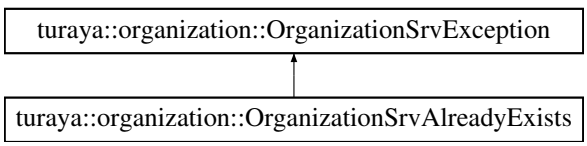
6.47.5.2 sirrix::utils::Signal1<turaya::OrganizationID> turaya::organization::-OrganizationSrv::signalRemoved

Emitted on removement of this Organization.

6.48 turaya::organization::OrganizationSrvAlreadyExists Class - Reference

`#include <OrganizationExceptionsSrv.hxx>`

Inheritance diagram for turaya::organization::OrganizationSrvAlreadyExists:



Public Member Functions

- OrganizationSrvAlreadyExists (const std::string &context="OrganizationSrv-AlreadyExists")

- virtual ~OrganizationSrvAlreadyExists () throw ()

6.48.1 Constructor & Destructor Documentation

6.48.1.1 turaya::organization::OrganizationSrvAlreadyExists::-OrganizationSrvAlreadyExists (const std::string & *context* = "OrganizationSrvAlreadyExists") [inline]

```
:
OrganizationSrvException(context){};
```

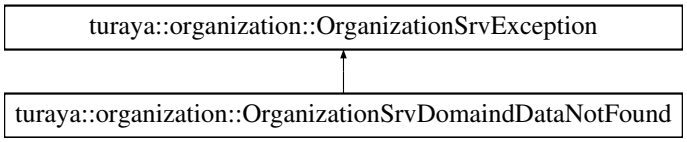
6.48.1.2 virtual turaya::organization::OrganizationSrvAlreadyExists-::~OrganizationSrvAlreadyExists ( ) throw () [inline, virtual]

```
{};
```

6.49 turaya::organization::OrganizationSrvDomaindDataNotFound Class Reference

```
#include <OrganizationExceptionsSrv.hxx>
```

Inheritance diagram for turaya::organization::OrganizationSrvDomaindDataNotFound:



Public Member Functions

- OrganizationSrvDomaindDataNotFound (const std::string &context="Organization-SrvDomaindDataNotFound")
- virtual ~OrganizationSrvDomaindDataNotFound () throw ()

6.49.1 Constructor & Destructor Documentation

6.49.1.1 turaya::organization::OrganizationSrvDomaindDataNotFound::-OrganizationSrvDomaindDataNotFound (const std::string & *context* = "OrganizationSrvDomaindDataNotFound") [inline]

```
:
OrganizationSrvException(context){};
```

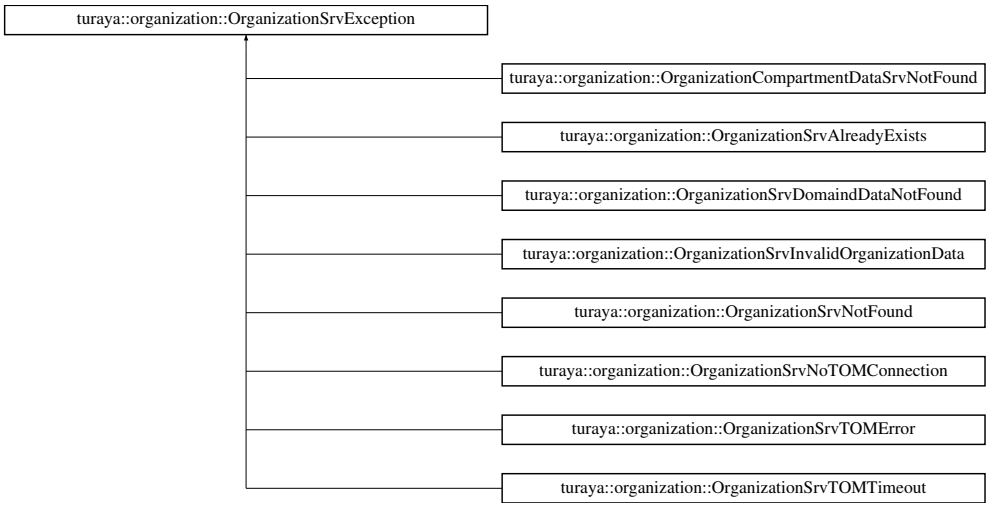
6.49.1.2 virtual turaya::organization::OrganizationSrvDomaindDataNotFound-::~OrganizationSrvDomaindDataNotFound ( ) throw () [inline, virtual]

```
{};
```

6.50 turaya::organization::OrganizationSrvException Class Reference

```
#include <OrganizationExceptionsSrv.hxx>
```

Inheritance diagram for turaya::organization::OrganizationSrvException:



Public Member Functions

- OrganizationSrvException (const std::string &context)
- virtual ~OrganizationSrvException () throw ()

6.50.1 Constructor & Destructor Documentation

6.50.1.1 turaya::organization::OrganizationSrvException::-OrganizationSrvException (const std::string & *context*) [inline]

```
                                                                :
        std::runtime_error(context) {};
```

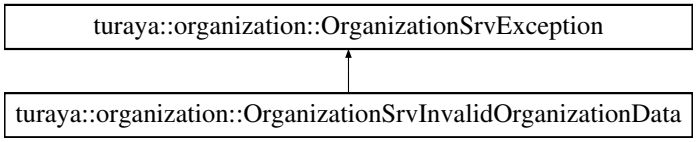
6.50.1.2 virtual turaya::organization::OrganizationSrvException-::~OrganizationSrvException ( ) throw () [inline, virtual]

```
{};
```

6.51 turaya::organization::OrganizationSrvInvalidOrganization-Data Class Reference

```
#include <OrganizationExceptionsSrv.hxx>
```

Inheritance diagram for turaya::organization::OrganizationSrvInvalidOrganizationData:



Public Member Functions

- OrganizationSrvInvalidOrganizationData (const std::string &context="Organization-SrvInvalidOrganizationData")
- virtual ~OrganizationSrvInvalidOrganizationData () throw ()

6.51.1 Constructor & Destructor Documentation

6.51.1.1 turaya::organization::OrganizationSrvInvalidOrganizationData::-OrganizationSrvInvalidOrganizationData (const std::string & *context* = "OrganizationSrvInvalidOrganizationData") [inline]

```
                                        :
                    OrganizationSrvException(context){};
```

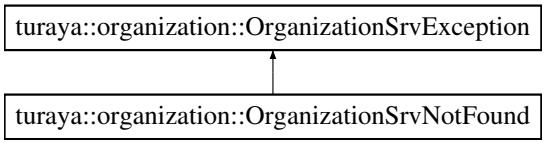
6.51.1.2 virtual turaya::organization::OrganizationSrvInvalidOrganizationData-::~OrganizationSrvInvalidOrganizationData ( ) throw () [inline, virtual]

```
{};
```

6.52 turaya::organization::OrganizationSrvNotFound Class Reference

`#include <OrganizationExceptionsSrv.hxx>`

Inheritance diagram for turaya::organization::OrganizationSrvNotFound:



Public Member Functions

- OrganizationSrvNotFound (const std::string &context="OrganizationSrvNotFound")
- virtual ~OrganizationSrvNotFound () throw ()

6.52.1 Constructor & Destructor Documentation

6.52.1.1 turaya::organization::OrganizationSrvNotFound::OrganizationSrvNotFound (const std::string & *context* = "OrganizationSrvNotFound") [inline]

```
:
   OrganizationSrvException(context){};
```

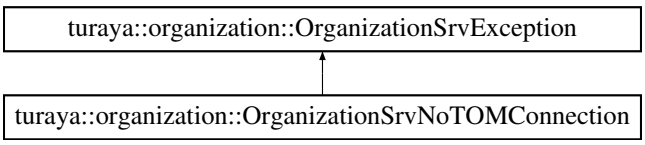
6.52.1.2 virtual turaya::organization::OrganizationSrvNotFound::~OrganizationSrvNotFound ( ) throw () [inline, virtual]

```
{};
```

6.53 turaya::organization::OrganizationSrvNoTOMConnection Class Reference

`#include <OrganizationExceptionsSrv.hxx>`

Inheritance diagram for turaya::organization::OrganizationSrvNoTOMConnection:



Public Member Functions

- OrganizationSrvNoTOMConnection (const std::string &context="OrganizationSrvNoTOMConnection")
- virtual ~OrganizationSrvNoTOMConnection () throw ()

6.53.1 Constructor & Destructor Documentation

6.53.1.1 turaya::organization::OrganizationSrvNoTOMConnection::OrganizationSrvNoTOMConnection (const std::string & *context* = "OrganizationSrvNoTOMConnection") [inline]

```
:
OrganizationSrvException(context){};
```

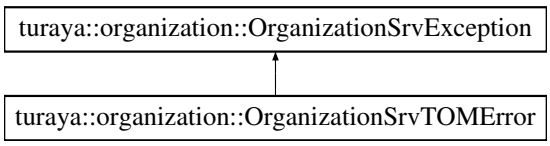
6.53.1.2 virtual turaya::organization::OrganizationSrvNoTOMConnection::~OrganizationSrvNoTOMConnection ( ) throw () [inline, virtual]

```
{};
```

6.54 turaya::organization::OrganizationSrvTOMError Class Reference

```
#include <OrganizationExceptionsSrv.hxx>
```

Inheritance diagram for turaya::organization::OrganizationSrvTOMError:



Public Member Functions

- OrganizationSrvTOMError (const std::string &context="OrganizationSrvTOMError")

- virtual ~OrganizationSrvTOMError () throw ()

6.54.1 Constructor & Destructor Documentation

6.54.1.1 turaya::organization::OrganizationSrvTOMError::OrganizationSrvTOMError (const std::string & *context* = "OrganizationSrvTOMError") [inline]

```
:
    OrganizationSrvException(context){};
```

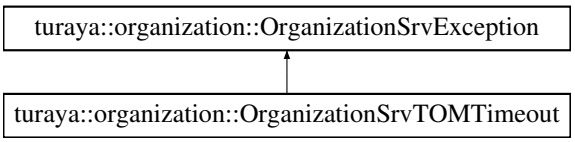
6.54.1.2 virtual turaya::organization::OrganizationSrvTOMError::~OrganizationSrvTOMError ( ) throw () [inline, virtual]

```
{};
```

6.55 turaya::organization::OrganizationSrvTOMTimeout Class Reference

```
#include <OrganizationExceptionsSrv.hxx>
```

Inheritance diagram for turaya::organization::OrganizationSrvTOMTimeout:



Public Member Functions

- OrganizationSrvTOMTimeout (const std::string &context="OrganizationSrvTOMTimeout")
- virtual ~OrganizationSrvTOMTimeout () throw ()

6.55.1 Constructor & Destructor Documentation

6.55.1.1 turaya::organization::OrganizationSrvTOMTimeout::OrganizationSrvTOMTimeout (const std::string & *context* = "OrganizationSrvTOMTimeout") [inline]

```
:
    OrganizationSrvException(context){};
```

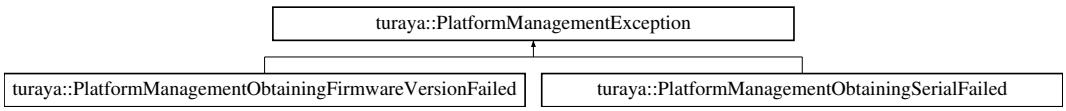
6.55.1.2 virtual turaya::organization::OrganizationSrvTOMTimeout-::~OrganizationSrvTOMTimeout ( ) throw () [inline, virtual]

```
{};
```

6.56 turaya::PlatformManagementException Class Reference

```
#include <PlatformManagementExceptions.hxx>
```

Inheritance diagram for turaya::PlatformManagementException:



Public Member Functions

- PlatformManagementException (const string &context)
- virtual ~PlatformManagementException () throw ()

6.56.1 Constructor & Destructor Documentation

6.56.1.1 turaya::PlatformManagementException::PlatformManagementException (const string & *context*) [inline]

```
                                                                    :
        runtime_error(context) {};
```

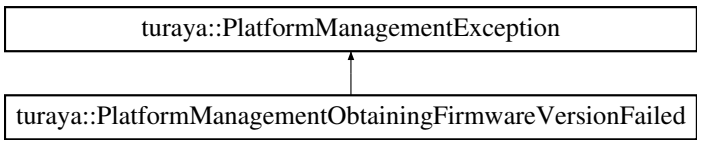
6.56.1.2 virtual turaya::PlatformManagementException::~-PlatformManagementException ( ) throw () [inline, virtual]

```
{};
```

6.57 turaya::PlatformManagementObtainingFirmwareVersion-Failed Class Reference

```
#include <PlatformManagementExceptions.hxx>
```

Inheritance diagram for turaya::PlatformManagementObtainingFirmwareVersionFailed:



Public Member Functions

- PlatformManagementObtainingFirmwareVersionFailed (const string &context="-PlatformManagementObtainingFirmwareVersionFailed")
- virtual ~PlatformManagementObtainingFirmwareVersionFailed () throw ()

6.57.1 Constructor & Destructor Documentation

6.57.1.1 turaya::PlatformManagementObtainingFirmwareVersionFailed::-PlatformManagementObtainingFirmwareVersionFailed (const string & *context* = "PlatformManagementObtainingFirmwareVersion-Failed") [inline]

:

```
PlatformManagementException(context){};
```

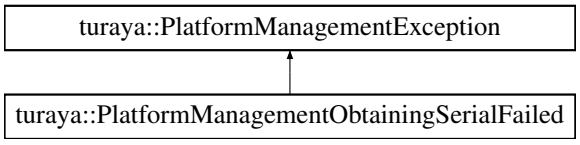
6.57.1.2 virtual turaya::PlatformManagementObtainingFirmwareVersionFailed-::~PlatformManagementObtainingFirmwareVersionFailed ( ) throw () [inline, virtual]

```
{};
```

6.58 turaya::PlatformManagementObtainingSerialFailed Class - Reference

```
#include <PlatformManagementExceptions.hxx>
```

Inheritance diagram for turaya::PlatformManagementObtainingSerialFailed:



Public Member Functions

- PlatformManagementObtainingSerialFailed (const string &context="PlatformManagementObtainingSerialFailed")
- virtual ~PlatformManagementObtainingSerialFailed () throw ()

6.58.1 Constructor & Destructor Documentation

6.58.1.1 turaya::PlatformManagementObtainingSerialFailed::PlatformManagementObtainingSerialFailed (const string & *context* = "PlatformManagementObtainingSerialFailed") [inline]

```
                                                :
    PlatformManagementException(context){};
```

6.58.1.2 virtual turaya::PlatformManagementObtainingSerialFailed::~PlatformManagementObtainingSerialFailed ( ) throw () [inline, virtual]

```
{};
```

6.59 turaya::tcd2::PluginNotFoundException Class Reference

```
#include <PluginNotFoundException.hxx>
```

Public Member Functions

- PluginNotFoundException (const string &message)

6.59.1 Constructor & Destructor Documentation

6.59.1.1 turaya::tcd2::PluginNotFoundException::PluginNotFoundException (const string & *message*) [inline]

```
: std::runtime_error(string("The requested TCD2 plugin was not found: ").append
      (message)) { };
```

6.60 turaya::compartment::SambaAccess Class Reference

```
#include <SambaAccess.hxx>
```

Public Member Functions

- SambaAccess ()
- virtual ~SambaAccess ()
- void getFile (const std::string &smburl, const sirrix::os::Path &targetPath, sirrix::utils::Delegate1< UInt64 > delegate=sirrix::utils::Delegate1< UInt64 >())
- void setFile (const sirrix::os::Path &sourcePath, const std::string &smburl, sirrix::utils::Delegate1< UInt64 > delegate=sirrix::utils::Delegate1< UInt64 >())
- UInt64 getFileSize (const std::string &smburl)

Static Public Member Functions

- static void no_auth_data_fn (const char *pServer, const char *pShare, char *pWorkgroup, int maxLenWorkgroup, char *pUsername, int maxLenUsername, char *pPassword, int maxLenPassword)

6.60.1 Constructor & Destructor Documentation

6.60.1.1 SambaAccess::SambaAccess ( )

```
                        {
        int result = 0;
        result = smbc_init( &SambaAccess::no_auth_data_fn, 0);
        if (result < 0){
                int errsv = errno;
                if (errsv == ENOMEM){
                        throw SambaException("smbc_init: Out of memory");
                }else if (errsv == ENOENT ){
             throw SambaException("smbc_init: The smb.conf file would not load"
    );
                }else{
                        throw SambaException("smbc_init: unknown error");
                }
        }
}
```

6.60.1.2 SambaAccess::~SambaAccess ( ) [virtual]

```
                         {

}
```

6.60.2 Member Function Documentation

6.60.2.1 void SambaAccess::getFile (const std::string & *smburl,* const sirrix::os::Path & *targetPath,* sirrix::utils::Delegate1< UInt64 > *delegate* = sirrix::utils::Delegate1<UInt64>())

```
                                          {
```

```
        static const ssize_t BSIZE = 64*1024;
        int f_dest, f_source;
        UInt64 bytesCopied = 0;
        ssize_t bytesRead, bytesWritten;

        f_source = smbOpen(smburl, O_RDONLY, 0);
        try {
                f_dest = openfile(targetPath,  O_WRONLY | O_CREAT | O_TRUNC, 06
      44);
        } catch (SambaException &e) {
                smbc_close (f_source);
                throw;
        }

        // register slot for signalling progress
        sirrix::utils::Signal1<UInt64> copyProgress;
        if (delegate != sirrix::utils::Delegate1<UInt64>()){
                copyProgress.Connect( delegate );
        }

        char* buffer = new char[BSIZE];
        while((bytesRead = smbc_read(f_source, buffer, BSIZE)) > 0) {
                bytesWritten = write(f_dest, buffer, bytesRead);
                if(bytesWritten == -1) {
                        int errsv = errno;
                        close(f_dest);
                        smbc_close (f_source);
                        delete[] buffer;
                        stringstream sstr;
                        sstr << "Could not write target file " << targetPath.
      getPath() << ": " << strerror(errsv) << endl;
                        throw SambaException(sstr.str());
                } else if( bytesWritten != bytesRead ) {
                        close(f_dest);
                        smbc_close (f_source);
                        delete[] buffer;
                        stringstream sstr;
                        sstr << "Read " << bytesRead << " bytes, but only " <<
      bytesWritten << " bytes could be written." << endl;
                        throw SambaException(sstr.str());
                }
                bytesCopied += bytesWritten;
                copyProgress( bytesCopied );
        }
        delete[] buffer;
        close(f_dest);
        smbClose(f_source);
}
```

6.60.2.2 UInt64 SambaAccess::getFileSize (const std::string & *smburl*)

```
                                                        {
        struct stat st;
        memset(&st, 0, sizeof(struct stat));
        int result = smbc_stat  (smburl.c_str(), &st);
        if (result < 0){
                int errsv = errno;
            if (errsv == ENOENT) {
                throw SambaException ("smbc_stat: A component of the path
       file_name does not exist.");
```

```
            } else if (errsv == EINVAL) {
                throw SambaException ("smbc_stat: a NULL url was passed or
       smbc_init not called.");
            } else if (errsv == EACCES) {
                throw SambaException ("smbc_stat: Permission denied.");
            } else if (errsv == ENOMEM) {
                throw SambaException("smbc_stat: Out of memory");
            } else if (errsv == ENOTDIR) {
                throw SambaException("smbc_stat: The target dir, url, is not a
       directory.");
            } else {
                throw SambaException("smbc_stat: Unknown error.");
            }
        }
        cout << "Size: " << st.st_size << endl;
        return static_cast<UInt64>( st.st_size );
}
```

6.60.2.3 void SambaAccess::no_auth_data_fn (const char * *pServer,* const char * *pShare,* char * *pWorkgroup,* int *maxLenWorkgroup,* char * *pUsername,* int *maxLenUsername,* char * *pPassword,* int *maxLenPassword*) [static]

```
                                    {
    return;
}
```

6.60.2.4 void SambaAccess::setFile (const sirrix::os::Path & *sourcePath,* const std::string & *smburl,* sirrix::utils::Delegate1< UInt64 > *delegate* = sirrix::utils::Delegate1<UInt64>())

```
                                                {
        static const ssize_t BSIZE = 64*1024;
        int f_dest, f_source;
        UInt64 bytesCopied = 0;
        ssize_t bytesRead, bytesWritten;

        f_source = openfile(sourcePath, O_RDONLY, 0);
        try {
                f_dest = smbOpen(smburl,  O_WRONLY | O_CREAT | O_TRUNC, 0644);
        } catch (SambaException &e) {
                close (f_source);
                throw;
        }

        // register slot for signalling progress
        sirrix::utils::Signal1<UInt64> copyProgress;
        if (delegate != sirrix::utils::Delegate1<UInt64>()){
                copyProgress.Connect( delegate );
        }

        char* buffer = new char[BSIZE];
        while((bytesRead = read(f_source, buffer, BSIZE)) > 0) {
                bytesWritten = smbc_write(f_dest, buffer, bytesRead);
                if(bytesWritten == -1) {
                        int errsv = errno;
                        close(f_source);
```

```
                        smbc_close (f_dest);
                        delete[] buffer;
                        stringstream sstr;
                        sstr << "Could not write target file " << smburl << ":
      " << strerror(errsv) << endl;
                        throw SambaException(sstr.str());
                } else if( bytesWritten != bytesRead ) {
                        close(f_source);
                        smbc_close (f_dest);
                        delete[] buffer;
                        stringstream sstr;
                        sstr << "Read " << bytesRead << " bytes, but only " <<
      bytesWritten << " bytes could be written." << endl;
                        throw SambaException(sstr.str());
                }
                bytesCopied += bytesWritten;
                copyProgress( bytesCopied );
          }
          delete[] buffer;
          close(f_source);
          smbClose(f_dest);
}
```

6.61 turaya::compartment::SambaException Class Reference

`#include <SambaAccess.hxx>`

Public Member Functions

- SambaException (const std::string &context)
- virtual ~SambaException () throw ()

6.61.1 Constructor & Destructor Documentation

6.61.1.1 turaya::compartment::SambaException::SambaException (const std::string & *context*) [inline]

```
                                                                    :
                                    runtime_error(context) {};
```

6.61.1.2 virtual turaya::compartment::SambaException::~SambaException ( ) throw () [inline, virtual]

```
{};
```

6.62 turaya::tcd2::TCDRootJob::ServerStateThread Class Reference

`#include <TCDRootJob.hxx>`

Public Member Functions

- ServerStateThread (TCDRootJob &tcdRootJob)
- void * run ()

6.62.1 Constructor & Destructor Documentation

6.62.1.1 TCDRootJob::ServerStateThread::ServerStateThread (TCDRootJob & *tcdRootJob*)

```
                                                                    :
      myTCDRootJob(tcdRootJob)
{
}
```

6.62.2 Member Function Documentation

6.62.2.1 void * TCDRootJob::ServerStateThread::run ( )

```
{
        try {
                while (1) {
                        InputByteVectorBinaryStream istream( myTCDRootJob.
      processIncomingMessage() );
                        AtomicValue av;
                        istream >> av;
                        Object serverState = av.get<Object> ();

                        myTCDRootJob.receiveServerState( serverState );
                }
        } catch ( exception &ex ) {
                debugFFL << "Exception in ServerStateThread has been thrown: "
      << ex.what() << endl;
                myTCDRootJob.fail();
        }

        return 0;
}
```

6.63 turaya::tcd2::TCDJobFactory Class Reference

```
#include <TCDJobFactory.hxx>
```

Public Member Functions

- TCDJobFactory ()
- virtual ~TCDJobFactory ()
- virtual TCDRootJob & createTCDRootJob (sirrix::dispatcher::Dispatcher &dispatcher, const sirrix::dispatcher::JobType &type, std::map< string, TCDPluginFactory * > &plugins)=0

- virtual void setTCDRootJob (TCDRootJob *tcdRootJob)
- virtual TCDRootJob * getTCDRootJob () const
- virtual std::string getModuleName () const =0
- virtual std::string getLongHelp () const
- virtual void setOptions (const std::vector< sirrix::utils::StringPair > &options)

Protected Attributes

- std::vector < sirrix::utils::StringPair > myOptions

6.63.1 Constructor & Destructor Documentation

6.63.1.1 TCDJobFactory::TCDJobFactory ( )

```
                                :
   myOptions(),
//   myPlugins(plugins),
   myTCDRootJob( 0 )
{}
```

6.63.1.2 TCDJobFactory::~TCDJobFactory ( ) [virtual]

```
                                {
}
```

6.63.2 Member Function Documentation

6.63.2.1 virtual TCDRootJob& turaya::tcd2::TCDJobFactory::createTCDRootJob (sirrix::dispatcher::Dispatcher & *dispatcher,* const sirrix::dispatcher::JobType & *type,* std::map< string, TCDPluginFactory * > & *plugins*) [pure virtual]

6.63.2.2 string TCDJobFactory::getLongHelp ( ) const [virtual]

Returns a help text about the options supported by this module.

```
                                          {
   return "Module '" + getModuleName() + "' Options:\n   No additional options.
      \n";
}
```

6.63.2.3 virtual std::string turaya::tcd2::TCDJobFactory::getModuleName ( ) const [pure virtual]

6.63.2.4 TCDRootJob * TCDJobFactory::getTCDRootJob ( ) const [virtual]

```
                                                {
        return myTCDRootJob;
}
```

6.63.2.5 void TCDJobFactory::setOptions (const std::vector< sirrix::utils::StringPair > & *options*) [virtual]

Set additional options to be used by the created factory.

```
        {
   myOptions = options;
}
```

6.63.2.6 void TCDJobFactory::setTCDRootJob (TCDRootJob * *tcdRootJob*) [virtual]

```
                                                        {
        myTCDRootJob = tcdRootJob;
}
```

6.63.3 Member Data Documentation

6.63.3.1 std::vector<sirrix::utils::StringPair> turaya::tcd2::TCDJobFactory::myOptions [protected]

6.64 turaya::tcd2::TCDPluginFactory Class Reference

```
#include <TCDPluginFactory.hxx>
```

Public Member Functions

- TCDPluginFactory ()
- virtual ~TCDPluginFactory ()
- virtual TCDPluginJob & createTCDPluginJob ()=0
- virtual void setTCDPluginJob (TCDPluginJob *tcdPluginJob)
- virtual TCDPluginJob * getTCDPluginJob () const
- virtual std::string getPluginName () const =0
- virtual std::string getLongHelp () const
- virtual void setOptions (const std::vector< sirrix::utils::StringPair > &options)

Protected Attributes

- std::vector < sirrix::utils::StringPair > myOptions

6.64.1 Constructor & Destructor Documentation

6.64.1.1 TCDPluginFactory::TCDPluginFactory ( )

:

```
   myOptions(),
   myTCDPluginJob( 0 )
{}
```

6.64.1.2 TCDPluginFactory::~TCDPluginFactory ( ) [virtual]

```
                                    {
}
```

6.64.2 Member Function Documentation

6.64.2.1 virtual TCDPluginJob& turaya::tcd2::TCDPluginFactory::createTCDPluginJob ( ) [pure virtual]

6.64.2.2 string TCDPluginFactory::getLongHelp ( ) const [virtual]

Returns a help text about the options supported by this module.

```
                                          {
   return "Plugin '" + getPluginName() + "' Options:\n   No additional options.
      \n";
}
```

6.64.2.3 virtual std::string turaya::tcd2::TCDPluginFactory::getPluginName ( ) const [pure virtual]

6.64.2.4 TCDPluginJob * TCDPluginFactory::getTCDPluginJob ( ) const [virtual]

```
                                                  {
        return myTCDPluginJob;
}
```

6.64.2.5 void TCDPluginFactory::setOptions (const std::vector< sirrix::utils::StringPair > & *options*) [virtual]

Set additional options to be used by the created factory.

```
         {
   myOptions = options;
}
```

6.64.2.6 void TCDPluginFactory::setTCDPluginJob (TCDPluginJob * *tcdPluginJob*) [virtual]

```
                                                        {
        myTCDPluginJob = tcdPluginJob;
}
```

6.64.3 Member Data Documentation

6.64.3.1 std::vector<sirrix::utils::StringPair> turaya::tcd2::TCDPluginFactory::myOptions [protected]

6.65 turaya::tcd2::TCDPluginJob Class Reference

`#include <TCDPluginJob.hxx>`

Public Member Functions

- TCDPluginJob (TCDPluginFactory &myFactory)
- virtual void runJob (const sirrix::dispatcher::JobType &type, sirrix::dispatcher::Job &parent, const sirrix::dispatcher::JobInstance &Instance)=0
- virtual ~TCDPluginJob ()

Protected Attributes

- TCDPluginFactory & myFactory

6.65.1 Constructor & Destructor Documentation

6.65.1.1 TCDPluginJob::TCDPluginJob (TCDPluginFactory & *myFactory*)

```
                                                        :
        myFactory(factory)
{
        myFactory.setTCDPluginJob(this);
}
```

6.65.1.2 TCDPluginJob::~TCDPluginJob ( ) [virtual]

```
                            {
        myFactory.setTCDPluginJob(0);
}
```

6.65.2 Member Function Documentation

6.65.2.1 virtual void turaya::tcd2::TCDPluginJob::runJob (const sirrix::dispatcher::JobType & *type,* sirrix::dispatcher::Job & *parent,* const sirrix::dispatcher::JobInstance & *Instance*) [pure virtual]

6.65.3 Member Data Documentation

6.65.3.1 TCDPluginFactory& turaya::tcd2::TCDPluginJob::myFactory [protected]

6.66 turaya::tcd2::TCDRootJob Class Reference

`#include <TCDRootJob.hxx>`

Classes

- class ClientStateThread
- class ServerStateThread

Public Member Functions

- TCDRootJob (sirrix::dispatcher::Dispatcher &dispatcher, const sirrix::dispatcher::JobType &type, TCDJobFactory &factory, std::map< string, TCDPluginFactory * > &plugins)
- virtual ~TCDRootJob ()
- virtual void open (const sirrix::dispatcher::JobType &method, const sirrix::dispatcher::JobInstance &childJob, const sirrix::utils::ByteVector &data)
- virtual void open (const sirrix::dispatcher::JobType &method, const sirrix::dispatcher::JobInstance &childJob)
- virtual void onConfiguration (const sirrix::utils::ByteVector &data)=0
- virtual void detachManager (const sirrix::utils::ByteVector &data)=0
- virtual void receiveServerHello ()=0
- virtual void extendClientHello ()=0
- virtual void receiveServerState (sirrix::encoding::Object &serverState)=0
- virtual sirrix::encoding::Object prepareClientState ()=0

Protected Member Functions

- void sendClientHello ()

Protected Attributes

- TCDJobFactory & myFactory
- std::map< string, TCDPluginFactory * > & myPlugins
- sirrix::encoding::Object clientHello
- ClientStateThread myClientStateThread
- ServerStateThread myServerStateThread

6.66.1 Constructor & Destructor Documentation

6.66.1.1 TCDRootJob::TCDRootJob (sirrix::dispatcher::Dispatcher & *dispatcher,* const sirrix::dispatcher::JobType & *type,* TCDJobFactory & *factory,* std::map< string, TCDPluginFactory * > & *plugins*)

```
        :
ThreadedJob(dispatcher, type, 0),
myFactory(factory),
myPlugins(plugins),
clientHello(),
myClientStateThread(*this),
myServerStateThread(*this)
{
myFactory.setTCDRootJob(this);

prepareClientHello();
}
```

6.66.1.2 TCDRootJob::~TCDRootJob ( ) [virtual]

```
                {
myFactory.setTCDRootJob(0);
}
```

6.66.2 Member Function Documentation

6.66.2.1 virtual void turaya::tcd2::TCDRootJob::detachManager (const sirrix::utils::ByteVector & *data*) [pure virtual]

6.66.2.2 virtual void turaya::tcd2::TCDRootJob::extendClientHello ( ) [pure virtual]

6.66.2.3 virtual void turaya::tcd2::TCDRootJob::onConfiguration (const sirrix::utils::ByteVector & *data*) [pure virtual]

6.66.2.4 virtual void turaya::tcd2::TCDRootJob::open (const sirrix::dispatcher::JobType & *method,* const sirrix::dispatcher::JobInstance & *childJob,* const sirrix::utils::ByteVector & *data*) [virtual]

6.66.2.5 virtual void turaya::tcd2::TCDRootJob::open (const sirrix::dispatcher::JobType & *method,* const sirrix::dispatcher::JobInstance & *childJob*) [virtual]

6.66.2.6 virtual sirrix::encoding::Object turaya::tcd2::TCDRootJob::prepareClientState ( ) [pure virtual]

6.66.2.7 virtual void turaya::tcd2::TCDRootJob::receiveServerHello ( ) [pure virtual]

6.66.2.8 virtual void turaya::tcd2::TCDRootJob::receiveServerState (sirrix::encoding::Object & *serverState*) [pure virtual]

6.66.2.9 void TCDRootJob::sendClientHello ( ) [protected]

```
{
        extendClientHello();

        Sequence features;
        features.append<string>("test1");
        features.append<string>("test2");
        features.append<string>("test3");
        features.append<string>("test4");

        clientHello.insert<Sequence> ( CB_SUPPORTED_FEATURES, features );

//      Object firmware = clientHello.get<Object> ( CB_CURRENT_FIRMWARE );
//      Sequence supFeatures = clientHello.get<Sequence> (
       CB_SUPPORTED_FEATURES );

//      debug << "Sending this ClientHello:\n FIRMWARE:\n  LABEL: " <<
       firmware.get<string>( CB_LABEL ) << "\n  REVISION: " << firmware.get<UInt32>(
       CB_REVISION ) << "\n  FIRMWARE: " << firmware.get<string>( CB_FIRMWARE ) << "\n  RELEA
       " << firmware.get<string>( CB_RELEASE ) << "\n SUPPORTTED_FEATURES:\n  " /*<<
       supFeatures*/ << endl;

        this->prepareOutgoingMessage( AtomicValue( clientHello ).getByteVector(
      ) );
        debug << "Sent ClientHello successfully!" << endl;
}
```

6.66.3 Member Data Documentation

6.66.3.1 sirrix::encoding::Object turaya::tcd2::TCDRootJob::clientHello [protected]

6.66.3.2 ClientStateThread turaya::tcd2::TCDRootJob::myClientStateThread [protected]

6.66.3.3 TCDJobFactory& turaya::tcd2::TCDRootJob::myFactory [protected]

6.66.3.4 std::map<string,TCDPluginFactory*>& turaya::tcd2::TCDRootJob::myPlugins [protected]

6.66.3.5 ServerStateThread turaya::tcd2::TCDRootJob::myServerStateThread [protected]

6.67 turaya::tcd2::TCReadThread Class Reference

`#include <TCReadThread.hxx>`

Public Member Functions

- TCReadThread (sirrix::dispatcher::Dispatcher &dispatcher, sirrix::tc::BinaryTrustedChannel &binaryStream)
- virtual ~TCReadThread ()
- void * run ()

6.67.1 Constructor & Destructor Documentation

6.67.1.1 TCReadThread::TCReadThread (sirrix::dispatcher::Dispatcher & *dispatcher*, sirrix::tc::BinaryTrustedChannel & *binaryStream*)

```
                :
        mDispatcher( dispatcher ),
        mBinaryStream( binaryStream ){
}
```

6.67.1.2 TCReadThread::~TCReadThread ( ) [virtual]

```
                    {
}
```

6.67.2 Member Function Documentation

6.67.2.1 void * TCReadThread::run ( )

```
                    {
    try {
        ByteVector dataReceive;
        Number num = 0;
        mBinaryStream >> num;
        dataReceive.resize( static_cast<UInt32>(num) );
        mBinaryStream >> dataReceive;
        mDispatcher.setIncomingMessage( dataReceive );
        dataReceive.clear();
        while ( 1 ) {
            Number num = 0;
            mBinaryStream >> num;
            dataReceive.resize( static_cast<UInt32>(num) );
            mBinaryStream >> dataReceive;
            mDispatcher.setIncomingMessage( dataReceive );
            dataReceive.clear();
        }
    } catch(exception &e) {
        if ( mBinaryStream.rdstate() & ios::eofbit ) {
            debugFFL << "eof: TOM has disconnected" << endl;
            ::exit(EXIT_SUCCESS);
        }
        mBinaryStream.close();
        debugFFL << "TOM has disconnected, read thread exception: " <<
  e.what() << endl;
    } catch( ... ) {
        debugFFL << "Unknown exception thrown." << endl;
```

```
                mBinaryStream.close();
                ::exit(EXIT_FAILURE);
        }
        return 0;
}
```

6.68 turaya::tcd2::TCWriteThread Class Reference

`#include <TCWriteThread.hxx>`

Public Member Functions

- TCWriteThread (sirrix::dispatcher::Dispatcher &dispatcher, sirrix::tc::BinaryTrustedChannel &binaryStream)
- virtual ~TCWriteThread ()
- void * run ()

6.68.1 Constructor & Destructor Documentation

6.68.1.1 TCWriteThread::TCWriteThread (sirrix::dispatcher::Dispatcher & *dispatcher*, sirrix::tc::BinaryTrustedChannel & *binaryStream*)

```
                                                                :
        mDispatcher( dispatcher ), mBinaryStream( binaryStream ) {
}
```

6.68.1.2 TCWriteThread::~TCWriteThread ( ) [virtual]

```
                                {
}
```

6.68.2 Member Function Documentation

6.68.2.1 void * TCWriteThread::run ( )

```
                        {

        try {
                OutputByteVectorBinaryStream os;
                debugFFL << "send CodeBook " << endl;
                os << mDispatcher.sendCodeBook();
                mBinaryStream << os.getByteVector() << flush;
                while ( 1 ) {
                        ByteVector dataSend = mDispatcher.getOutgoingMessage();
                        if ( dataSend.size() ) {
                                OutputByteVectorBinaryStream ostr;
                                ostr << Number( dataSend.size() );
                                ostr << dataSend;
```

```
                                debugFFL << "Write to binary trusted Channel of
        size " << dataSend.size() << endl;
                                mBinaryStream << ostr.getByteVector() << flush;
                        } else {
                                 debugFFL << "Error: Trying to send an empty
        message" << endl;
                        }
                }
         } catch ( exception &e ) {
                debugFFL<< "Write thread exception" << e.what() << endl;
                mBinaryStream.close();
                ::exit(EXIT_FAILURE);
         } catch( ... ) {
                debugFFL << "Unknown exception thrown." << endl;
                mBinaryStream.close();
                ::exit(EXIT_FAILURE);
         }
         return 0;
}
```

6.69 turaya::organization::TOM Class Reference

Server side TOM representation This Class encapsulates the CommunicationChannel (TrustedChannel) to the Trusted Object Manager server.

`#include <TOM.hxx>`

Public Types

- enum Status { disconnected, connecting, connected, disconnecting, connectionLost }
- typedef sirrix::utils::SharedPointer < TOM > Pointer

Public Member Functions

- TOM (TomID tomID)

  Creates a new TOM.
- virtual ~TOM ()
- std::string getIP () const

  returns the IP address of the TOM server
- TomID getTomID () const

  returns the ID of the TOM server
- Status getStatus () const
- void connect ()

  connects to TOM
- void disconnect ()

  disconnects from TOM
- sirrix::os::Path downloadVDIFile (CompartmentID id)

  Downloads the VDI File for the given compartment ID.

- CompartmentDataMap getAllCompartmentData ()

  returns all CompartmentData this Tom provides
- DomainData getDomainData (DomainID domainID)

  returns all DomainData for the given DomainID
- void installCompartmentRequest (CompartmentID compartmentID)

  sends a compartment installation request to the TOM
- void removeCompartmentRequest (CompartmentID compartmentID)

  sends a compartment removal request to the TOM
- void installShareRequest (std::string name, std::string uri, share::Share::Type type, UInt32 typeID=0)

  sends a share installation request to the TOM
- void removeShareRequest (share::ShareID shareID)

  sends a share removal request to the TOM
- UserData authenticateUser (std::string username, std::string password)

  send user credentials to TOM to perform authentication
- void update (TOMData data)

  updates this TOM

Public Attributes

- sirrix::utils::Signal0< void > signalConnectionLost

6.69.1 Detailed Description

Server side TOM representation This Class encapsulates the CommunicationChannel (TrustedChannel) to the Trusted Object Manager server.

6.69.2 Member Typedef Documentation

6.69.2.1 typedef sirrix::utils::SharedPointer< TOM > turaya::organization::TOM::Pointer

6.69.3 Member Enumeration Documentation

6.69.3.1 enum turaya::organization::TOM::Status

Enumerator:

disconnected

connecting

connected

disconnecting

connectionLost

```
{disconnected, connecting, connected, disconnecting, connectionLost} Status;
```

6.69.4 Constructor & Destructor Documentation

6.69.4.1 TOM::TOM (TomID *tomID*)

Creates a new TOM.

Parameters

tomID	ID of the TOM.

```
                :
          signalConnectionLost(),
          myID(tomID),
          myIP(DataBaseUtils::getString(TOMTABLE, IP_ADDRESS, myID)),
          myPort(PORT),
          myCAfilesDir(TOM_SIGNING_CHAINS_DIR),
          myPCRCertDir(PCR_CERTIFICATES_DIR),
          myCAfiles(os::FileManager::listFiles( myCAfilesDir )),
          myPCRCerts(os::FileManager::listFiles( myPCRCertDir )),
          myIdentityKeyNVRAMPosition(TPM_IDENT_KEY_NVRAM_INDEX),
          myTLSHandshakeKeyFile(TLH_HANDSHAKE_KEY_FILE),
          myTPMChannelConfig(/*TPM_IDENT_KEY_FILE*/
TPM_IDENT_KEY_NVRAM_INDEX, myTLSHandshakeKeyFile, myPCRCerts),
          myTrustedChannel(myTPMChannelConfig),
          myDispatcher(CentralCodeBook::getInstance().getEntry( "
tdDispatcher" ), Even),
          myApplianceRootJob(myDispatcher, CentralCodeBook::getInstance()
.getEntry("root"), myIP),
          myTDJobFactory(),
          myVPNJobFactory(),
          myReadThread(myDispatcher, myTrustedChannel),
          myWriteThread(myDispatcher, myTrustedChannel),
          myStatus(disconnected),
          myNetworkManager(NetworkManager::getInstance(
CentralTrustedServer::getInstance())){

  debugFFL << "enter" << endl;

  debugFFL << "Tom created ID: " << myID << " Host: " << myIP << ":" <<
myPort << endl <<
                  "CAfilesDir :" << myCAfilesDir << endl <<
                  "PCRCertDir: "<< myPCRCertDir << endl <<
                  "TPM Ident. key NVRAMPos: " <<
myIdentityKeyNVRAMPosition << endl <<
                  "TLS Handshake key: " << myTLSHandshakeKeyFile << endl;

  //register Module Job factories
  try {
          myApplianceRootJob.registerFactory(myTDJobFactory);
          myApplianceRootJob.registerFactory(myVPNJobFactory);
  } catch (AlreadyRegistered &e) {
          debugFFL << "Failed to Register module job : " << e.what() <<
endl;
  }

  myReadThread.signalConnectionLost.Connect(this, &
TOM::slotConnectionLost);
  myNetworkManager.connectionLost.Connect(this, &TOM::slotConnectionLost)
;
  //myApplianceRootJob.signalConnectionLost.Connect(this,
 &TOM::slotConnectionLost);
```

```
        //
      myDispatcher.signalInstallationProgressChanged.Connect(&signalInstallationProgressChang
        //              &sirrix::utils::Signal3<turaya::CompartmentID,
       turaya::CompartmentInstallationStatus, UInt32>::Emit);

        try {
                debugFFL << "registering ROOTJOB" << endl;
                myDispatcher.registerRootJob(myApplianceRootJob);
        }catch (exception &e){
                debugFFL << "Exception thrown during
       Dispatcher.registerRootJob() :" << e.what() << endl;
                myStatus = disconnected;
                return;
                //throw e;
        }

        debug << "-----CA Certificates------" << endl;
        for (std::vector<std::string>::const_iterator i = myCAfiles.begin(); i
      != myCAfiles.end(); i++){
                debug << *i << endl;
        }
        debug << endl;
        debug << "-----PCR Certificates------" << endl;
        if (myPCRCerts.size() == 0){
                debugFFL << "No PCR Certificates found" << endl;
        }
        for (std::vector<std::string>::const_iterator i = myPCRCerts.begin(); i
       != myPCRCerts.end(); i++){
                debug << *i << endl;
        }
        debug << endl;
}
```

6.69.4.2 TOM::~TOM ( ) [virtual]

```
          {
        if (myStatus == connected){
                disconnect();
        }
        myDispatcher.close(myApplianceRootJob);
}
```

6.69.5 Member Function Documentation

6.69.5.1 UserData TOM::authenticateUser (std::string *username*, std::string *password*)

send user credentials to TOM to perform authentication

Parameters

username	login name of user
password	password the user entered

Returns

UserData

```
                                                                {
  debugFFL << "User authentication triggered" << endl;

  TrustedServerJob& trustedDesktopJob = static_cast<TrustedServerJob&>(
myApplianceRootJob.getModuleJob("turaya"));

  AuthResultSharedPtr authResult(new AuthenticateUserResult);
  new AuthenticateUserJob(trustedDesktopJob, username, password,
authResult);

  Semaphore::Result waitResult = authResult->mySemaphore.timedwait(5000);
  if (waitResult == Semaphore::success){
          debugFFL << "Waiting for semaphore succeed, result: " <<
authResult->myUserData.getAuthenticationStatus() << endl;
  }else{
          debugFFL << "Timeout waiting for auth user result" << endl;
  }

  return authResult->myUserData;
}
```

6.69.5.2 void TOM::connect ( )

connects to TOM

```
           {
  myStatus = connecting;

  debugFFL << "Opening trustedChannel" << endl;
  try {
          myTrustedChannel.open(myIP, myPort, myCAfiles);
          debugFFL << "TrustedChannel connection established" << endl;
  }catch (exception &e){
          debugFFL << "Exception thrown during TrustedChannel.Open() :" <
< e.what() << endl;
          myStatus = disconnected;
          return;
  }
  myApplianceRootJob.start();
  myReadThread.start();
  myWriteThread.start();
  myStatus = connected;
}
```

6.69.5.3 void TOM::disconnect ( )

disconnects from TOM

```
             {
  myStatus = disconnecting;
  myWriteThread.stop();
```

```
        myReadThread.stop();
        debugFFL << "Waiting for ReadThread and WriteThread" << endl;
        try{
                myReadThread.join();
        }catch (ThreadError &e){
                debugFFL << "Exception occurred while joining ReadThread: " <<
      e.what() << endl;
        }
        try{
                myWriteThread.join();
        }catch (ThreadError &e){
                debugFFL << "Exception occurred while joining WriteThread: " <<
       e.what() << endl;
        }

        debugFFL << "ReadThread and WriteThread finished" << endl;
        myApplianceRootJob.stop();
        try{
                myApplianceRootJob.join();
        }catch (ThreadError &e){
                debugFFL << "Exception occurred while joining
       pplianceRootJob-Thread: " << e.what() << endl;
        }
        /* Close the channel */
        myTrustedChannel.close();
        myStatus = disconnected;
}
```

6.69.5.4 Path TOM::downloadVDIFile (CompartmentID *id*)

Downloads the VDI File for the given compartment ID.

```
                                        {
        TrustedServerJob& trustedDesktopJob = static_cast<TrustedServerJob&>(
      myApplianceRootJob.getModuleJob("turaya"));
        Path vdiFilePath;

        vdiFilePath = VDI_TEMP_DIR;

        try {
                //ensure that the directory exists
                if (!FileManager::isDir(vdiFilePath)){
                        debugFFL << "Directory "<< vdiFilePath.getPath() << "
       does not exist -> create it." << endl;
                        FileManager::createDir(vdiFilePath);
                        debugFFL << "Directory "<< vdiFilePath.getPath() << "
       created." << endl;
                }else{
                        debugFFL << "Directory "<< vdiFilePath.getPath() << "
       already exists." << endl;
                }
        } catch (FileManagerException &e){
                throw OrganizationSrvException(e.what());
        }
        stringstream sstr;
        sstr << "VDI_ID_" << id << ".vdi";
        vdiFilePath.addSubDir(sstr.str());
        if (FileManager::fileExists(vdiFilePath)) {
                FileManager::deleteFile(vdiFilePath);
```

```
        }

        debugFFL << "Using " << vdiFilePath.getPath() << " as temporary VDI
       file path" << endl;

        GetCompartmentJobResultSharedPtr result(new GetCompartmentJobResult);
        result->setFilePath(vdiFilePath);
        new GetCompartmentJob(trustedDesktopJob, id, result);
        result->waitUntilFinished();
        if (result->isSucceeded()){
                return vdiFilePath;
        }
        throw OrganizationSrvException("VDI Download failed");
}
```

6.69.5.5 CompartmentDataMap TOM::getAllCompartmentData ( )

returns all CompartmentData this Tom provides

Returns

map of CompartmentData

```
                                        {
  TrustedServerJob& trustedDesktopJob = static_cast<TrustedServerJob&>(
 myApplianceRootJob.getModuleJob("turaya"));
  GetAvailableCompartmentsResultSharedPtr result(new
 GetAvailableCompartmentsResult);
  new GetAvailableCompartmentsJob(trustedDesktopJob, result);
  debugFFL << "Waiting for GetAvailableCompartmentsJob to finish" << endl
 ;
  CompartmentDataMap compartmentData = result->getData();
  debugFFL << "GetAvailableCompartmentsJob  finished, received " <<
 compartmentData.size() << " CompartmentData." <<  endl;
  return compartmentData;
}
```

6.69.5.6 DomainData TOM::getDomainData (DomainID *domainID*)

returns all DomainData for the given DomainID

Returns

DomainData

```
                                        {
  debugFFL << "enter" << endl;
  TrustedServerJob& trustedDesktopJob = static_cast<TrustedServerJob&>(
 myApplianceRootJob.getModuleJob("turaya"));
  GetDomainDataResultSharedPtr result(new GetDomainDataResult);
  new GetDomainDataJob(trustedDesktopJob, domainID, result);
  DomainData receivedData = result->getData();
  return receivedData;
}
```

6.69.5.7 string TOM::getIP ( ) const

returns the IP address of the TOM server

Returns

IP Address as a string

```
                        {
        return myIP;
}
```

6.69.5.8 TOM::Status TOM::getStatus ( ) const

checks if the TOM is available

Returns

true if the TOM is available otherwise false

```
                              {
        return myStatus;
}
```

6.69.5.9 TomID TOM::getTomID ( ) const

returns the ID of the TOM server

Returns

TomID

```
                          {
        return myID;
}
```

6.69.5.10 void TOM::installCompartmentRequest (CompartmentID *compartmentID*)

sends a compartment installation request to the TOM

Parameters

compartmentID	ID of CompartmentData describing the Compartment to install

```
                                                    {
        TrustedServerJob& trustedDesktopJob = static_cast<TrustedServerJob&>(
```

```
      myApplianceRootJob.getModuleJob("turaya"));
        new CompartmentRequestJob(trustedDesktopJob, compartmentID,
      CompartmentRequestJob::install);
        debugFFL << "Compartment installation triggered." << endl;
}
```

6.69.5.11 void TOM::installShareRequest (std::string *name*, std::string *uri*, share::Share::Type *type*, UInt32 *typeID* = 0)

sends a share installation request to the TOM

Parameters

name	the name of the new share
uri	the Uniform Resource Identifier of the new share
type	type of the new share (Appliance, Compartment or Domain)
typeID	if type is Compartment or Domain the ID of the appropriate Domain or Compartment object

```
                                                                {
        TrustedServerJob& trustedDesktopJob = static_cast<TrustedServerJob&>(
      myApplianceRootJob.getModuleJob("turaya"));
        new ShareRequestJob(trustedDesktopJob,name,uri,type,typeID);
        debugFFL << "Share installation triggered." << endl;
}
```

6.69.5.12 void TOM::removeCompartmentRequest (CompartmentID *compartmentID*)

sends a compartment removal request to the TOM

Parameters

compartment-ID	ID of CompartmentData describing the Compartment to removed

```
                                                              {
        TrustedServerJob& trustedDesktopJob = static_cast<TrustedServerJob&>(
      myApplianceRootJob.getModuleJob("turaya"));
        new CompartmentRequestJob(trustedDesktopJob, compartmentID,
      CompartmentRequestJob::remove);
        debugFFL << "Compartment removal triggered." << endl;

}
```

6.69.5.13 void TOM::removeShareRequest (share::ShareID *shareID*)

sends a share removal request to the TOM

Parameters

shareID	ID of share to be removed

```
                                                {
  TrustedServerJob& trustedDesktopJob = static_cast<TrustedServerJob&>(
myApplianceRootJob.getModuleJob("turaya"));
  new ShareRequestJob(trustedDesktopJob, shareID);
  debugFFL << "Share removal triggered." << endl;
}
```

6.69.5.14 void TOM::update (TOMData *data*)

updates this TOM

Parameters

data	the new TOM data for this TOM

```
                                {
  if (data.getIP() == myIP){
          debugFFL << "Nothing to update for TOM "<<  myID <<" with IP "<
< myIP << endl;
  }else{
          debugFFL << "Updating TOM from IP: " << myIP << " to " << data.
getIP() << endl;
          DataBaseUtils::setString(TOMTABLE, IP_ADDRESS, myID, data.getIP
());
          myIP = data.getIP();
  }
}
```

6.69.6 Member Data Documentation

6.69.6.1 sirrix::utils::Signal0<void> turaya::organization::TOM::signalConnectionLost

6.70 turaya::tcd2::TrustedChannelDaemon2 Class Reference

```
#include <TrustedChannelDaemon2.hxx>
```

Public Types

- enum Status { connecting, connected, disconnecting, disconnected }

Public Member Functions

- TrustedChannelDaemon2 (sirrix::tc::ChannelConfig &config, TCDJobFactory &rootjobfactory, std::map< string, TCDPluginFactory * > &plugins)

- ~TrustedChannelDaemon2 ()

  Closes connection to management console.
- void connect (const std::string &hostname, UInt16 port, const std::vector< std::string > &caCerts)
- void disconnect ()

  Close connection to management console.
- Status getStatus () const

  returns the connection status of the TOM.
- string getError () const

6.70.1 Detailed Description

Encapsulation of the management protocol based on a trusted channel.

Note

It could make sense to use a binary channel reference instead of creating a trusted channel internally. This would make the TOM independent of the trusted channel (and more testable).

6.70.2 Member Enumeration Documentation

6.70.2.1 enum turaya::tcd2::TrustedChannelDaemon2::Status

Enumerator:

connecting

connected

disconnecting

disconnected

```
        {
    connecting,
    connected,
    disconnecting,
    disconnected,
};
```

6.70.3 Constructor & Destructor Documentation

6.70.3.1 TrustedChannelDaemon2::TrustedChannelDaemon2 (sirrix::tc::ChannelConfig & *config*, TCDJobFactory & *rootjobfactory*, std::map< string, TCDPluginFactory * > & *plugins*)

Creates a new TOM proxy.

Parameters

config	Configuration of the trusted channel.
rootjob	The root job that has to be used.

```
                                                                    :
   myStatus( disconnected ),
   myError( "noerror" ),
   myTrustedChannel( config ),
   myDispatcher( CB_TD_DISPATCHER, Even ),
   myTcdRootJob( rootjobfactory.createTCDRootJob( myDispatcher, CB_ROOT,
      plugins ) ),
   myReadThread( myDispatcher, myTrustedChannel ),
   myWriteThread( myDispatcher, myTrustedChannel )
{
   myDispatcher.registerRootJob( myTcdRootJob );
}
```

6.70.3.2 TrustedChannelDaemon2::~TrustedChannelDaemon2 ( )

Closes connection to management console.

```
{
   disconnect();
}
```

6.70.4 Member Function Documentation

6.70.4.1 void TrustedChannelDaemon2::connect (const std::string & *hostname*, UInt16 *port*, const std::vector< std::string > & *caCerts*)

Connect to management console.

Parameters

hostname	Hostname or IP address of management console.
port	Port of management console.
caCerts	Certificates to authenticate the TOM.

```
                                {
        myStatus = connecting;

        debug << "Starting trusted channel..." << endl;
        try {
                myTrustedChannel.open( hostname, port, caCerts );
        } catch (CException &e) {
                debug << "Connection to TOM was not possible." << e.what() <<
     endl;
                myError = "Connection to TOM was not possible.";
                return;
        } catch (...) {
                debug << "Unknown exception/error thrown!" << endl;
                myError = "Unknown error while connecting to TOM.";
                return;
```

```
        }
        debug << "TC opened!" << endl;
        try {
                debug << "Starting TCD2 root job..." << endl;
                myTcdRootJob.start();
                debug << "Starting read thread..." << endl;
                myReadThread.start();
                debug << "Starting write thread..." << endl;
                myWriteThread.start();

                myStatus = connected;

                myReadThread.join();
                myWriteThread.join();
   }
   catch (CException &e) {
                debug << "Exception geflogen1" << endl;
   }
   catch (...) {
                disconnect();
                throw;
   }

}
```

6.70.4.2 void TrustedChannelDaemon2::disconnect ( )

Close connection to management console.

```
                                        {
   if (myStatus == disconnected)
      return;

   myStatus = disconnecting;

   myWriteThread.stop();
   myReadThread.stop();

   myReadThread.join();
   myWriteThread.join();

   myReadThread.join();
   myWriteThread.join();

   myTcdRootJob.stop();
   myTcdRootJob.join();

   myTrustedChannel.close();

   myStatus = disconnected;
}
```

6.70.4.3 string turaya::tcd2::TrustedChannelDaemon2::getError ( ) const [inline]

```
                                                {
```

```
        return myError;
    }
```

6.70.4.4 TrustedChannelDaemon2::Status turaya::tcd2::TrustedChannelDaemon2::getStatus ( ) const [inline]

returns the connection status of the TOM.

```
                                                                    {
        return myStatus;
    }
```

6.71 turaya::TrustedDesktop Class Reference

Represents a concrete TrustedDesktop.

`#include <TrustedDesktop.hxx>`

Public Member Functions

- TrustedDesktop ()
- virtual ~TrustedDesktop ()
- PlatformID getPlatformID () const
- FirmwareVersion getFirmwareVersion () const
- dbuspp::dbus & getDBus ()

  Returns a reference to the dbus.

6.71.1 Detailed Description

Represents a concrete TrustedDesktop.

6.71.2 Constructor & Destructor Documentation

6.71.2.1 TrustedDesktop::TrustedDesktop ( )

```
                            :
            myID( loadPlatformID() ),
            myFirmwareVersion( loadFirmwareVersion() ) {
    debugFFL << "enter" << endl;
    debugFFL << "leave" << endl;
}
```

6.71.2.2 TrustedDesktop::~TrustedDesktop ( ) [virtual]

```
                                {
        debugFFL << "enter" << endl;
        debugFFL << "leave" << endl;
}
```

6.71.3 Member Function Documentation

6.71.3.1 dbuspp::dbus & TrustedDesktop::getDBus ( )

Returns a reference to the dbus.

```
                                        {
        return myDbus;
}
```

6.71.3.2 FirmwareVersion TrustedDesktop::getFirmwareVersion ( ) const

```
                                                        {
        return myFirmwareVersion;
}
```

6.71.3.3 PlatformID TrustedDesktop::getPlatformID ( ) const

```
                                                    {
        return myID;
}
```

6.72 turaya::user::UserAdaptor Class Reference

Adaptor class to make the UserSrv class accessible over DBus.

`#include <UserAdaptor.hxx>`

Public Types

- typedef sirrix::utils::SharedPointer < UserAdaptor > Pointer

Public Member Functions

- UserAdaptor (UserSrv::Pointer user, std::string dBusPath)
- virtual ~UserAdaptor () throw ()
- void update (dbuspp::fixed_array< unsigned char > dBusUserData)
- UserID getUserID () const

- std::string getUsername () const
- dbuspp::fixed_array< unsigned char > getUserData () const
- void remove ()
- UInt32 getStatus () const
- bool accountHasExpired () const
- virtual void onStatusChanged (User::Status status)

Public Attributes

- sirrix::utils::Signal1 < turaya::UserID > onUserAdaptorRemoved

6.72.1 Detailed Description

Adaptor class to make the UserSrv class accessible over DBus.

6.72.2 Member Typedef Documentation

6.72.2.1 typedef sirrix::utils::SharedPointer<UserAdaptor> turaya::user::UserAdaptor::Pointer

6.72.3 Constructor & Destructor Documentation

6.72.3.1 DBUSPP_INTERFACE_IMPLEMENT_END_MAP DBUSPP_OBJECT_IMPLEMENT_END_MAP UserAdaptor::UserAdaptor (UserSrv::Pointer *user*, std::string *dBusPath*)

:

```
dbuspp::object(dBusPath),
onUserAdaptorRemoved(),
myUser(user),
statusChanged() {
        debugFFL << "enter" << endl;
        myUser->onStatusChanged.Connect(this, &UserAdaptor::onStatusChanged);
        debugFFL << "leave" << endl;
}
```

6.72.3.2 UserAdaptor::~UserAdaptor ( ) throw () [virtual]

```
                                {
        debugFFL << "enter" << endl;
        myUser->onStatusChanged.Disconnect( this, &UserAdaptor::onStatusChanged
       );
        debugFFL << "leave" << endl;
}
```

6.72.4 Member Function Documentation

6.72.4.1 bool UserAdaptor::accountHasExpired ( ) const

```
                                    {
        return myUser->accountHasExpired();
}
```

6.72.4.2 UInt32 UserAdaptor::getStatus ( ) const

```
                                   {
        return myUser->getStatus();
}
```

6.72.4.3 dbuspp::fixed_array< unsigned char > UserAdaptor::getUserData ( ) const

```
                                                                    {
        ByteVector bv = ( myUser->getUserData() ).getEncodedData();
        dbuspp::fixed_array<unsigned char> fa( bv.toCArray(), bv.size() );

        return fa;
}
```

6.72.4.4 UserID UserAdaptor::getUserID ( ) const

```
                                   {
        return myUser->getUserID();
}
```

6.72.4.5 std::string UserAdaptor::getUsername ( ) const

```
                                     {
        return myUser->getUsername();
}
```

6.72.4.6 void UserAdaptor::onStatusChanged (User::Status *status*) [virtual]

```
                                                      {
        statusChanged( status );
}
```

6.72.4.7 void UserAdaptor::remove ( )

```
                     {
        myUser->remove();
}
```

6.72.4.8 void UserAdaptor::update (dbuspp::fixed_array< unsigned char > *dBusUserData*)

```
                                                                                {

        try {
                ByteVector bv( dBusUserData.get(), dBusUserData.size() );
                UserData userData( bv );
                myUser->update( userData );
        } catch( UserSrvInvalidUserData &e ) {
                debugFFL << "updateUser failed: " << e.what() << endl;
                DBUSPP_THROW_ERROR_EXCEPTION( ERROR_INVALID_USER_DATA.c_str(),
      e.what() );
        }
}
```

6.72.5 Member Data Documentation

6.72.5.1 sirrix::utils::Signal1<turaya::UserID> turaya::user::UserAdaptor::onUserAdaptorRemoved

6.73 turaya::user::UserHypervisor Class Reference

Abstraction of platform specific user functions.

`#include <UserHypervisor.hxx>`

Public Member Functions

- UserHypervisor ()
- virtual ~UserHypervisor () throw ()
- UserManagerSrv::UserSrvPtrs loadUsers (UserManagerSrv *userMgrSrvPtr)
- void installUser (UserData userData)

  Install a new User.
- void updateUser (UserData userData)

  Modify a User.
- void removeUser (std::string username)

  Remove a user.

6.73.1 Detailed Description

Abstraction of platform specific user functions.

6.73.2 Constructor & Destructor Documentation

6.73.2.1 UserHypervisor::UserHypervisor ( )

```
                                :
                myLockingMutex() {
}
```

6.73.2.2 UserHypervisor::~UserHypervisor ( ) throw () [virtual]

```
                                                {
}
```

6.73.3 Member Function Documentation

6.73.3.1 void UserHypervisor::installUser (UserData *userData*)

Install a new User.

Parameters

userData	UserData describing the User to install

Exceptions

UserSrvAlready-Exists	if the User already exists
UserSrvInvalidUser-Data	if the given UserData is invalid

```
                                                        {
        debugFFL << "enter update" << endl;
        ScopedMutex scopedMutex( myLockingMutex );
        updatePasswdFile( userData );
        updateShadowFile( userData );
        updateGroupFile( userData );
        addToLocalPasswordCache( userData );
}
```

6.73.3.2 UserManagerSrv::UserSrvPtrs UserHypervisor::loadUsers (UserManagerSrv * *userMgrSrvPtr*)

```
          {
        debugFFL << "enter" << endl;
        ifstream passwd( PATH_PASSWD.c_str() );
        ifstream shadow;
        vector < UserID > userIDs;
        UserManagerSrv::UserSrvPtrs userPtrs;

        try {
                debugFFL << "enter passwd loop" << endl;
                while( !passwd.eof() ) {
```

```
                    string passwdLine;
                    string username;
                    string password;
                    string userId;
                    string groupId;
                    string realname;
                    string homedir;
                    string shell;

                    // read one line from passwd file
                    std::getline( passwd, passwdLine, '\n' );

                    if( passwdLine.empty() ) {
                            continue;
                    }

                    stringstream strings( passwdLine );
                    std::getline( strings, username, ':' );
                    std::getline( strings, password, ':' );
                    std::getline( strings, userId, ':' );
                    std::getline( strings, groupId, ':' );
                    std::getline( strings, realname, ':' );
                    std::getline( strings, homedir, ':' );
                    std::getline( strings, shell, ':' );

                    stringstream uidstr;
                    uidstr << userId;
                    unsigned int uid;
                    uidstr >> uid;

                    stringstream gidstr;
                    gidstr << groupId;
                    unsigned int gid;
                    gidstr >> gid;

                    if( ( uid >= MIN_UID ) && ( uid <= MAX_UID ) ) {
                            debugFFL << "found user with ID " << uid << "
 in passwd" << endl;

                            shadow.open( _PATH_SHADOW );
                            debugFFL << "enter shadow loop" << endl;
                            while( !shadow.eof() ) {
                                    string shadowLine;
                                    string shadowUsername;
                                    string shadowPassword;
                                    string passwordLastChanged;
                                    string daysBeforePasswordCanChanged;
                                    string daysAfterPasswordMustChanged;
                                    string daysBeforePasswordWarning;
                                    string daysBeforeAccInact;
                                    string daysBeforeAccExpire;
                                    string flags;

                                    // read one line from shadow file
                                    std::getline( shadow, shadowLine, '\n'
);

                                    stringstream strings( shadowLine );
                                    std::getline( strings, shadowUsername,
':' );
                                    std::getline( strings, shadowPassword,
':' );
```

```
                                        std::getline( strings,
      passwordLastChanged, ':' );
                                        std::getline( strings,
      daysBeforePasswordCanChanged, ':' );
                                        std::getline( strings,
      daysAfterPasswordMustChanged, ':' );
                                        std::getline( strings,
      daysBeforePasswordWarning, ':' );
                                        std::getline( strings,
      daysBeforeAccInact, ':' );
                                        std::getline( strings,
      daysBeforeAccExpire, ':' );
                                        std::getline( strings, flags, ':' );

                                        //debugFFL << shadowUsername << " == "
       << username <<endl;
                                        if( shadowUsername == username ) {
                                                debugFFL << "found user with ID
       " << uid << " in shadow" << endl;
                                                time_t expiration;
                                                istringstream(
      daysBeforeAccExpire ) >> expiration;

                                                bool expirationIsGlobal;
                                                if( flags == "0" ) {
                                                        expirationIsGlobal =
      false;
                                                }else {
                                                        expirationIsGlobal =
      true;
                                                }

                                                UserSrv::Pointer userSrv(
                                                                new UserSrv(
      username, realname, uid, gid, expiration, expirationIsGlobal,
      UserData::ACCESS_DENIED ) );
                                                userSrv->signalUserRemoved.
      Connect( userMgrSrvPtr, &UserManagerSrv::slotUserRemoved );
                                                userPtrs.insert( pair<UserID,
       UserSrv::Pointer>( uid, userSrv ) );
                                        }
                                }
                                shadow.close();
                        }
                }
                debugFFL << "leave passwd loop" << endl;
        } catch( ios::failure& e ) {
                if( !passwd.eof() || !shadow.eof() ) {
                        debugFFL << "Exception: " << e.what();
                }
                passwd.close();
                shadow.close();
        }
        passwd.close();

        return userPtrs;
}
```

6.73.3.3 void UserHypervisor::removeUser (std::string *username*)

Remove a user.

Parameters

username	describing the User to remove

Exceptions

UserSrvNotFound	if User is already removed

```
                                                    {
        if( username != "turaya" ) {
                removeUserPasswd( username );
                removeUserShadow( username );
                removeUserGroup( username );

                tcd::Appliance::executeShellCommand( "/bin/rm -rf /home/" +
      username );
                tcd::Appliance::executeShellCommand( "/usr/bin/cc_test -update
       any " + username + " -" );
        }

}
```

6.73.3.4 void UserHypervisor::updateUser (UserData *userData*)

Modify a User.

Parameters

userData	UserData describing the User to install

Exceptions

UserSrvInvalidUserData	if the given UserData is invalid

```
                                                    {
        debugFFL << "enter update" << endl;
        ScopedMutex scopedMutex( myLockingMutex );
        updatePasswdFile( userData );
        updateShadowFile( userData );
        updateLocalPasswordCache( userData );
}
```

6.74 turaya::user::UserManagerAdaptor Class Reference

Adaptor class to make the UserManagerSrv class accessible over DBus.

```
#include <UserManagerAdaptor.hxx>
```

Public Types

- typedef std::map< UserID, UserAdaptor::Pointer > UserAdaptorPtrs

Public Member Functions

- UserManagerAdaptor (const std::string &objectPath, dbuspp::service_sptr spService)
- virtual ~UserManagerAdaptor () throw ()
- std::vector< std::string > getAllUsers ()
- bool hasUser (std::string username)
- void installUser (dbuspp::fixed_array< unsigned char > dBusUserData)
- void createUserAdaptors ()

6.74.1 Detailed Description

Adaptor class to make the UserManagerSrv class accessible over DBus.

6.74.2 Member Typedef Documentation

6.74.2.1 typedef std::map<UserID, UserAdaptor::Pointer> turaya::user::UserManagerAdaptor::UserAdaptorPtrs

6.74.3 Constructor & Destructor Documentation

6.74.3.1 DBUSPP_INTERFACE_IMPLEMENT_END_MAP DBUSPP_OBJECT_IMPLEMENT_END_MAP UserManagerAdaptor::UserManagerAdaptor (const std::string & *objectPath*, dbuspp::service_sptr *spService*)

```
                        :
dbuspp::object(objectPath),
myUserAdaptors(),
myUserManagerSrv(),
myService(spService),
userInstalled(),
userRemoved() {
        debugFFL << "enter" << endl;

        myUserManagerSrv.signalUserInstalled.Connect(this, &
      UserManagerAdaptor::slotUserInstalled);
        myUserManagerSrv.signalUserRemoved.Connect(this, &
      UserManagerAdaptor::slotUserRemoving);
        createUserAdaptors();
        debugFFL << "leave" << endl;
}
```

6.74.3.2 UserManagerAdaptor::~UserManagerAdaptor ( ) throw () [virtual]

```
                                                {
        debugFFL << "enter" << endl;
        myUserAdaptors.erase( myUserAdaptors.begin(), myUserAdaptors.end() );
        debugFFL << "leave" << endl;
}
```

6.74.4 Member Function Documentation

6.74.4.1 void UserManagerAdaptor::createUserAdaptors ( )

```
                                            {
        UserManagerSrv::UserSrvPtrs users = myUserManagerSrv.getAllUsers();

        UserManagerSrv::UserSrvPtrs::iterator i;
        for( i = users.begin(); i != users.end(); i++ ) {
                string path = createUserPath( i->second->getUserID() );
                debugFFL << "Create UserAdaptor for User with ID " << i->second
      ->getUserID() << " at path " << path << endl;

                UserAdaptor::Pointer userAdapt( new UserAdaptor( i->second,
      path ) );
                userAdapt->connect( myService );
                userAdapt->onUserAdaptorRemoved.Connect( this, &
      UserManagerAdaptor::slotUserRemoving );
                myUserAdaptors.insert( pair<UserID, UserAdaptor::Pointer>( i->
      second->getUserID(), userAdapt ) );
          }
}
```

6.74.4.2 std::vector< std::string > UserManagerAdaptor::getAllUsers ( )

```
                                                    {

        UserAdaptorPtrs::iterator i;
        std::vector < std::string > paths;
        for( i = myUserAdaptors.begin(); i != myUserAdaptors.end(); i++ ) {
                paths.push_back( ( *i ).second->path() );
        }
        return paths;
}
```

6.74.4.3 bool UserManagerAdaptor::hasUser (std::string *username*)

```
                                                    {
        return myUserManagerSrv.hasUser( username );
}
```

6.74.4.4 void UserManagerAdaptor::installUser (dbuspp::fixed_array< unsigned char > *dBusUserData*)

```
           {
```

```
        try {
                ByteVector bv( dBusUserData.get(), dBusUserData.size() );
                UserData userData = UserData( bv );
                myUserManagerSrv.installUser( userData );
        } catch( UserSrvAlreadyExists &e ) {
                debugFFL << "installUser failed: " << e.what() << endl;
                DBUSPP_THROW_ERROR_EXCEPTION( ERROR_USER_ALREADY_EXISTS.c_str()
      , e.what() );
        } catch( UserSrvInvalidUserData &e ) {
                debugFFL << "installUser failed: " << e.what() << endl;
                DBUSPP_THROW_ERROR_EXCEPTION( ERROR_INVALID_USER_DATA.c_str(),
      e.what() );
        }
}
```

6.75 turaya::user::UserManagerSrv Class Reference

Server side UserManager representation.

`#include <UserManagerSrv.hxx>`

Public Types

- typedef std::map< UserID, UserSrv::Pointer > UserSrvPtrs

Public Member Functions

- UserManagerSrv ()
- virtual ~UserManagerSrv () throw ()
- UserSrvPtrs getAllUsers ()

  Returns all installed Users.
- bool hasUser (std::string username)

  checks if a user is already installed.
- void installUser (UserData userData)

  Install a new User.

Public Attributes

- sirrix::utils::Signal1 < UserSrv::Pointer > signalUserInstalled
- sirrix::utils::Signal1 < turaya::UserID > signalUserRemoved

  Emitted on removement of a user.

Friends

- class UserHypervisor

6.75.1 Detailed Description

Server side UserManager representation.

6.75.2 Member Typedef Documentation

6.75.2.1 typedef std::map<UserID, UserSrv::Pointer> turaya::user::UserManagerSrv::UserSrvPtrs

6.75.3 Constructor & Destructor Documentation

6.75.3.1 UserManagerSrv::UserManagerSrv ( )

```
                              :
                signalUserInstalled(), signalUserRemoved(), myUsers() {
        debugFFL << "enter" << endl;
        loadUsers();
        debugFFL << "leave" << endl;
}
```

6.75.3.2 UserManagerSrv::~UserManagerSrv ( ) throw () [virtual]

```
                                          {
        debugFFL << "~UserManagerSrv()" << endl;
        myUsers.erase( myUsers.begin(), myUsers.end() );
        debugFFL << "leave" << endl;
}
```

6.75.4 Member Function Documentation

6.75.4.1 UserManagerSrv::UserSrvPtrs UserManagerSrv::getAllUsers ( )

Returns all installed Users.

Returns

All Users.

```
                                                                {
        return myUsers;
}
```

6.75.4.2 bool UserManagerSrv::hasUser (std::string *username*)

checks if a user is already installed.

Returns

true if user exists else false.

```
                                                    {
        UserSrvPtrs::iterator i;

        for( i = myUsers.begin(); i != myUsers.end(); i++ ) {
                if( i->second->getUsername() == username ) {
                        return true;
                }
        }

        return false;
}
```

6.75.4.3 void UserManagerSrv::installUser (UserData *userData*)

Install a new User.

Parameters

userData	UserData describing the User to install

Exceptions

UserSrvAlready-Exists	if the User already exists
UserSrvInvalidUser-Data	if the given UserData is invalid

```
                                                        {

        debugFFL << "Installing new User with ID " << userData.getUserId() <<
      endl;
        if( myUsers.size() > 0 ) {
                debugFFL << "One User already exists." << endl;
                throw UserSrvWrongState();
        }

        stringstream sstr;
        UserSrvPtrs::iterator i = myUsers.find( userData.getUserId() );
        if( i != myUsers.end() ) {
                throw UserSrvAlreadyExists();
        }

        UserHypervisor userHypervisor;
        userHypervisor.installUser( userData );
        UserSrv::Pointer userSrv(
                        new UserSrv( userData.getUsername(), userData.
      getRealname(), userData.getUserId(), userData.getGroupId(),
                                        userData.getExpirationDate(), userData.
      getExpirationType(), userData.getAuthenticationStatus() ) );
        userSrv->signalUserRemoved.Connect( this, &
      UserManagerSrv::slotUserRemoved );
        myUsers.insert( pair<UserID, UserSrv::Pointer>( userSrv->getUserID(),
```

```
      userSrv ) );
        signalUserInstalled( userSrv );
}
```

6.75.5 Friends And Related Function Documentation

6.75.5.1 friend class UserHypervisor [friend]

6.75.6 Member Data Documentation

6.75.6.1 sirrix::utils::Signal1<UserSrv::Pointer> turaya::user::UserManagerSrv::signalUserInstalled

6.75.6.2 sirrix::utils::Signal1<turaya::UserID> turaya::user::UserManagerSrv::signalUserRemoved

Emitted on removement of a user.

6.76 turaya::user::UserSrv Class Reference

Server side user representation.

`#include <UserSrv.hxx>`

Public Types

- typedef sirrix::utils::SharedPointer < UserSrv > Pointer

Public Member Functions

- UserSrv (std::string username, std::string realname, unsigned int userid, unsigned int groupid, time_t expirationdate, bool expirationtype, UserData::AuthenticationStatus authenticationstatus)
- virtual ~UserSrv ()
- bool accountHasExpired () const
- UserID getUserID () const

  Returns the user identifier of this user.
- std::string getUsername () const

  Return the Username of this user.
- turaya::User::Status getStatus () const

  returns DBUS status of this UserSrv object
- UserData::AuthenticationStatus getAuthenticationStatus () const

  returns users authentication status. ACCESS_DENIED or ACCESS_GRANTED
- std::string getRealname () const

returns users realname

- std::string getHomedir () const

  returns users home directory path
- unsigned int getGroupID () const

  returns users groupID
- int getExpirationType () const

  returns expirationType as integer. 0 if expiration affects local cache only, 1 if the account globally expires
- time_t getExpirationDate () const

  returns expiration date in number of days since 1.1.1970
- UserData getUserData () const

  returns all user related data
- void update (const UserData &userData)
- void remove ()

  removes this User. Keeps home directory and compartments

Public Attributes

- sirrix::utils::Signal1 < turaya::User::Status > onStatusChanged

  Emitted when status of this User has changed.
- sirrix::utils::Signal1 < turaya::UserID > signalUserRemoved

  Emitted on removement of this User.

6.76.1 Detailed Description

Server side user representation.

6.76.2 Member Typedef Documentation

6.76.2.1 typedef sirrix::utils::SharedPointer<UserSrv> turaya::user::UserSrv::Pointer

6.76.3 Constructor & Destructor Documentation

6.76.3.1 UserSrv::UserSrv (std::string *username*, std::string *realname*, unsigned int *userid*, unsigned int *groupid*, time_t *expirationdate*, bool *expirationtype*, UserData::AuthenticationStatus *authenticationstatus*)

```
                :
                onStatusChanged(), signalUserRemoved(), myStatus(
      turaya::User::undefined ), myUserID( userid ), myUsername( username ), myAuthenticationStatus
      (
                                authenticationstatus ), myExpirationIsGlobal(
      expirationtype ), myRealname( realname ), myHomedir( HOME + username ), myGroupId
      (
                                groupid ), myExpirationDate( expirationdate ) {
        debugFFL << "myExpirationDate: " << expirationdate << endl;
}
```

6.76.3.2 UserSrv::~UserSrv ( ) [virtual]

```
                {
        debugFFL << "~UserSrv()" << endl;
}
```

6.76.4 Member Function Documentation

6.76.4.1 bool UserSrv::accountHasExpired ( ) const

```
                                        {
        time_t now = time( NULL ) / 86400;
        debugFFL << "Time: " << now << ", ExpireDate: " << myExpirationDate <<
      endl;

        if( now > myExpirationDate ) {
                return true;
        }else {
                return false;
        }
}
```

6.76.4.2 UserData::AuthenticationStatus UserSrv::getAuthenticationStatus ( ) const

returns users authentication status. ACCESS_DENIED or ACCESS_GRANTED

```
                                                                        {
        return myAuthenticationStatus;
}
```

6.76.4.3 time_t UserSrv::getExpirationDate ( ) const

returns expiration date in number of days since 1.1.1970

```
                                        {
        return myExpirationDate;
}
```

6.76.4.4 int UserSrv::getExpirationType ( ) const

returns expirationType as integer. 0 if expiration affects local cache only, 1 if the account globally expires

```
                                        {
        return myExpirationIsGlobal;
}
```

6.76.4.5 unsigned int UserSrv::getGroupID ( ) const

returns users groupID

```
                                        {
        return myGroupId;
}
```

6.76.4.6 string UserSrv::getHomedir ( ) const

returns users home directory path

```
                                    {
        return myHomedir;
}
```

6.76.4.7 string UserSrv::getRealname ( ) const

returns users realname

```
                                     {
        return myRealname;
}
```

6.76.4.8 turaya::User::Status UserSrv::getStatus ( ) const

returns DBUS status of this UserSrv object

```
                                          {
        return myStatus;
}
```

6.76.4.9 UserData UserSrv::getUserData ( ) const

returns all user related data

```
                                    {
        UserData userData( myUsername, "placeholder", myRealname, myUserID,
      myGroupId, myExpirationDate, myExpirationIsGlobal,
                        myAuthenticationStatus );

        return userData;
}
```

6.76.4.10 UserID UserSrv::getUserID ( ) const

Returns the user identifier of this user.

```
                                {
        return myUserID;
}
```

6.76.4.11 string UserSrv::getUsername ( ) const

Return the Username of this user.

```
                                  {
        return myUsername;
}
```

6.76.4.12 void UserSrv::remove ( )

removes this User. Keeps home directory and compartments

Exceptions

UserSrvWrongState	if this user in not currently stopped

```
                {
        debugFFL << "enter" << endl;
        myStatus = User::removed;
        onStatusChanged( myStatus );

        UserHypervisor userHypervisor;
        userHypervisor.removeUser( myUsername );

        signalUserRemoved( myUserID );
        debugFFL << "leave" << endl;
}
```

6.76.4.13 void UserSrv::update (const UserData & *userData*)

```
                                           {
        debugFFL << "enter" << endl;

        if( myUserID != userData.getUserId() ) {
                debugFFL << "UserID mismatch. User with ID " << myUserID << "
       can not be updated through UserData with ID "
                                << userData.getUserId() << endl;
                throw UserSrvInvalidUserData( "userID mismatch" );
        }

        // really update
        // in a later implementation (that uses the >user ID< as the unique
```

```
        user identifier), the username should be updateable
         // myUsername = userData.getUsername(); // ToDo: what about the home
        directory?
         if( myRealname != userData.getRealname() )
                 myRealname = userData.getRealname();

         if( myExpirationIsGlobal != userData.getExpirationType() )
                 myExpirationIsGlobal = userData.getExpirationType();

         if( myExpirationDate != userData.getExpirationDate() )
                 myExpirationDate = userData.getExpirationDate();

         // update password if changed
         UserHypervisor userHypervisor;
         userHypervisor.updateUser( userData );

         onStatusChanged( myStatus );
         debugFFL << "leave" << endl;
}
```

6.76.5 Member Data Documentation

6.76.5.1 sirrix::utils::Signal1<turaya::User::Status> turaya::user::UserSrv::onStatusChanged

Emitted when status of this User has changed.

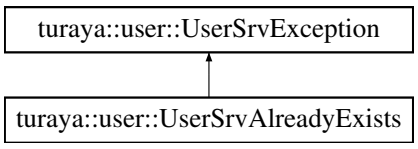
6.76.5.2 sirrix::utils::Signal1<turaya::UserID> turaya::user::UserSrv::signalUserRemoved

Emitted on removement of this User.

6.77 turaya::user::UserSrvAlreadyExists Class Reference

`#include <UserExceptionsSrv.hxx>`

Inheritance diagram for turaya::user::UserSrvAlreadyExists:



Public Member Functions

- UserSrvAlreadyExists (const string &context="UserSrvAlreadyExists")
- virtual ~UserSrvAlreadyExists () throw ()

6.77.1 Constructor & Destructor Documentation

6.77.1.1 turaya::user::UserSrvAlreadyExists::UserSrvAlreadyExists (const string & *context* = "UserSrvAlreadyExists") [inline]

```
:
                                UserSrvException( context ) {
  }
```

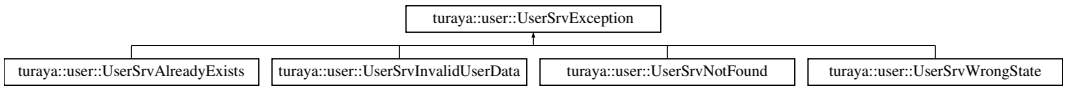
6.77.1.2 virtual turaya::user::UserSrvAlreadyExists::~UserSrvAlreadyExists ( ) throw () [inline, virtual]

```
                                                                {
  }
```

6.78 turaya::user::UserSrvException Class Reference

`#include <UserExceptionsSrv.hxx>`

Inheritance diagram for turaya::user::UserSrvException:



Public Member Functions

- UserSrvException (const string &context)
- virtual ~UserSrvException () throw ()

6.78.1 Constructor & Destructor Documentation

6.78.1.1 turaya::user::UserSrvException::UserSrvException (const string & *context*) [inline]

```
                                                                :
                                runtime_error( context ) {
  }
```

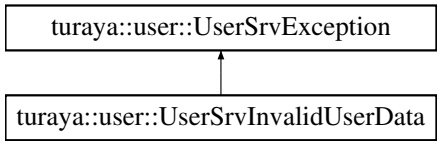
6.78.1.2 virtual turaya::user::UserSrvException::~UserSrvException ( ) throw () [inline, virtual]

```
                                                                {
  }
```

6.79 turaya::user::UserSrvInvalidUserData Class Reference

`#include <UserExceptionsSrv.hxx>`

Inheritance diagram for turaya::user::UserSrvInvalidUserData:



Public Member Functions

- UserSrvInvalidUserData (const string &context="UserSrvInvalidUserData")
- virtual ~UserSrvInvalidUserData () throw ()

6.79.1 Constructor & Destructor Documentation

6.79.1.1 turaya::user::UserSrvInvalidUserData::UserSrvInvalidUserData (const string & *context* = "UserSrvInvalidUserData") [inline]

```
:
                                UserSrvException( context ) {
}
```

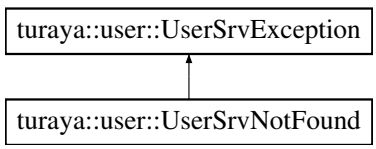
6.79.1.2 virtual turaya::user::UserSrvInvalidUserData::~UserSrvInvalidUserData ( ) throw () [inline, virtual]

```
                                                                {
}
```

6.80 turaya::user::UserSrvNotFound Class Reference

`#include <UserExceptionsSrv.hxx>`

Inheritance diagram for turaya::user::UserSrvNotFound:



Public Member Functions

- UserSrvNotFound (const string &context="UserSrvNotFound")
- virtual ~UserSrvNotFound () throw ()

6.80.1 Constructor & Destructor Documentation

6.80.1.1 turaya::user::UserSrvNotFound::UserSrvNotFound (const string & *context* = "UserSrvNotFound") [inline]

```
                :
                                                UserSrvException( context ) {
                                }
```

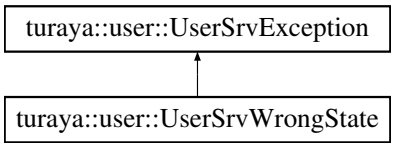
6.80.1.2 virtual turaya::user::UserSrvNotFound::~UserSrvNotFound ( ) throw () [inline, virtual]

```
                                                                        {
                                }
```

6.81 turaya::user::UserSrvWrongState Class Reference

```
#include <UserExceptionsSrv.hxx>
```

Inheritance diagram for turaya::user::UserSrvWrongState:



Public Member Functions

- UserSrvWrongState (const string &context="UserSrvWrongState")
- virtual ~UserSrvWrongState () throw ()

6.81.1 Constructor & Destructor Documentation

6.81.1.1 turaya::user::UserSrvWrongState::UserSrvWrongState (const string & *context* = "UserSrvWrongState") [inline]

```
                :
                                                UserSrvException( context ) {
                                }
```

6.81.1.2 virtual turaya::user::UserSrvWrongState::~UserSrvWrongState ( ) throw () [inline, virtual]

```
                                                                    {
                            }
```

6.82 turaya::organization::VDIDownloader Class Reference

```
#include <CompartmentManagerObserver.hxx>
```

Public Member Functions

- VDIDownloader (OrganizationSrv &org, CompartmentID id)
- void * run ()

6.82.1 Constructor & Destructor Documentation

6.82.1.1 turaya::organization::VDIDownloader::VDIDownloader (OrganizationSrv & *org,* CompartmentID *id*) [inline]

```
        :
                                    myOrganization(org),
                                    myCompartmentId(id){};
```

6.82.2 Member Function Documentation

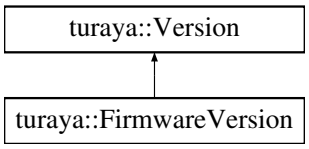
6.82.2.1 void * VDIDownloader::run ( )

```
                         {
        while (true) {
                try {
                        myOrganization.downloadAndInstallVDIFile(
    myCompartmentId);
                        return 0;
                } catch(OrganizationSrvNoTOMConnection &e) {
                        debugFFL << "No connection to TOM, trying again in 30
     seconds..." << endl;
                        sleep(30);
                        continue;
                } catch (OrganizationSrvException &e) {
                        debugFFL << "Exception during
     downloadAndInstallVDIFile: " << e.what() << endl;
                        debugFFL << "Trying again in 30 seconds..." << endl;
                        sleep(30);
                        continue;
                }
        }
}
```

6.83 turaya::Version Class Reference

`#include <TrustedDesktop.hxx>`

Inheritance diagram for turaya::Version:



Public Member Functions

- Version (UInt32 major, UInt32 minor, UInt32 patchlevel)
- virtual ~Version ()

  Default destructor.
- UInt32 getMajor () const

  Get major number.
- UInt32 getMinor () const

  Get minor number.
- UInt32 getPatchlevel () const

  Get patch level.
- std::string getString () const

  Get version as string: "major.minor.patchlevel".
- bool operator== (const Version &other)

  compare equal operator
- bool operator!= (const Version &other)

  compare not equal operator
- bool operator>= (const Version &other)

  compare greater or equal operator
- bool operator> (const Version &other)

  compare greater operator
- bool operator<= (const Version &other)

  compare smaller or equal operator
- bool operator< (const Version &other)

  compare smaller operator

Protected Member Functions

- Version ()

Protected Attributes

- UInt32 myMajor
- UInt32 myMinor
- UInt32 myPatchlevel

6.83.1 Detailed Description

An abstraction for a software version number.

6.83.2 Constructor & Destructor Documentation

6.83.2.1 turaya::Version::Version (UInt32 *major*, UInt32 *minor*, UInt32 *patchlevel*) [inline]

Constructor

Parameters

major	Major version number
minor	Minor version number
patchlevel	Patch level

```
:
                                        myMajor( major ), myMinor( minor ),
myPatchlevel( patchlevel ) {
                    }
```

6.83.2.2 virtual turaya::Version::~Version ( ) [inline, virtual]

Default destructor.

```
                                        {
                    }
```

6.83.2.3 turaya::Version::Version ( ) [protected]

6.83.3 Member Function Documentation

6.83.3.1 UInt32 turaya::Version::getMajor ( ) const [inline]

Get major number.

```
                                    {
                return myMajor;
                    }
```

6.83.3.2 UInt32 turaya::Version::getMinor ( ) const [inline]

Get minor number.

```
                                    {
                        return myMinor;
                    }
```

6.83.3.3 UInt32 turaya::Version::getPatchlevel ( ) const [inline]

Get patch level.

```
                                        {
                        return myPatchlevel;
                    }
```

6.83.3.4 std::string turaya::Version::getString ( ) const [inline]

Get version as string: "major.minor.patchlevel".

Reimplemented in turaya::FirmwareVersion.

```
                                        {
                        std::stringstream ss;
                        ss << myMajor << "." << myMinor << "." <<
    myPatchlevel;
                        return ss.str();
                    }
```

6.83.3.5 bool turaya::Version::operator!= (const Version & *other*) [inline]

compare not equal operator

```
                                                {
                        return !( *this == other );
                    }
```

6.83.3.6 bool turaya::Version::operator< (const Version & *other*) [inline]

compare smaller operator

```
                                                    {
                        return ( ( myPatchlevel < other.myPatchlevel &&
     myMinor == other.myMinor && myMajor == other.myMajor ) /* 1.0.1 > 1.0.2 */
                        || ( myMinor < other.myMinor && myMajor ==
    other.myMajor ) /* 1.0.2 > 1.1.0 */
                        || ( myMajor < other.myMajor ) ); /* 1.0.2 <
     2.0.0 */
                    }
```

6.83.3.7 bool turaya::Version::operator<= (const Version & *other*) [inline]

compare smaller or equal operator

```
                                                                {
                        return ( myPatchlevel <= other.myPatchlevel &&
myMinor <= other.myMinor && myMajor <= other.myMajor );
                }
```

6.83.3.8 bool turaya::Version::operator== (const Version & *other*) [inline]

compare equal operator

```
                                                                {
                        return ( myMajor == other.myMajor && myMinor ==
 other.myMinor && myPatchlevel == other.myPatchlevel );
                }
```

6.83.3.9 bool turaya::Version::operator> (const Version & *other*) [inline]

compare greater operator

```
                                                               {
                        return ( ( myPatchlevel > other.myPatchlevel &&
 myMinor == other.myMinor && myMajor == other.myMajor ) /* 1.0.2 > 1.0.1 */
                        || ( myMinor > other.myMinor && myMajor ==
other.myMajor ) /* 1.1.0 > 1.0.2 */
                        || ( myMajor > other.myMajor ) ); /* 2.0.0 >
 1.0.2 */
                }
```

6.83.3.10 bool turaya::Version::operator>= (const Version & *other*) [inline]

compare greater or equal operator

```
                                                                {
                        return ( myPatchlevel >= other.myPatchlevel &&
myMinor >= other.myMinor && myMajor >= other.myMajor );
                }
```

6.83.4 Member Data Documentation

6.83.4.1 UInt32 turaya::Version::myMajor [protected]

6.83.4.2 UInt32 turaya::Version::myMinor [protected]

6.83.4.3 UInt32 turaya::Version::myPatchlevel [protected]

6.84 turaya::compartment::VMOptionParser Class Reference

`#include <VMOptionParser.hxx>`

Public Member Functions

- VMOptionParser (const std::string &options)
- virtual ~VMOptionParser ()
- void parseVMOptions (UInt32 &mem, UInt32 &vmem, std::string &audioIF, std::string &audioHW, std::string &nicHW, bool &pae, bool &smartCard, bool &usbPlainAccess, bool &cdPassthru, std::string &smbShare, bool &amd64)

6.84.1 Constructor & Destructor Documentation

6.84.1.1 VMOptionParser::VMOptionParser (const std::string & *options*)

```
                                                    :
        myOptions(options){
}
```

6.84.1.2 VMOptionParser::~VMOptionParser ( ) [virtual]

```
{}
```

6.84.2 Member Function Documentation

6.84.2.1 void VMOptionParser::parseVMOptions (UInt32 & *mem,* UInt32 & *vmem,* std::string & *audioIF,* std::string & *audioHW,* std::string & *nicHW,* bool & *pae,* bool & *smartCard,* bool & *usbPlainAccess,* bool & *cdPassthru,* std::string & *smbShare,* bool & *amd64*)

```
                                                                    {

        //use default values if something goes wrong
        mem = myDefMem;
        vmem = myDefVmem;
        audioIF = myDefAudioIF;
        audioHW = myDefAudioHW;
        nicHW = myDefNicHW;
        pae = myDefPae;
        smartCard = myDefSmartCard;
        usbPlainAccess = myDefUsbPlainAccess;
        cdPassthru = myDefCdPassthru;
        smbShare = myDefSmbShare;
        amd64 = myDefAmd64;

        string pae_string = myDefPae?"on":"off";
        string smartCard_string = myDefSmartCard?"on":"off";
```

```
  string usbPlainAccess_string = myDefUsbPlainAccess?"on":"off";
  string cdPassthru_string = myDefCdPassthru?"on":"off";
  string amd64_string = myDefAmd64?"on":"off";

  OptionList optionList;
  Option<UInt32> memOption(optionList, "ram", "r", myDefMem,
BaseOption::optional);
  Option<UInt32> vmemOption(optionList, "vram", "v", myDefVmem,
BaseOption::optional);
  Option<string> audioIFOption(optionList, "audioIF", "a", myDefAudioIF,
BaseOption::optional);
  Option<string> audioHWOption(optionList, "audioHW", "u", myDefAudioHW,
BaseOption::optional);
  Option<string> nicHWOption(optionList, "nicHW", "n", myDefNicHW,
BaseOption::optional);
  Option<string> paeOption(optionList, "pae", "p", pae_string,
BaseOption::optional);
  Option<string> smartCardOption(optionList, "smartCard", "s",
smartCard_string, BaseOption::optional);
  Option<string> usbPlainAccessOption(optionList, "usbPlainAccess", "b",
usbPlainAccess_string, BaseOption::optional);
  Option<string> cdPassthruOption(optionList, "cdPassthru", "c",
cdPassthru_string, BaseOption::optional);
  Option<string> smbShareOption(optionList, "smbBackup", "m",
myDefSmbShare, BaseOption::optional);
  Option<string> amd64Option(optionList, "amd64", "d", amd64_string,
BaseOption::optional);
  CmdLineParser parser(optionList);

  string tmps;
  istringstream is( myOptions );
  vector<string> params;

  while( is.good() ) {
          is >> tmps;
          params.push_back( tmps );
  }

  unsigned int argc = params.size()+1;
  const char** argv = new const char*[argc];
  argv[0] = 0;

  for(unsigned int i = 1; i < argc; i++){
          argv[i] = params[i-1].c_str();
  }

  try {
          parser.parseCmdLine(argc, argv);
          delete[] argv;
  }catch (ParseError& e){
          debugFFL << "ParseError: " << e.what() << endl;
          delete[] argv;
          return;
  } catch (OptionError& e){
          debugFFL << "OptionError: " << e.what() << endl;
          delete[] argv;
          return;
  }

  if (myMemRange.find( memOption.getValue() ) != myMemRange.end()){
          mem =  memOption.getValue();
  }
```

```
        if (myVmemRange.find (vmemOption.getValue()) != myVmemRange.end()){
                vmem = vmemOption.getValue();
        }
        if (myAudioIFRange.find(audioIFOption.getValue()) != myAudioIFRange.end
      ()){
                audioIF = audioIFOption.getValue();
        }
        if (myAudioHWRange.find(audioHWOption.getValue()) != myAudioHWRange.end
      ()){
                audioHW = audioHWOption.getValue();
        }
        if (myNicHWRange.find(nicHWOption.getValue()) != myNicHWRange.end()){
                nicHW = nicHWOption.getValue();
        }
        pae = paeOption.getValue()=="off"?false:true;
        smartCard = smartCardOption.getValue()=="on"?true:false;
        usbPlainAccess = usbPlainAccessOption.getValue()=="on"?true:false;
        cdPassthru = cdPassthruOption.getValue()=="on"?true:false;
        amd64 = amd64Option.getValue()=="on"?true:false;
        smbShare = smbShareOption.getValue();

}
```

Chapter 7

File Documentation

7.1 CompartmentAdaptor.cxx File Reference

```
#include <stdio.h>  #include <string>  #include <vector>
#include <sstream> #include <iostream> #include <Turaya/-
CompartmentManagementService1/CompartmentAdaptor.hxx>
#include <Turaya/CompartmentProxy1/CompartmentExceptions.-
hxx>  #include <Sirrix/Utils2/Debugging.hxx>  #include <-
Sirrix/Utils2/ByteVector.hxx>
```

Functions

- const std::string COMPARTMENT_INTERFACE ("turaya.compartment.-compartment")

7.1.1 Function Documentation

7.1.1.1 const std::string COMPARTMENT_INTERFACE ("turaya.compartment.compartment")

7.2 CompartmentAdaptor.hxx File Reference

```
#include <string> #include <vector> #include <dbus++/object.-
hpp> #include <dbus++/variant.hpp> #include <dbus++/fixed-
_array.hpp>  #include <dbus++/dbus_type_cast_traits.hpp>
#include <Sirrix/Utils2/Signal.hxx>  #include <Turaya/-
CompartmentManagementService1/CompartmentSrv.hxx> #include
<Turaya/CompartmentManagementService1/CompartmentExceptions-
Srv.hxx>
```

Classes

- class turaya::compartment::CompartmentAdaptor

  Adaptor class to make the CompartmentSrv class accessible over DBus.

Namespaces

- namespace turaya
- namespace turaya::compartment

7.3 CompartmentExceptionsSrv.hxx File Reference

```
#include <stdexcept>
```

Classes

- class turaya::compartment::CompartmentSrvException
- class turaya::compartment::CompartmentSrvAlreadyExists
- class turaya::compartment::CompartmentSrvNotFound
- class turaya::compartment::CompartmentSrvHasNoDomamin
- class turaya::compartment::CompartmentSrvInvalidCompartmentData
- class turaya::compartment::CompartmentSrvWrongState
- class turaya::compartment::CompartmentSrvMissingDomain
- class turaya::compartment::CompartmentSrvVDIException
- class turaya::compartment::CompartmentSrvVDIHashMismatch

Namespaces

- namespace turaya
- namespace turaya::compartment

7.4 CompartmentManagement_Tests.cxx File Reference

```
#include <Sirrix/TestFramework1/Framework.hxx>  #include
<Sirrix/TestFramework1/HelperMacros.hxx>  #include <-
Sirrix/Utils2/Debugging.hxx> #include <Turaya/Compartment-
ManagementService1/CompartmentManagerAdaptor.hxx> #include
<Turaya/CompartmentProxy1/CompartmentExceptions.hxx>
```

Classes

- class unittests::CompartmentManagement_Test::CompartmentManager-Observer
- class unittests::CompartmentManagement_Test::CompartmentObserver

Namespaces

- namespace unittests
- namespace unittests::CompartmentManagement_Test

Functions

- unittests::CompartmentManagement_Test::TEST_CASE (Compartment-Management_Tests)
- unittests::CompartmentManagement_Test::TEST_CASE (Compartment_Tests)

7.5 CompartmentManagerAdaptor.cxx File Reference

```
#include <Turaya/CompartmentManagementService1/Compartment-
ManagerAdaptor.hxx> #include <Turaya/CompartmentManagement-
Service1/CompartmentExceptionsSrv.hxx> #include <Turaya/-
CompartmentProxy1/CompartmentExceptions.hxx>  #include <-
Sirrix/Utils2/ByteVector.hxx>   #include <Sirrix/Utils2/-
Debugging.hxx>
```

Functions

- const std::string COMPARTMENT_MANAGER_INTERFACE ("turaya.compartment.-manager")
- const std::string COMPARTMENTS_PATH ("/compartments/Compartment_")

7.5.1 Function Documentation

7.5.1.1 const std::string COMPARTMENT_MANAGER_INTERFACE ("turaya.compartment.manager")

7.5.1.2 const std::string COMPARTMENTS_PATH ("/compartments/Compartment_")

7.6 CompartmentManagerAdaptor.hxx File Reference

```
#include <dbus++/object.hpp>   #include <dbus++/variant.-
hpp> #include <dbus++/fixed_array.hpp> #include <dbus++/dbus-
```

```
_type_cast_traits.hpp>  #include <dbus++/exception.hpp> ×
#include <Sirrix/Utils2/SharedPointer.hxx>    #include <-
Sirrix/Utils2/Types.hxx>    #include <Turaya/Compartment-
ManagementService1/CompartmentAdaptor.hxx>    #include <-
Turaya/CompartmentManagementService1/CompartmentManager-
Srv.hxx>
```

Classes

- class __installedCompartmentInfo
- class turaya::compartment::CompartmentManagerAdaptor

  Adaptor class to make the CompartmentManagerSrv class accessible over DBus.

Namespaces

- namespace turaya
- namespace turaya::compartment

Typedefs

- typedef ::__installedCompartmentInfo turaya::compartment::InstalledCompartment-Info

7.7 CompartmentManagerObserver.cxx File Reference

```
#include <Turaya/OrganizationManagementService1/Organization-
Srv.hxx> #include <Turaya/OrganizationManagementService1/-
CompartmentManagerObserver.hxx> #include <Turaya/Compartment-
Proxy1/CompartmentExceptions.hxx> #include <Turaya/Organization-
ManagementService1/OrganizationExceptionsSrv.hxx> #include
<Turaya/PlatformManagement1/TrustedDesktop.hxx>  #include
<Sirrix/Utils2/Debugging.hxx> #include <Sirrix/Hypervisor2/-
ScopedMutex.hxx>
```

7.8 CompartmentManagerObserver.hxx File Reference

```
#include <set> #include <Sirrix/Utils2/Types.hxx> #include
<Sirrix/Utils2/Debugging.hxx> #include <Turaya/Compartment-
Proxy1/CompartmentManager.hxx> #include <Sirrix/Hypervisor2/-
Thread.hxx> #include <Sirrix/Hypervisor2/Semaphore.hxx>
```

Classes

- class turaya::organization::VDIDownloader
- class turaya::organization::CompartmentManagerObserver

Namespaces

- namespace turaya
- namespace turaya::organization

7.9 CompartmentManagerSrv.cxx File Reference

```
#include <sqlite3.h>     #include <Turaya/Database1/Data-
BaseUtils.hxx> #include <Turaya/Database1/DataBaseError.-
hxx>    #include <Turaya/DomainProxy1/DomainManager.hxx> ×
#include <Turaya/DomainProxy1/DomainExceptions.hxx> ×
#include <Turaya/CompartmentManagementService1/Compartment-
ManagerSrv.hxx>    #include <Turaya/CompartmentManagement-
Service1/VMOptionParser.hxx>     #include <Sirrix/Utils2/-
Debugging.hxx>
```

7.10 CompartmentManagerSrv.hxx File Reference

```
#include <stdio.h>  #include <string>  #include <Sirrix/-
Utils2/SharedPointer.hxx> #include <Sirrix/Utils2/Signal.-
hxx> #include <Turaya/CompartmentProxy1/CompartmentManager.-
hxx>     #include <Turaya/CompartmentManagementService1/-
CompartmentSrv.hxx> #include <Turaya/CompartmentManagement-
Service1/CompartmentExceptionsSrv.hxx>
```

Classes

- class turaya::compartment::CompartmentManagerSrv

  Server side CompartmentManager representation.

Namespaces

- namespace turaya
- namespace turaya::compartment

7.11 CompartmentSrv.cxx File Reference

```
#include <sqlite3.h> #include <stdio.h> #include <iostream> ×
#include <sys/stat.h>  #include <sys/types.h>  #include
<pwd.h>  #include <Turaya/CompartmentManagementService1/-
CompartmentSrv.hxx> #include <Turaya/CompartmentManagement-
Service1/CompartmentExceptionsSrv.hxx> #include <Sirrix/-
Utils2/Debugging.hxx> #include <Sirrix/Utils2/FileBinary-
Stream.hxx>  #include <Turaya/Database1/DataBaseUtils.-
hxx> #include <Turaya/Database1/DataBaseError.hxx> #include
<Sirrix/CryptoManager1/SystemFactory.hxx>  #include <-
Sirrix/CryptoManager1/CryptoManager.hxx> #include <Sirrix/-
Hypervisor2/System.hxx>  #include <Sirrix/Hypervisor2/-
FileManager.hxx> #include <Turaya/UserProxy1/UserManager.-
hxx>  #include <Turaya/UserProxy1/UserExceptions.hxx> ×
#include <Turaya/DomainProxy1/DomainManager.hxx> #include
<Turaya/DomainProxy1/DomainExceptions.hxx>  #include <-
Turaya/CompartmentManagementService1/VMOptionParser.-
hxx> #include <Turaya/PlatformManagement1/TrustedDesktop.-
hxx>  #include <Turaya/CompartmentManagementService1/FTP-
Access.hxx> #include <Sirrix/Utils2/string_cast.hxx>
```

Functions

- const Path SHARE_MOUNT_PATH_ROOT ("/vmdata/shared/guestfs")

7.11.1 Function Documentation

7.11.1.1 const Path SHARE_MOUNT_PATH_ROOT ("/vmdata/shared/guestfs")

7.12 CompartmentSrv.hxx File Reference

```
#include <string>  #include <vector>  #include <Sirrix/-
Utils2/Types.hxx>  #include <Sirrix/Utils2/Signal.hxx> ×
#include <Sirrix/Utils2/Utils.hxx>  #include <Turaya/-
CompartmentProxy1/Compartment.hxx>  #include <Turaya/-
CompartmentProxy1/CompartmentData.hxx> #include <Sirrix/-
Hypervisor2/Path.hxx> #include <Sirrix/Hypervisor2/Thread.-
hxx> #include <Sirrix/Hypervisor2/Semaphore.hxx>
```

Classes

- class turaya::compartment::CompartmentSrv

  Server side compartment representation.
- class **turaya::compartment::CompartmentSrv::StopThread**

- class **turaya::compartment::CompartmentSrv::StartThread**
- class **turaya::compartment::CompartmentSrv::ProgressThread**
- class **turaya::compartment::CompartmentSrv::ExportThread**
- class **turaya::compartment::CompartmentSrv::SMBAccessThread**
- class **turaya::compartment::CompartmentSrv::RunningCompartmentWatchDog**

Namespaces

- namespace turaya
- namespace turaya::compartment

Functions

- const std::string COMPARTMENTTABLE ("Compartments")
- const std::string COMPARTMENTTABLEPID ("CompartmentPids")

7.12.1 Function Documentation

7.12.1.1 const std::string COMPARTMENTTABLE ("Compartments")

7.12.1.2 const std::string COMPARTMENTTABLEPID ("CompartmentPids")

7.13 DomainAdaptor.cxx File Reference

```
#include <Turaya/DomainManagementService1/DomainAdaptor.-
hxx>     #include <Turaya/DomainProxy1/DomainExceptions.-
hxx>  #include <Sirrix/Utils2/Debugging.hxx>  #include <-
Sirrix/Utils2/ByteVector.hxx> #include <stdio.h> #include
<string>  #include <vector>  #include <sstream>  #include
<iostream>
```

Functions

- const std::string DOMAIN_INTERFACE ("turaya.domain.domain")

7.13.1 Function Documentation

7.13.1.1 const std::string DOMAIN_INTERFACE ("turaya.domain.domain")

7.14 DomainAdaptor.hxx File Reference

```
#include <string> #include <vector> #include <dbus++/object.-
hpp> #include <dbus++/variant.hpp> #include <dbus++/fixed-
_array.hpp>  #include <dbus++/dbus_type_cast_traits.hpp>
#include <Sirrix/Utils2/Signal.hxx>  #include <Sirrix/-
Utils2/Types.hxx> #include <Turaya/DomainManagementService1/-
DomainSrv.hxx> #include <Turaya/DomainManagementService1/-
DomainExceptionsSrv.hxx>
```

Classes

- class turaya::domain::DomainAdaptor

  Adaptor class to make the DomainSrv class accessible over DBus.

Namespaces

- namespace turaya
- namespace turaya::domain

7.15 DomainExceptionsSrv.hxx File Reference

```
#include <stdexcept>
```

Classes

- class turaya::domain::DomainSrvException
- class turaya::domain::DomainSrvAlreadyExists
- class turaya::domain::DomainSrvNotFound
- class turaya::domain::DomainSrvHasNoDomamin
- class turaya::domain::DomainSrvInvalidDomainData
- class turaya::domain::DomainSrvWrongState
- class turaya::domain::DomainSrvCouldNotCreateDomainEncryptionOverlay
- class turaya::domain::DomainSrvCouldNotRemoveDomainEncryptionOverlay

Namespaces

- namespace turaya
- namespace turaya::domain

7.16 DomainManagerAdaptor.cxx File Reference

```
#include <Turaya/DomainManagementService1/DomainManager-
Adaptor.hxx>  #include <Turaya/DomainManagementService1/-
DomainExceptionsSrv.hxx>  #include <Turaya/DomainProxy1/-
DomainExceptions.hxx> #include <Sirrix/Utils2/Debugging.-
hxx>   #include <Sirrix/Utils2/ByteVector.hxx>   #include
<iostream>
```

Functions

- const std::string DOMAIN_MANAGER_INTERFACE ("turaya.domain.manager")
- const std::string DOMAINS_PATH ("/domains/Domain_")

7.16.1 Function Documentation

7.16.1.1 const std::string DOMAIN_MANAGER_INTERFACE ("turaya.domain.manager")

7.16.1.2 const std::string DOMAINS_PATH ("/domains/Domain_")

7.17 DomainManagerAdaptor.hxx File Reference

```
#include <dbus++/object.hpp>   #include <dbus++/variant.-
hpp> #include <dbus++/fixed_array.hpp> #include <dbus++/dbus-
_type_cast_traits.hpp>  #include <dbus++/exception.hpp> ×
#include <Sirrix/Utils2/SharedPointer.hxx>    #include <-
Sirrix/Utils2/Types.hxx> #include <Turaya/DomainManagement-
Service1/DomainAdaptor.hxx> #include <Turaya/DomainManagement-
Service1/DomainManagerSrv.hxx>
```

Classes

- class __installedDomainInfo
- class turaya::domain::DomainManagerAdaptor

  Adaptor class to make the DomainManagerSrv class accessible over DBus.

Namespaces

- namespace turaya
- namespace turaya::domain

Typedefs

- typedef ::__installedDomainInfo turaya::domain::InstalledDomainInfo

7.18 DomainManagerSrv.cxx File Reference

```
#include <sqlite3.h>    #include <Turaya/Database1/Data-
BaseUtils.hxx> #include <Turaya/Database1/DataBaseError.-
hxx> #include <Sirrix/Hypervisor2/FileManager.hxx> #include
<Turaya/DomainManagementService1/DomainManagerSrv.hxx> ×
#include <Sirrix/Utils2/Debugging.hxx>
```

7.19 DomainManagerSrv.hxx File Reference

```
#include <stdio.h> #include <string> #include <Sirrix/-
Utils2/SharedPointer.hxx> #include <Turaya/DomainProxy1/-
DomainManager.hxx> #include <Turaya/DomainProxy1/Domain-
Data.hxx>    #include <Turaya/DomainManagementService1/-
DomainSrv.hxx> #include <Turaya/DomainManagementService1/-
DomainExceptionsSrv.hxx>
```

Classes

- class turaya::domain::DomainManagerSrv

  Server side DomainManager representation.

Namespaces

- namespace turaya
- namespace turaya::domain

7.20 DomainSrv.cxx File Reference

```
#include <sstream>    #include <Sirrix/CryptoManager1/-
Encrypter.hxx> #include <Sirrix/CryptoManager1/Decrypter.-
hxx> #include <Sirrix/Hypervisor2/rndstream.hxx> #include
<Sirrix/Hypervisor2/MountUtils.hxx>   #include <Turaya/-
DomainManagementService1/DomainSrv.hxx> #include <Turaya/-
DomainManagementService1/DomainExceptionsSrv.hxx> #include
<Sirrix/Utils2/Debugging.hxx>  #include <Sirrix/Utils2/-
ByteVector.hxx>  #include <Sirrix/Utils2/ScopedPointer.-
hxx> #include <Turaya/Database1/DataBaseUtils.hxx>
```

7.21 DomainSrv.hxx File Reference

```
#include <string>  #include <vector>  #include <Sirrix/-
Utils2/Types.hxx>    #include <Sirrix/Utils2/Signal.hxx>
#include <Sirrix/Utils2/ByteVector.hxx> #include <Sirrix/-
CryptoManager1/SystemFactory.hxx> #include <Turaya/Domain-
Proxy1/Domain.hxx>  #include <Turaya/DomainProxy1/Color.-
hxx> #include <Turaya/DomainProxy1/DomainData.hxx>
```

Classes

- class turaya::domain::DomainSrv

  Server side domain representation.

Namespaces

- namespace turaya
- namespace turaya::domain

Functions

- const std::string DOMAINTABLE ("Domains")
- const std::string COMP_DOMAIN_TABLE ("Compartment_Domain")

7.21.1 Function Documentation

7.21.1.1 const std::string COMP_DOMAIN_TABLE ("Compartment_Domain")

7.21.1.2 const std::string DOMAINTABLE ("Domains")

7.22 FTPAccess.cxx File Reference

```
#include <Turaya/CompartmentManagementService1/FTPAccess.-
hxx> #include <Sirrix/Utils2/Signal.hxx> #include <curl/curl.-
h> #include <errno.h> #include <sstream> #include <cstring> ×
#include <Sirrix/Utils2/Debugging.hxx> #include <Sirrix/-
Hypervisor2/FileManager.hxx> #include <Sirrix/Utils2/string-
_cast.hxx>
```

7.23 FTPAccess.hxx File Reference

```
#include <Sirrix/Hypervisor2/Path.hxx> #include <Sirrix/-
Utils2/Delegate.hxx>  #include <Sirrix/Utils2/Types.hxx>
```

```
#include <Sirrix/Utils2/Signal.hxx>   #include <string> ×
#include <stdexcept>
```

Classes

- class turaya::compartment::FTPException
- class turaya::compartment::FTPAccess

Namespaces

- namespace turaya
- namespace turaya::compartment

7.24 NetworkManager.cxx File Reference

```
#include <stdio.h>  #include <string>  #include <vector>
#include <sstream> #include <iostream> #include <boost/bind.-
hpp>    #include <Turaya/OrganizationManagementService1/-
NetworkManager.hxx>   #include <Sirrix/Utils2/Debugging.-
hxx> #include <Sirrix/Utils2/ByteVector.hxx>
```

Functions

- const std::string NETWORK_MANAGER_SERVICE_NAME ("system://org.-freedesktop.NetworkManager")
- const std::string NETWORK_MANAGER_PATH ("/org/freedesktop/Network-Manager")

7.24.1 Function Documentation

7.24.1.1 const std::string NETWORK_MANAGER_PATH ("/org/freedesktop/NetworkManager")

7.24.1.2 const std::string NETWORK_MANAGER_SERVICE_NAME ("system://org.freedesktop.NetworkManager")

7.25 NetworkManager.hxx File Reference

```
#include <string> #include <vector> #include <dbus++/dbus.-
hpp> #include <Sirrix/Utils2/Types.hxx> #include <Sirrix/-
Utils2/SharedPointer.hxx> #include <Sirrix/Utils2/Signal.-
hxx> #include <stdexcept> #include <Turaya/Organization-
ManagementService1/NetworkManagerProxyWrapper.hxx> #include
<Turaya/PlatformManagement1/TrustedDesktop.hxx>
```

Classes

- class turaya::NetworkManagerInvalidInterface
- class turaya::NetworkManager

  Client side NetworkManager representation.

Namespaces

- namespace turaya

7.26 NetworkManagerProxy.hxx File Reference

```
#include <dbus++/dbus++.hpp>  #include <dbus++/interface-
_proxy.hpp>      #include <dbus++/basic_object_proxy.hpp>
#include <dbus++/basic_struct.hpp> #include <dbus++/variant.-
hpp> #include <dbus++/fixed_array.hpp> #include <dbus++/standard-
_interface_proxy.hpp>
```

7.27 NetworkManagerProxyWrapper.cxx File Reference

```
#include <Turaya/OrganizationManagementService1/Network-
ManagerProxyWrapper.hxx> #include <Sirrix/Utils2/Debugging.-
hxx> #include <boost/bind.hpp>
```

7.28 NetworkManagerProxyWrapper.hxx File Reference

```
#include <string>    #include <Sirrix/Utils2/Types.hxx> ×
#include <Sirrix/Utils2/Debugging.hxx> #include <Sirrix/-
Utils2/Signal.hxx> #include <Turaya/OrganizationManagement-
Service1/NetworkManagerProxy.hxx> #include <dbus++/dbus++.-
hpp> #include <dbus++/interface_proxy.hpp> #include <dbus++/basic-
_object_proxy.hpp>    #include <dbus++/basic_struct.hpp> ×
#include <dbus++/variant.hpp>    #include <dbus++/fixed_-
array.hpp>    #include <dbus++/standard_interface_proxy.-
hpp>
```

Classes

- class turaya::NetworkManagerProxyWrapper

Namespaces

- namespace turaya

7.29 OrganizationAdaptor.cxx File Reference

```
#include <Turaya/OrganizationManagementService1/Organization-
Adaptor.hxx> #include <Turaya/OrganizationProxy1/Organization-
Exceptions.hxx>     #include <Turaya/OrganizationProxy1/-
Organization.hxx>  #include <Turaya/UserProxy1/UserData.-
hxx>  #include <Sirrix/Utils2/Debugging.hxx>  #include <-
Sirrix/Utils2/ByteVector.hxx>     #include <Turaya/Share-
ManagementService1/Share.hxx> #include <stdio.h> #include
<string>  #include <vector>  #include <sstream>  #include
<iostream>
```

Functions

- const std::string ORGANIZATION_INTERFACE ("turaya.organization.organization")

7.29.1 Function Documentation

7.29.1.1 const std::string ORGANIZATION_INTERFACE ("turaya.organization.organization")

7.30 OrganizationAdaptor.hxx File Reference

```
#include <string> #include <vector> #include <dbus++/object.-
hpp> #include <dbus++/variant.hpp> #include <dbus++/fixed-
_array.hpp>   #include <dbus++/dbus_type_cast_traits.hpp>
#include <Sirrix/Utils2/Signal.hxx>    #include <Turaya/-
OrganizationManagementService1/OrganizationSrv.hxx> ×
#include <Turaya/OrganizationManagementService1/Organization-
ExceptionsSrv.hxx> #include <Turaya/UserProxy1/UserData.-
hxx>   #include <Turaya/OrganizationProxy1/Organization.-
hxx>
```

Classes

- class __ShareInfo
- class turaya::organization::OrganizationAdaptor

Namespaces

- namespace turaya
- namespace turaya::organization

Typedefs

- typedef ::__ShareInfo turaya::organization::ShareInfo

  Adaptor class to make the OrganizationSrv class accessible over DBus.

7.31 OrganizationExceptionsSrv.hxx File Reference

```
#include <stdexcept>
```

Classes

- class turaya::organization::OrganizationSrvException
- class turaya::organization::OrganizationSrvAlreadyExists
- class turaya::organization::OrganizationSrvNotFound
- class turaya::organization::OrganizationCompartmentDataSrvNotFound
- class turaya::organization::OrganizationSrvDomaindDataNotFound
- class turaya::organization::OrganizationSrvInvalidOrganizationData
- class turaya::organization::OrganizationSrvNoTOMConnection
- class turaya::organization::OrganizationSrvTOMError
- class turaya::organization::OrganizationSrvTOMTimeout

Namespaces

- namespace turaya
- namespace turaya::organization

7.32 OrganizationManagement_Tests.cxx File Reference

```
#include <Sirrix/TestFramework1/Framework.hxx>   #include
<Sirrix/TestFramework1/HelperMacros.hxx>     #include <-
Turaya/OrganizationManagementService1/OrganizationManager-
Adaptor.hxx> #include <Turaya/OrganizationProxy1/Organization-
Exceptions.hxx>     #include <Turaya/OrganizationProxy1/-
OrganizationManager.hxx>   #include <Turaya/Organization-
Proxy1/OrganizationData.hxx>    #include <Sirrix/Utils2/-
Debugging.hxx>
```

Classes

- class unittests::OrganizationManagement_Test::OrganizationManagerObserver
- class unittests::OrganizationManagement_Test::OrganizationObserver

Namespaces

- namespace unittests
- namespace unittests::OrganizationManagement_Test

7.33 OrganizationManagerAdaptor.cxx File Reference

```
#include <Turaya/OrganizationManagementService1/Organization-
ManagerAdaptor.hxx> #include <Turaya/OrganizationManagement-
Service1/OrganizationExceptionsSrv.hxx> #include <Turaya/-
OrganizationProxy1/OrganizationExceptions.hxx>   #include
<Sirrix/Utils2/ByteVector.hxx>  #include <Sirrix/Utils2/-
Debugging.hxx>
```

Functions

- const std::string ORGANIZATION_MANAGER_INTERFACE ("turaya.organization.-manager")
- const std::string ORGANIZATION_PATH ("/organizations/Organization_")

7.33.1 Function Documentation

7.33.1.1 const std::string ORGANIZATION_MANAGER_INTERFACE ("turaya.organization.manager")

7.33.1.2 const std::string ORGANIZATION_PATH ("/organizations/Organization_")

7.34 OrganizationManagerAdaptor.hxx File Reference

```
#include <dbus++/object.hpp>   #include <dbus++/variant.-
hpp> #include <dbus++/fixed_array.hpp> #include <dbus++/dbus-
_type_cast_traits.hpp>   #include <dbus++/exception.hpp>
#include <Turaya/OrganizationManagementService1/Organization-
Adaptor.hxx>    #include <Turaya/OrganizationManagement-
Service1/OrganizationManagerSrv.hxx>   #include <Sirrix/-
Utils2/SharedPointer.hxx> #include <Sirrix/Utils2/Types.-
hxx>
```

Classes

- class __installedOrganizationInfo
- class turaya::organization::OrganizationManagerAdaptor

  Adaptor class to make the OrganizationManagerSrv class accessible over DBus.

Namespaces

- namespace turaya
- namespace turaya::organization

Typedefs

- typedef ::__installedOrganizationInfo turaya::organization::InstalledOrganization-Info

7.35 OrganizationManagerSrv.cxx File Reference

```
#include <sqlite3.h>     #include <Turaya/Database1/Data-
BaseUtils.hxx> #include <Turaya/Database1/DataBaseError.-
hxx>    #include <Turaya/OrganizationManagementService1/-
OrganizationExceptionsSrv.hxx> #include <Turaya/Organization-
ManagementService1/OrganizationManagerSrv.hxx>    #include
<Sirrix/Utils2/Debugging.hxx> #include <Sirrix/Hypervisor2/-
FileManager.hxx> #include <Turaya/TrustedVPNGeneralUtils1/-
Utilities.hxx>  #include <Turaya/TrustedVPNUtils1/Crypto-
Helper.hxx> #include <Turaya/TrustedVPNUtils1/Appliance.-
hxx> #include <Turaya/UserProxy1/UserManager.hxx> #include
<Turaya/UserProxy1/UserExceptions.hxx> #include <Turaya/-
ShareManagementService1/ShareManager.hxx>     #include <-
Turaya/ShareManagementService1/ShareExceptions.hxx> ×
#include <Sirrix/Hypervisor2/Privileges.hxx>
```

7.36 OrganizationManagerSrv.hxx File Reference

```
#include <stdio.h>  #include <string>  #include <Sirrix/-
Utils2/SharedPointer.hxx> #include <Sirrix/Utils2/Signal.-
hxx>    #include <Turaya/OrganizationManagementService1/-
OrganizationSrv.hxx> #include <Turaya/OrganizationProxy1/-
Organization.hxx>   #include <Turaya/OrganizationProxy1/-
OrganizationData.hxx> #include <Sirrix/Encoding2/Sequence.-
hxx> #include <Sirrix/Hypervisor2/Thread.hxx>
```

Classes

- class turaya::organization::OrganizationManagerSrv

  Server side OrganizationManager representation.
- class **turaya::organization::OrganizationManagerSrv::SetupShareManagerThread**
- class **turaya::organization::OrganizationManagerSrv::TearDownShareManagerThread**

Namespaces

- namespace turaya
- namespace turaya::organization

7.37 OrganizationSrv.cxx File Reference

```
#include <Turaya/OrganizationManagementService1/OrganizationSrv.hxx> #include <Turaya/OrganizationManagementService1/OrganizationExceptionsSrv.hxx> #include <sqlite3.h> × #include <Sirrix/Utils2/Debugging.hxx> #include <Turaya/Database1/DataBaseUtils.hxx> #include <Turaya/Database1/DataBaseError.hxx> #include <Sirrix/Hypervisor2/Path.hxx> #include <Turaya/CompartmentProxy1/CompartmentManager.hxx> #include <Turaya/CompartmentProxy1/CompartmentExceptions.hxx> #include <Turaya/PlatformManagement1/TrustedDesktop.hxx> #include <Sirrix/tss1/tddl/SocketTDDL.hxx>
```

7.38 OrganizationSrv.hxx File Reference

```
#include <string> #include <vector> #include <Turaya/OrganizationManagementService1/TOM.hxx> #include <Sirrix/Utils2/Types.hxx> #include <Sirrix/Utils2/Signal.hxx> × #include <Turaya/OrganizationProxy1/Organization.hxx> × #include <Turaya/CompartmentProxy1/CompartmentData.hxx> #include <Turaya/DomainProxy1/DomainData.hxx> #include <Sirrix/Hypervisor2/Thread.hxx> #include <Sirrix/Hypervisor2/Semaphore.hxx> #include <Turaya/OrganizationManagementService1/CompartmentManagerObserver.hxx>
```

Classes

- class turaya::organization::OrganizationSrv

  Server side organization representation.

Namespaces

- namespace turaya
- namespace turaya::organization

Functions

- const std::string ORGANIZATIONTABLE ("Companies")
- const std::string ORGANIZATION_TOM_TABLE ("Organization_Tom")

7.38.1 Function Documentation

7.38.1.1 const std::string ORGANIZATION_TOM_TABLE ("Organization_Tom")

7.38.1.2 const std::string ORGANIZATIONTABLE ("Companies")

7.39 Platform_Tests.cxx File Reference

```
#include <Sirrix/TestFramework1/Framework.hxx>   #include
<Sirrix/TestFramework1/HelperMacros.hxx>      #include <-
Turaya/PlatformManagement1/TrustedDesktop.hxx>   #include
<Sirrix/Utils2/Debugging.hxx>
```

Namespaces

- namespace unittests
- namespace unittests::PlatformManagement_Test

Functions

- unittests::PlatformManagement_Test::TEST_CASE (TrustedDesktop_get-PlatformID)
- unittests::PlatformManagement_Test::TEST_CASE (TrustedDesktop_get-FirmwareVersion)

7.40 PlatformManagementExceptions.hxx File Reference

```
#include <stdexcept>
```

Classes

- class turaya::PlatformManagementException

- class turaya::PlatformManagementObtainingSerialFailed
- class turaya::PlatformManagementObtainingFirmwareVersionFailed

Namespaces

- namespace turaya

7.41 PluginNotFoundException.hxx File Reference

```
#include <stdexcept> #include <string>
```

Classes

- class turaya::tcd2::PluginNotFoundException

Namespaces

- namespace turaya
- namespace turaya::tcd2

7.42 SambaAccess.cxx File Reference

```
#include <Turaya/CompartmentManagementService1/Samba-
Access.hxx> #include <Sirrix/Utils2/Signal.hxx> #include
<libsmbclient.h> #include <errno.h> #include <sstream>
#include <cstring> #include <Sirrix/Utils2/Debugging.-
hxx>
```

7.43 SambaAccess.hxx File Reference

```
#include <Sirrix/Hypervisor2/Path.hxx> #include <Sirrix/-
Utils2/Delegate.hxx> #include <Sirrix/Utils2/Types.hxx>
#include <string> #include <stdexcept>
```

Classes

- class turaya::compartment::SambaException
- class turaya::compartment::SambaAccess

Namespaces

- namespace turaya
- namespace turaya::compartment

7.44 TCDJobFactory.cxx File Reference

```
#include <Turaya/TrustedChannelService2/TCDJobFactory.-
hxx>
```

7.45 TCDJobFactory.hxx File Reference

```
#include <string> #include <map> #include <Sirrix/Utils2/-
Types.hxx> #include <Sirrix/Utils2/Debugging.hxx> #include
<Sirrix/Utils2/BaseOption.hxx> #include <Sirrix/Dispatcher1/-
Dispatcher.hxx> #include <Turaya/TrustedChannelService2/-
TCDRootJob.hxx> #include <Turaya/TrustedChannelService2/-
TCDPluginFactory.hxx>
```

Classes

- class turaya::tcd2::TCDJobFactory

Namespaces

- namespace turaya
- namespace turaya::tcd2

Typedefs

- typedef std::map< string, TCDJobFactory * > turaya::tcd2::ModuleList
- typedef std::map< string, TCDPluginFactory * > turaya::tcd2::PluginList

7.46 TCDPluginFactory.cxx File Reference

```
#include <Turaya/TrustedChannelService2/TCDPluginFactory.-
hxx>
```

7.47 TCDPluginFactory.hxx File Reference

```
#include <string> #include <map> #include <Sirrix/Utils2/-
Types.hxx> #include <Sirrix/Utils2/Debugging.hxx> #include
<Sirrix/Utils2/BaseOption.hxx>  #include <Turaya/Trusted-
ChannelService2/TCDPluginJob.hxx>
```

Classes

- class turaya::tcd2::TCDPluginFactory

Namespaces

- namespace turaya
- namespace turaya::tcd2

7.48 TCDPluginJob.cxx File Reference

```
#include <Turaya/TrustedChannelService2/TCDPluginJob.-
hxx>
```

7.49 TCDPluginJob.hxx File Reference

```
#include <Sirrix/Utils2/Types.hxx>     #include <Sirrix/-
Utils2/Debugging.hxx> #include <Sirrix/Encoding2/Object.-
hxx> #include <Sirrix/Dispatcher1/ThreadedJob.hxx> #include
<Sirrix/Dispatcher1/JobType.hxx> #include <Sirrix/Dispatcher1/-
JobInstance.hxx>    #include <Sirrix/Hypervisor2/Thread.-
hxx>   #include <Turaya/TrustedChannelService2/TCDPlugin-
Factory.hxx>
```

Classes

- class turaya::tcd2::TCDPluginJob

Namespaces

- namespace turaya
- namespace turaya::tcd2

7.50 TCDRootJob.cxx File Reference

```
#include <Turaya/TrustedChannelService2/TCDRootJob.hxx>
#include <Sirrix/Hypervisor2/System.hxx>
```

7.51 TCDRootJob.hxx File Reference

```
#include <Sirrix/Dispatcher1/ThreadedJob.hxx> #include <-
Sirrix/Dispatcher1/JobType.hxx> #include <Sirrix/Dispatcher1/-
JobInstance.hxx>  #include <Sirrix/Encoding2/Object.hxx>
#include <Sirrix/Encoding2/Sequence.hxx>  #include <-
Sirrix/Utils2/Types.hxx> #include <Sirrix/Utils2/Debugging.-
hxx>  #include <Sirrix/Utils2/ByteVector.hxx>  #include
<Sirrix/Utils2/CodeBookEntrys.hxx>  #include <Turaya/-
TrustedChannelService2/TCDPluginFactory.hxx>  #include <-
Turaya/TrustedChannelService2/TCDJobFactory.hxx>
```

Classes

- class turaya::tcd2::TCDRootJob
- class turaya::tcd2::TCDRootJob::ClientStateThread
- class turaya::tcd2::TCDRootJob::ServerStateThread

Namespaces

- namespace turaya
- namespace turaya::tcd2

7.52 TCReadThread.cxx File Reference

```
#include <iostream>  #include <Turaya/TrustedChannel-
Service2/TCReadThread.hxx>  #include <Sirrix/Utils2/Byte-
Vector.hxx> #include <Sirrix/Utils2/Debugging.hxx>
```

7.53 TCReadThread.hxx File Reference

```
#include <Sirrix/TrustedChannel1/BinaryTrustedChannel.-
hxx> #include <Sirrix/Dispatcher1/Dispatcher.hxx> #include
<Sirrix/Hypervisor2/Thread.hxx> #include <Sirrix/Hypervisor2/-
Semaphore.hxx> #include <Sirrix/Utils2/Signal.hxx>
```

Classes

- class turaya::tcd2::TCReadThread

Namespaces

- namespace turaya
- namespace turaya::tcd2

7.54 TCWriteThread.cxx File Reference

```
#include <Turaya/TrustedChannelService2/TCWriteThread.-
hxx> #include <Sirrix/Utils2/ByteVector.hxx> #include <-
Sirrix/Utils2/Debugging.hxx>
```

7.55 TCWriteThread.hxx File Reference

```
#include <Sirrix/TrustedChannel1/BinaryTrustedChannel.-
hxx> #include <Sirrix/Dispatcher1/Dispatcher.hxx> #include
<Sirrix/Hypervisor2/Thread.hxx>
```

Classes

- class turaya::tcd2::TCWriteThread

Namespaces

- namespace turaya
- namespace turaya::tcd2

7.56 TOM.cxx File Reference

```
#include <Turaya/OrganizationManagementService1/TOM.-
hxx> #include <Turaya/Database1/DataBaseUtils.hxx> #include
<Sirrix/Utils2/CentralCodeBook.hxx>    #include <Turaya/-
TrustedVPNUtils1/Appliance.hxx> #include <Turaya/Trusted-
VPNGeneralUtils1/Utilities.hxx> #include <Sirrix/Utils2/-
Debugging.hxx>   #include <Sirrix/Hypervisor2/Semaphore.-
hxx> #include <Sirrix/Hypervisor2/FileManager.hxx> #include
```

```
<Turaya/OrganizationManagementService1/OrganizationExceptions-
Srv.hxx> #include <Turaya/TrustedDesktopJobs1/GetAvailable-
CompartmentsJob.hxx>     #include <Turaya/TrustedDesktop-
Jobs1/GetDomainDataJob.hxx>     #include <Turaya/Trusted-
DesktopJobs1/CompartmentRequestJob.hxx> #include <Turaya/-
TrustedDesktopJobs1/ShareRequestJob.hxx> #include <Turaya/-
TrustedDesktopJobs1/AuthenticateUserJob.hxx>  #include <-
Turaya/TrustedDesktopJobs1/GetCompartmentJob.hxx> #include
<Turaya/CompartmentProxy1/CompartmentExceptions.hxx> ×
#include <Turaya/DomainProxy1/DomainExceptions.hxx> ×
#include <Turaya/OrganizationProxy1/Organization.hxx> ×
#include <Turaya/PlatformManagement1/TrustedDesktop.-
hxx>  #include <iostream>  #include <sstream>  #include
<stdlib.h>
```

7.57 TOM.hxx File Reference

```
#include <Sirrix/TrustedChannel1/TokenChannelConfig.-
hxx>  #include <Sirrix/TrustedChannel1/TPMChannelConfig.-
hxx>  #include <string>  #include <Sirrix/Utils2/Types.-
hxx>   #include <Turaya/OrganizationProxy1/TOMData.hxx> ×
#include <Sirrix/Utils2/SharedPointer.hxx>    #include <-
Sirrix/TrustedChannel1/BinaryTrustedChannel.hxx> #include
<Sirrix/Dispatcher1/Dispatcher.hxx>    #include <Turaya/-
MgmtProtocolRoot1/TCReadThread.hxx>    #include <Turaya/-
MgmtProtocolRoot1/TCWriteThread.hxx>   #include <Turaya/-
MgmtProtocolRoot1/ApplianceRootJob.hxx> #include <Turaya/-
OrganizationProxy1/Organization.hxx>   #include <Turaya/-
UserProxy1/UserData.hxx>   #include <Sirrix/Hypervisor2/-
Thread.hxx> #include <Sirrix/Utils2/Signal.hxx> #include
<Sirrix/Hypervisor2/BinarySocketStream.hxx>   #include <-
Sirrix/Hypervisor2/Path.hxx> #include <Turaya/ShareManagement-
Service1/Share.hxx> #include <Turaya/TrustedDesktopJobs1/-
TrustedDesktopJob.hxx>   #include <Turaya/TrustedDesktop-
Jobs1/VPNJob.hxx> #include <Turaya/OrganizationManagement-
Service1/NetworkManager.hxx>
```

Classes

- class turaya::organization::TOM

  Server side TOM representation This Class encapsulates the CommunicationChannel (TrustedChannel) to the Trusted Object Manager server.

Namespaces

- namespace turaya

- namespace turaya::organization

Functions

- const std::string TOMTABLE ("TOMs")

7.57.1 Function Documentation

7.57.1.1 const std::string TOMTABLE ("TOMs")

7.58 TrustedChannelDaemon2.cxx File Reference

```
#include <Turaya/TrustedChannelService2/TrustedChannel-
Daemon2.hxx>    #include <Sirrix/Utils2/CentralCodeBook.-
hxx> #include <Sirrix/Utils2/CodeBookEntrys.hxx> #include
<Sirrix/Hypervisor2/Thread.hxx>
```

7.59 TrustedChannelDaemon2.hxx File Reference

```
#include <string>   #include <list>   #include <Sirrix/-
TrustedChannel1/BinaryTrustedChannel.hxx>     #include <-
Turaya/TrustedChannelService2/TCDRootJob.hxx> #include <-
Turaya/TrustedChannelService2/TCReadThread.hxx>  #include
<Turaya/TrustedChannelService2/TCWriteThread.hxx>
```

Classes

- class turaya::tcd2::TrustedChannelDaemon2

Namespaces

- namespace turaya
- namespace turaya::tcd2

Defines

- #define turaya_tcd_TrustedChannelDaemon_hxx_included

7.59.1 Define Documentation

7.59.1.1 #define turaya_tcd_TrustedChannelDaemon_hxx_included

7.60 TrustedDesktop.cxx File Reference

```
#include <sys/stat.h> #include <Sirrix/Utils2/FileBinary-
Stream.hxx> #include <Turaya/PlatformManagement1/Trusted-
Desktop.hxx> #include <Turaya/PlatformManagement1/Platform-
ManagementExceptions.hxx>
```

Variables

- const std::string myPlatformSerial = "/etc/sirrix/general/serial"
- const std::string myProductVersion = "/etc/sirrix.version"

7.60.1 Variable Documentation

7.60.1.1 const std::string myPlatformSerial = "/etc/sirrix/general/serial"

7.60.1.2 const std::string myProductVersion = "/etc/sirrix.version"

7.61 TrustedDesktop.hxx File Reference

```
#include <string> #include <sstream> #include <dbus++/dbus.-
hpp> #include <Sirrix/Utils2/Types.hxx> #include <Sirrix/-
Utils2/Debugging.hxx> #include <Sirrix/Utils2/Singleton.-
hxx> #include <Sirrix/Utils2/ByteVector.hxx>
```

Classes

- class turaya::Version
- class turaya::FirmwareVersion
- class turaya::TrustedDesktop

  Represents a concrete TrustedDesktop.

Namespaces

- namespace turaya

Typedefs

- typedef std::string turaya::PlatformID
- typedef sirrix::utils::Singleton < TrustedDesktop > turaya::CentralTrusted-Desktop

7.62 UserAdaptor.cxx File Reference

```
#include <Turaya/UserManagementService1/UserAdaptor.-
hxx>    #include <Turaya/UserProxy1/UserExceptions.hxx> ×
#include <Sirrix/Utils2/Debugging.hxx> #include <Sirrix/-
Utils2/ByteVector.hxx>  #include <Turaya/UserProxy1/User-
Data.hxx> #include <stdio.h> #include <string> #include
<vector> #include <sstream> #include <iostream>
```

Functions

- const std::string USER_INTERFACE ("turaya.user.user")

7.62.1 Function Documentation

7.62.1.1 const std::string USER_INTERFACE ("turaya.user.user")

7.63 UserAdaptor.hxx File Reference

```
#include <string> #include <vector> #include <dbus++/object.-
hpp> #include <dbus++/variant.hpp> #include <dbus++/fixed-
_array.hpp>  #include <dbus++/dbus_type_cast_traits.hpp>
#include <Sirrix/Utils2/Signal.hxx>    #include <Turaya/-
UserManagementService1/UserSrv.hxx>    #include <Turaya/-
UserManagementService1/UserExceptionsSrv.hxx> #include <-
Turaya/UserProxy1/UserData.hxx>
```

Classes

- class turaya::user::UserAdaptor

  Adaptor class to make the UserSrv class accessible over DBus.

Namespaces

- namespace turaya
- namespace turaya::user

7.64 UserExceptionsSrv.hxx File Reference

```
#include <stdexcept>
```

Classes

- class turaya::user::UserSrvException
- class turaya::user::UserSrvAlreadyExists
- class turaya::user::UserSrvNotFound
- class turaya::user::UserSrvInvalidUserData
- class turaya::user::UserSrvWrongState

Namespaces

- namespace turaya
- namespace turaya::user

7.65 UserHypervisor.cxx File Reference

```
#include <Turaya/UserManagementService1/UserHypervisor.-
hxx>    #include <Turaya/UserManagementService1/UserSrv.-
hxx> #include <Turaya/UserManagementService1/UserManager-
Srv.hxx>    #include <Turaya/TrustedVPNUtils1/Appliance.-
hxx>  #include <Sirrix/Utils2/Debugging.hxx>  #include <-
Sirrix/Hypervisor2/Mutex.hxx> #include <Sirrix/Hypervisor2/-
ScopedMutex.hxx> #include <Sirrix/Hypervisor2/FileManager.-
hxx>   #include <stdio.h>   #include <stdlib.h>   #include
<sys/types.h>  #include <sys/stat.h>  #include <pwd.h> ×
#include <shadow.h>  #include <grp.h>  #include <paths.h>
#include <errno.h>  #include <string.h>  #include <time.-
h> #include <string> #include <vector> #include <fstream> ×
#include <sstream> #include <set>
```

7.66 UserHypervisor.hxx File Reference

```
#include <stdio.h>  #include <string>  #include <Sirrix/-
Utils2/SharedPointer.hxx>    #include <Turaya/UserProxy1/-
UserData.hxx>    #include <Turaya/UserManagementService1/-
UserSrv.hxx>    #include <Turaya/UserManagementService1/-
UserManagerSrv.hxx> #include <Turaya/UserManagementService1/-
UserExceptionsSrv.hxx>
```

Classes

- class turaya::user::UserHypervisor

  Abstraction of platform specific user functions.

Namespaces

- namespace turaya
- namespace turaya::user

7.67 UserManagerAdaptor.cxx File Reference

```
#include <Turaya/UserManagementService1/UserManagerAdaptor.-
hxx> #include <Turaya/UserManagementService1/UserExceptions-
Srv.hxx>     #include <Turaya/UserProxy1/UserExceptions.-
hxx>  #include <Turaya/UserProxy1/UserData.hxx>  #include
<Turaya/UserProxy1/UserProxyWrapper.hxx>      #include <-
Sirrix/Utils2/Debugging.hxx>    #include <Sirrix/Utils2/-
ByteVector.hxx> #include <iostream>
```

Functions

- const std::string USER_MANAGER_INTERFACE ("turaya.user.manager")
- const std::string USERS_PATH ("/users/User_")

7.67.1 Function Documentation

7.67.1.1 const std::string USER_MANAGER_INTERFACE ("turaya.user.manager")

7.67.1.2 const std::string USERS_PATH ("/users/User_")

7.68 UserManagerAdaptor.hxx File Reference

```
#include <dbus++/object.hpp>   #include <dbus++/variant.-
hpp> #include <dbus++/fixed_array.hpp> #include <dbus++/dbus-
_type_cast_traits.hpp>  #include <dbus++/exception.hpp> x
#include <Sirrix/Utils2/SharedPointer.hxx>    #include <-
Sirrix/Utils2/Types.hxx> #include <Turaya/UserManagement-
Service1/UserAdaptor.hxx> #include <Turaya/UserManagement-
Service1/UserManagerSrv.hxx> #include <Turaya/UserProxy1/-
UserProxyWrapper.hxx>
```

Classes

- class __installedUserInfo
- class turaya::user::UserManagerAdaptor

  Adaptor class to make the UserManagerSrv class accessible over DBus.

Namespaces

- namespace turaya
- namespace turaya::user

Typedefs

- typedef ::__installedUserInfo turaya::user::InstalledUserInfo

7.69 UserManagerSrv.cxx File Reference

```
#include <Turaya/UserManagementService1/UserManagerSrv.-
hxx> #include <Turaya/UserManagementService1/UserHypervisor.-
hxx>   #include <Sirrix/Utils2/Debugging.hxx>   #include
<Sirrix/Utils2/ShellCommand.hxx>    #include <stdio.h> ×
#include <stdlib.h>  #include <time.h>  #include <pwd.h>
#include <string> #include <vector> #include <fstream> ×
#include <sstream> #include <set>
```

7.70 UserManagerSrv.hxx File Reference

```
#include <stdio.h>  #include <string>  #include <Sirrix/-
Utils2/SharedPointer.hxx>   #include <Turaya/UserProxy1/-
UserManager.hxx>   #include <Turaya/UserProxy1/UserData.-
hxx>   #include <Turaya/UserManagementService1/UserSrv.-
hxx> #include <Turaya/UserManagementService1/UserExceptions-
Srv.hxx>
```

Classes

- class turaya::user::UserManagerSrv

  Server side UserManager representation.

Namespaces

- namespace turaya
- namespace turaya::user

7.71 UserSrv.cxx File Reference

```
#include <sstream> #include <Sirrix/Hypervisor2/rndstream.-
hxx>  #include <Turaya/UserProxy1/UserData.hxx>  #include
```

<Turaya/UserManagementService1/UserSrv.hxx> #include <-
Turaya/UserManagementService1/UserHypervisor.hxx> #include
<Turaya/UserManagementService1/UserExceptionsSrv.hxx> ×
#include <Sirrix/Utils2/Debugging.hxx> #include <Sirrix/-
Utils2/ByteVector.hxx>

7.72 UserSrv.hxx File Reference

#include <string> #include <vector> #include <Sirrix/-
Utils2/Types.hxx> #include <Sirrix/Utils2/Signal.hxx>
#include <Sirrix/Utils2/ByteVector.hxx> #include <Turaya/-
UserProxy1/User.hxx> #include <Turaya/UserProxy1/User-
Data.hxx>

Classes

- class turaya::user::UserSrv

  Server side user representation.

Namespaces

- namespace turaya
- namespace turaya::user

Functions

- const std::string USERTABLE ("Users")

7.72.1 Function Documentation

7.72.1.1 const std::string USERTABLE ("Users")

7.73 VMOptionParser.cxx File Reference

#include <Turaya/CompartmentManagementService1/VMOption-
Parser.hxx> #include <Sirrix/Utils2/CmdLineParser.hxx> ×
#include <Sirrix/Utils2/Debugging.hxx>

Variables

- const string _audioHW_range_ [] = {"ac97", "hda", "sb16"}
- const string _audioIF_range_ [] = {"none", "null", "oss", "alsa", "pulse"}

- const string _nicHW_range_ [] = {"Am79C970A", "Am79C973", "82540EM", "82543GC", "82545EM"}
- const UInt32 _mem_range_ [] = {64, 128, 256, 512, 768, 1024, 1280, 1536, 1792, 2048}
- const UInt32 _vmem_range_ [] = {8, 16, 32, 64, 128, 256, 512}

7.73.1 Variable Documentation

7.73.1.1 const string _audioHW_range_[] = {"ac97", "hda", "sb16"}

7.73.1.2 const string _audioIF_range_[] = {"none", "null", "oss", "alsa", "pulse"}

7.73.1.3 const UInt32 _mem_range_[] = {64, 128, 256, 512, 768, 1024, 1280, 1536, 1792, 2048}

7.73.1.4 const string _nicHW_range_[] = {"Am79C970A", "Am79C973", "82540EM", "82543GC", "82545EM"}

7.73.1.5 const UInt32 _vmem_range_[] = {8, 16, 32, 64, 128, 256, 512}

7.74 VMOptionParser.hxx File Reference

```
#include <string> #include <set> #include <Sirrix/Utils2/-
Types.hxx>
```

Classes

- class turaya::compartment::VMOptionParser

Namespaces

- namespace turaya
- namespace turaya::compartment

7.75 VMOptionParser_Tests.cxx File Reference

```
#include <Sirrix/TestFramework1/Framework.hxx>  #include
<Sirrix/TestFramework1/HelperMacros.hxx>  #include <-
Sirrix/Utils2/Debugging.hxx> #include <Turaya/Compartment-
ManagementService1/VMOptionParser.hxx>
```

Namespaces

- namespace unittests
- namespace unittests::VMOptionParser_Tests

Functions

- unittests::VMOptionParser_Tests::TEST_CASE (EmptyOption_Test)
- unittests::VMOptionParser_Tests::TEST_CASE (SomeOptions_Test)
- unittests::VMOptionParser_Tests::TEST_CASE (AllOptions_TEST)
- unittests::VMOptionParser_Tests::TEST_CASE (wrongFormat_TEST)
- unittests::VMOptionParser_Tests::TEST_CASE (outOfRange_TEST)